

The
Pragmatic
Programmers

HTML5 & CSS3

Webentwicklung mit den
Standards von morgen

Deutsche Übersetzung von
O'REILLY[®]

Brian P. Hogan
Übersetzt von Stefan Fröhlich



HTML5 & CSS3

Webentwicklung mit den Standards von morgen

HTML5 & CSS3

Webentwicklung mit den Standards von morgen

Brian P. Hogan

Deutsche Übersetzung von
Stefan Fröhlich

O'REILLY®

Beijing · Cambridge · Farnham · Köln · Sebastopol · Tokyo

Die Informationen in diesem Buch wurden mit größter Sorgfalt erarbeitet. Dennoch können Fehler nicht vollständig ausgeschlossen werden. Verlag, Autoren und Übersetzer übernehmen keine juristische Verantwortung oder irgendeine Haftung für eventuell verbliebene Fehler und deren Folgen.

Alle Warennamen werden ohne Gewährleistung der freien Verwendbarkeit benutzt und sind möglicherweise eingetragene Warenzeichen. Der Verlag richtet sich im Wesentlichen nach den Schreibweisen der Hersteller. Das Werk einschließlich aller seiner Teile ist urheberrechtlich geschützt. Alle Rechte vorbehalten einschließlich der Vervielfältigung, Übersetzung, Mikroverfilmung sowie Einspeicherung und Verarbeitung in elektronischen Systemen.

Kommentare und Fragen können Sie gerne an uns richten:

O'Reilly Verlag
Balthasarstr. 81
50670 Köln
E-Mail: kommentar@oreilly.de

Copyright der deutschen Ausgabe:
© 2011 by O'Reilly Verlag GmbH & Co. KG
1. Auflage 2011

Die Originalausgabe erschien 2011 unter dem Titel
HTML5 and CSS3 bei Pragmatic Bookshelf, Inc.

Bibliografische Information der Deutschen Nationalbibliothek
Die Deutsche Nationalbibliothek verzeichnet diese Publikation in der
Deutschen Nationalbibliografie; detaillierte bibliografische Daten
sind im Internet über <http://dnb.d-nb.de> abrufbar.

Übersetzung und deutsche Bearbeitung: Stefan Fröhlich, Berlin
Lektorat: Inken Kiupel, Köln
Korrektur: Friederike Daenecke, Zülpich
DTP: Andreas Franke, SatzWERK, Siegen; www.satz-werk.com
Produktion: Karin Driesen, Köln
Belichtung, Druck und buchbinderische Verarbeitung:
Druckerei Kösel, Krugzell; www.koeselbuch.de

ISBN 978-3-89721-316-6

Dieses Buch ist auf 100% chlorfrei gebleichtem Papier gedruckt.

Inhaltsverzeichnis

Danksagungen	1
Vorwort	3
HTML5: Plattform oder Spezifikation?	4
Wie die Inhalte organisiert sind	4
Über dieses Buch	5
Vorkenntnisse	6
Online-Ressourcen	6
1 HTML5 und CSS3 im Überblick	9
1.1 Eine Plattform für die Webentwicklung.	9
1.2 Abwärtskompatibilität	13
1.3 Der Weg in die Zukunft ist steinig	15
Teil I: Bessere Benutzeroberflächen	21
2 Neue strukturelle Tags und Attribute	23
Tipp 1 Einen Blog mit semantischem Markup neu definieren	26
Tipp 2 Pop-up-Fenster mit benutzerdefinierten Datenattributen	39
3 Benutzerfreundliche Webformulare	45
Tipp 3 Daten mit neuen Eingabefeldern beschreiben.	48
Tipp 4 Mit autofocus zum ersten Feld springen.	56
Tipp 5 Platzhaltertext für Hinweise nutzen	58
Tipp 6 In-Place-Editing mit contenteditable.	65

4	Bessere Benutzeroberflächen mit CSS3	73
	Tipp 7 Tabellen mit Pseudoklassen stylen	75
	Tipp 8 Links ausdrucken mit :after und content	85
	Tipp 9 Mehrspaltige Layouts	89
	Tipp 10 Benutzeroberflächen für mobile Geräte mit Media Queries.	95
5	Mehr Barrierefreiheit	99
	Tipp 11 Navigationshinweise mit ARIA-Rollen	101
	Tipp 12 Barrierefreie aktualisierbare Bereiche erstellen	106
	Teil II: Neue Perspektiven und Klänge	111
6	Zeichnen mit dem canvas-Element	113
	Tipp 13 Logos zeichnen	115
	Tipp 14 Statistiken grafisch darstellen mit RGraph	122
7	Audio und Video einbetten	131
	7.1 Ein bisschen Geschichte	132
	7.2 Container und Codecs	133
	Tipp 15 Mit Audio arbeiten	138
	Tipp 16 Video einbetten	142
8	Augenschmaus	149
	Tipp 17 Scharfe Ecken abrunden.	151
	Tipp 18 Schatten, Verläufe und Transformationen	159
	Tipp 19 Echte Schriften nutzen	169
	Teil III: Jenseits von HTML5	175
9	Mit clientseitigen Daten arbeiten	177
	Tipp 20 Einstellungen mit localStorage speichern	180
	Tipp 21 Daten in einer clientseitigen relationalen Datenbank speichern	186
	Tipp 22 Offline arbeiten	198
10	Mit anderen APIs spielen	201
	Tipp 23 Den Verlauf erhalten	203
	Tipp 24 Über Domains hinweg kommunizieren	206
	Tipp 25 Chatten mit Web Sockets	213
	Tipp 26 Finden Sie sich selbst: Geolocation	220

11 Wie es weitergeht	225
11.1 CSS3-Übergänge	226
11.2 Web Workers	229
11.3 Native Unterstützung für Drag-and-Drop	230
11.4 WebGL	236
11.5 Indexed Database API	237
11.6 Clientseitige Formularvalidierung	238
11.7 Vorwärts!	239
A Kurzreferenz	241
A.1 Neue Elemente	241
A.2 Attribute	242
A.3 Formulare	242
A.4 Attribute für Formularfelder	243
A.5 Barrierefreiheit.	244
A.6 Multimedia	244
A.7 CSS3	245
A.8 Clientseitige Speicherung	247
A.9 Zusätzliche APIs	247
B jQuery-Einführung	249
B.1 jQuery laden	249
B.2 jQuery-Grundlagen	250
B.3 Methoden zum Verändern von Inhalten	251
B.4 Elemente erstellen	254
B.5 Events	254
B.6 Die Funktion document.ready	255
C Audio und Video kodieren	257
C.1 Audio kodieren.	257
C.2 Video für das Web kodieren	258
D Ressourcen	259
D.1 Ressourcen im Web	259
E Literaturverzeichnis	263
Index	265

Danksagungen

Ich habe mich förmlich auf die Arbeit an diesem Buch gestürzt, noch bevor ich mit dem vorherigen fertig war. Die meisten Freunde, meine Familie und auch mein Verleger haben mich wahrscheinlich für verrückt erklärt, weil ich mir keine Pause gegönnt habe, aber sie haben mich dennoch unterstützt. Dieses Buch ist also das Ergebnis vieler wundervoller und hilfsbereiter Menschen.

Dave Thomas und Andy Hunt kann ich gar nicht genug dafür danken, dass sie mir die Möglichkeit gegeben haben, ein zweites Mal mit ihnen zusammenzuarbeiten. Ihre Anmerkungen haben mir sehr dabei geholfen, diesem Buch die richtige Form zu geben. Ich bin stolz darauf, ein Pragmatic Bookshelf-Autor zu sein.

Daniel Steinberg hat mir dabei geholfen, dieses Buch anzuschieben, unter Vertrag zu kommen und von Anfang an die richtigen Weichen zu stellen. Ich bin sehr dankbar für all die Unterstützung, die ich von ihm erhalten habe – und dafür, dass er mir beigebracht hat, meine Texte klarer zu formulieren. Auch jetzt habe ich beim Schreiben seine Stimme im Ohr.

Daniel konnte seine Arbeit an diesem Buch nicht fortsetzen, gab mich aber in wirklich gute Hände. Susannah Pfalzer hat mir während des ganzen Projekts unglaublich viel geholfen, mich bei der Stange gehalten und mich angetrieben, noch besser zu werden. Sie verstand es, immer genau zur richtigen Zeit die richtigen Fragen zu stellen. Ohne Susannah wäre dieses Buch nicht annähernd so gut geworden.

2 ► Danksagungen

Meine technischen Gutachter haben mir in beiden Feedback-Runden extrem viel dabei geholfen, Inhalte deutlicher herauszuarbeiten und verständlicher zu präsentieren. Vielen Dank an Aaron Godin, Ali Raza, Charles Leffingwell, Daniel Steinberg, David Kulberg, Don Henton, Doug Rhoten, Edi Schlechtinger, Jon Mischo, Jon Oebser, Kevin Gisi, Marc Harter, Mark Nichols, Noel Rappin, Paul Neibarger, Sam Elliott, Sean Canton, Srdjan Pejic, Stephen Wolff, Todd Dahl und Erik Watson.

Besonderer Dank gilt auch den tollen Leuten bei ZenCoder für ihre Hilfe bei der Videokodierung der Beispieldateien und dafür, dass sie es vielen Webautoren durch ihre Arbeit so viel leichter machen, Videos für HTML5 vorzubereiten.

Ein herzlicher Dank geht auch an meine Geschäftspartner Chris Johnson, Chris Warren, Mike Weber, Jon Kinney, Adam Ludwig, Gary Crabtree, Carl Hoover, Josh Anderson, Austen Ott und Nick Lamuro für die Unterstützung bei diesem und vielen anderen Projekten. Besonderer Dank gilt auch Erich Tesky dafür, dass er mich gelegentlich auf den Boden der Tatsachen zurückholte, und dafür, dass er für mich auch dann als Freund da war, wenn ich mal frustriert war.

Außerdem möchte ich meinem Vater dafür danken, dass er von mir immer nur das Beste erwartet und mich unermüdlich zum Durchhalten angetrieben hat. Erst dadurch wurde das Unmögliche möglich.

Mein ewiger Dank und meine Liebe gelten meiner wundervollen Frau Carissa und unseren Töchtern Ana und Lisa. Sie haben auf eine Menge Wochenenden und Abende verzichtet, damit ich im Büro weiterhacken und schreiben konnte. Jedes Mal, wenn ich nicht weiter wusste, hat mir Carissas unerschütterlicher Glaube, dass „ich es schon schaffe“, weitergeholfen. Ich habe großes Glück, dass die drei ein Teil meines Lebens sind.

Vorwort

Drei Monate im Web sind wie ein ganzes Jahr in Echtzeit.

Wir Webentwickler denken meistens so, weil wir ständig von irgendwas Neuem hören. Vor einem Jahr schienen HTML5 und CSS3 noch in weiter Ferne zu sein. Aber schon heute setzen Unternehmen diese Technologien ein, weil in Browsern wie Google Chrome, Safari, Firefox und Opera bereits Teile der Spezifikation implementiert wurden.

HTML5 und CSS3 helfen dabei, das Fundament für die nächste Generation von Webanwendungen zu legen. Sie geben uns die Möglichkeit, Websites zu erstellen, die leichter zu entwickeln, zu pflegen und die benutzerfreundlicher sind. HTML5 bietet neue Elemente für die Definition der Struktur einer Website und die Einbettung von Inhalten. Daher müssen wir nicht mehr auf zusätzliches Markup oder extra Plugins zurückgreifen. CSS3 bietet fortschrittliche Selektoren, grafische Erweiterungen und bessere Unterstützung für Schriften. Dadurch werden Websites visuell ansprechender, auch ohne die Ersetzung von Schriften durch Grafiken, komplexe JavaScripts oder grafische Tools. Die erweiterte Unterstützung für Barrierefreiheit ermöglicht bessere Ajax-Anwendungen für behinderte Menschen, und dank der Offline-Möglichkeiten können wir mit der Entwicklung von Anwendungen beginnen, die auch ohne Internetverbindung funktionieren.

In diesem Buch erfahren Sie alles darüber, was Sie bereits jetzt mit HTML5 und CSS3 machen können, selbst wenn Ihre Benutzer Browser verwenden, die noch nicht alle Funktionen unterstützen. Bevor wir damit anfangen, nehmen wir uns ein paar Sekunden Zeit und sprechen über HTML5 und andere Buzzwords.

HTML5: Plattform oder Spezifikation?

HTML5 ist eine Spezifikation, die einige neue Tags und neues Markup sowie wundervolle JavaScript-APIs beschreibt, aber gleichzeitig ist es in einen Sog aus Hypes und Versprechungen geraten. Leider hat sich der HTML5-Standard zu einer HTML5-Plattform entwickelt, was zu einer schrecklichen Verwirrung bei Entwicklern, Kunden und sogar bei Autoren führt. In manchen Fällen werden Teile der CSS3-Spezifikation wie etwa Schatten, Verläufe und Transformationen plötzlich „HTML“ genannt. Browserhersteller versuchen, sich gegenseitig darin zu übertrumpfen, wie viel „HTML5“ sie unterstützen. Und Kunden machen plötzlich komische Anfragen wie zum Beispiel: „Meine Website wird doch in HTML5 gemacht, oder?“

Für den größten Teil dieses Buches konzentrieren wir uns auf die Spezifikationen von HTML5 und CSS3 sowie darauf, wie Sie die damit beschriebenen Techniken nutzen können. Im letzten Teil des Buches sehen wir uns eine Reihe verwandter Spezifikationen an, die einmal Teil von HTML5 waren, aber bereits jetzt auf verschiedenen Plattformen im Einsatz sind. Dazu gehören Web SQL Databases, Geolocation und Web Sockets. Zwar gehören diese Dinge *technisch gesehen* nicht zu HTML5, können Ihnen aber in Kombination mit HTML5 und CSS3 dabei helfen, tolle Anwendungen zu erstellen.

Wie die Inhalte organisiert sind

Jedes Kapitel in diesem Buch konzentriert sich auf eine bestimmte Gruppe von Problemen, die wir mit HTML5 und CSS3 lösen können. Für jedes Kapitel gibt es einen Überblick sowie eine Tabelle mit den im Buch behandelten Tags, Funktionen und Konzepten. Der Hauptinhalt jedes Kapitels ist in „Tipps“ unterteilt, die Ihnen jeweils ein bestimmtes Konzept vorstellen und Sie durch die Erstellung eines einfachen Beispiels führen. Die Kapitel dieses Buchs wurden themenweise zusammengestellt. Anstatt die Themen in einen HTML5- und einen CSS3-Teil zu gliedern, ist es sinnvoller, die Kapitel anhand der gelösten Probleme zusammenzufassen.

Jeder Tipp enthält einen Abschnitt mit der Überschrift „Ausweichlösung“, in dem Lösungen für diejenigen Benutzer gezeigt werden, deren Browser noch keine Unterstützung für HTML5 und CSS3 bieten. Wir werden eine Vielzahl von Techniken verwenden, damit diese Ausweichlösungen funktionieren: von Bibliotheken von Drittherstellern bis hin zu unseren eigenen jQuery-Plugins. Die Tipps können Sie in beliebiger Reihenfolge lesen.

Jedes Kapitel schließt mit einem Abschnitt mit der Überschrift „Die Zukunft“, in dem wir diskutieren, wie das jeweilige Konzept angewendet werden kann, wenn es weitere Verbreitung findet.

Dieses Buch konzentriert sich auf das, was Sie heute schon verwenden können. Es gibt mehr HTML5- und CSS3-Funktionen, deren Verwendung noch nicht weit verbreitet ist. Darüber erfahren Sie mehr in Kapitel 11, *Wie es weitergeht*, auf Seite 225.

Über dieses Buch

Wir beginnen mit einem kurzen Überblick über HTML5 und CSS3 und sehen uns einige der neuen strukturellen Tags an, mit denen Sie den Inhalt Ihrer Seiten beschreiben können. Anschließend arbeiten wir mit Formularen, und Sie erhalten die Gelegenheit, einige Formularfelder und -funktionen wie zum Beispiel Autofokus und Platzhalter zu verwenden. Danach können Sie mit den neuen Selektoren von CSS3 spielen und lernen, wie Sie Stilregeln auf Elemente anwenden, ohne Ihrem Inhalt zusätzliches Markup hinzuzufügen.

Anschließend erkunden wir die Audio- und Videounterstützung in HTML5. Dabei lernen Sie, wie Sie mit dem canvas-Element Formen zeichnen. Außerdem werfen Sie einen Blick auf Schatten, Verläufe und Transformationen in CSS3 und lernen, wie Sie mit Schriften arbeiten.

Im letzten Abschnitt verwenden wir die clientseitigen Funktionen von HTML5, wie zum Beispiel Web Storage und Web SQL Databases, und behandeln die Erstellung von clientseitigen Offline-Anwendungen. Wir setzen Web Sockets ein, um uns mit einem einfachen Chat-Dienst zu unterhalten. Außerdem erfahren Sie, wie es mit HTML5 möglich ist, Nachrichten und Daten über Domains hinweg zu versenden. Sie spielen mit der Geolocation-API und manipulieren den Browserverlauf. Zum Abschluss werfen wir einen Blick auf einige Dinge, die zwar nicht jetzt gleich nützlich sind, aber in naher Zukunft wichtig sein werden.

In Anhang A auf Seite 241 finden Sie eine Auflistung aller in diesem Buch behandelten Funktionen mit Querverweisen auf die entsprechenden Kapitel. Wir arbeiten in diesem Buch häufig mit jQuery, deshalb gibt Ihnen Anhang B ab Seite 249 einen kurzen Überblick. Außerdem finden Sie hier Informationen zur Kodierung von Audio- und Videodateien für HTML5.

Vorkenntnisse

Dieses Buch richtet sich in erster Linie an Webentwickler, die ein solides Verständnis von HTML und CSS mitbringen. Wenn Sie gerade erst anfangen, ist dieses Buch trotzdem nützlich für Sie. Ich würde Ihnen dann aber die Bücher *Webdesign mit Webstandards* [Zel08] und *Web Design for Developers* [Hog09] ans Herz legen.

Ich gehe außerdem davon aus, dass Sie über ein grundlegendes Verständnis von JavaScript und jQuery¹ verfügen, denn damit implementieren wir viele unserer Ausweidlösungen. Anhang B auf Seite 249 gibt eine kurze Einführung in jQuery und die grundlegenden Methoden, die wir verwenden werden.

Sie brauchen Firefox 3.6, Google Chrome 5, Opera 10.6 oder Safari 5, um den Code in diesem Buch zu testen. Wahrscheinlich brauchen Sie sogar jeden dieser Browser, um auch alles zu testen, was wir schreiben. Denn jeder Browser macht alles ein bisschen anders.

Außerdem brauchen Sie eine Möglichkeit, Ihre Websites mit dem Internet Explorer zu testen. Nur so können Sie sichergehen, dass unsere Ausweidlösungen auch wirklich funktionieren. Wenn Sie Ihre Beispiele in mehreren Versionen des Internet Explorer testen müssen, können Sie IETester für Windows herunterladen. Die Software unterstützt IE 6, 7 und 8 in einer einzigen Anwendung. Wenn Sie nicht Windows verwenden, sollten Sie darüber nachdenken, eine virtuelle Maschine wie z.B. Virtual Box oder VMware oder einen Service wie etwa CrossBrowserTesting² oder MogoTest³ zu verwenden.

Online-Ressourcen

Auf der englischen Website zum Buch⁴ finden Sie Links auf ein interaktives Diskussionsforum sowie zu den Errata dieses Buchs. Dort finden Sie auch einen Link zum Quellcode für alle Beispiele in diesem Buch. Außerdem können Leser des E-Books auf den grauen Kasten oberhalb der Codeausschnitte klicken, um die Snippets direkt herunterzuladen.

1 <http://www.jquery.com>

2 <http://crossbrowsertesting.com/>

3 <http://www.mogotest.com/>

4 <http://www.pragprog.com/titles/bhh5/>

Sollten Sie einen Fehler finden, erstellen Sie bitte einen Eintrag auf der Seite mit den Errata, damit wir uns darum kümmern können. Wenn Sie eine elektronische Version dieses Buchs lesen, finden Sie in der Fußzeile jeder Seite Links, über die Sie ganz einfach Errata übermitteln können.

Außerdem sollten Sie unbedingt den Blog zu diesem Buch besuchen, „Beyond HTML5 and CSS3“.⁵ Ich veröffentliche dort entsprechendes Material, Aktualisierungen und funktionierende Beispiele zu diesem Buch.

Sind Sie bereit? Super! Legen wir los mit HTML5 und CSS3.

5 <http://www.beyondhtml5andcss3.com/>

Kapitel 1

HTML5 und CSS3 im Überblick

HTML5¹ und CSS3² sind mehr als lediglich zwei neue Standards, die vom World Wide Web Consortium (W3C) und seinen Arbeitsgruppen vorgeschlagen wurden. Für die Technologien, die Sie jeden Tag verwenden, sind die beiden Standards der nächste Schritt und sollen Ihnen dabei helfen, bessere und modernere Webanwendungen zu entwickeln. Bevor wir tief in die Details von HTML5 und CSS3 einsteigen, sprechen wir zunächst über einige Vorteile von HTML5 und CSS3 sowie über einige Herausforderungen, die sich daraus ergeben.

1.1 Eine Plattform für die Webentwicklung

Eine Menge neuer Funktionen in HTML drehen sich darum, eine bessere Plattform für webbasierte Anwendungen zu entwickeln. Von aussagekräftigeren Tags, besserer Cross-site- und Cross-window-Kommunikation, bis hin zu Animationen und verbesserter Multimedia-Unterstützung – mit HTML5 stehen Entwicklern eine Menge neuer Tools zur Verfügung, die der Verbesserung der User Experience dienen.

1 Die HTML5-Spezifikation finden Sie unter <http://www.w3.org/TR/html5/>.

2 CSS3 ist auf mehrere Module verteilt. Den aktuellen Status können Sie unter <http://www.w3.org/Style/CSS/current-work> mitverfolgen.

Aussagekräftigeres Markup

In jeder HTML-Version wird neues Markup eingeführt. Aber nie zuvor gab es so viele Erweiterungen, die sich direkt auf die Beschreibung von Inhalten beziehen. Elemente für die Definition von Überschriften, Fußzeilen, Navigationsabschnitten, Seitenleisten und Artikeln lernen Sie in Kapitel 2, *Neue strukturelle Tags und Attribute*, ab Seite 23 kennen. Dort behandeln wir auch Messwerte und Fortschrittsbalken und erklären, wie Ihnen benutzerdefinierte Datenattribute dabei helfen können, Daten in Markup abzubilden.

Multimedia mit weniger Plugins

Sie brauchen kein Flash und auch kein Silverlight mehr für Video, Audio und Vektorgrafiken. Flash-basierte Videoplayer sind zwar relativ einfach zu verwenden, funktionieren aber nicht auf den mobilen Geräten von Apple. Da diese Geräte einen wichtigen Markt ausmachen, lernen Sie, wie Sie Video ohne Flash verwenden können: In Kapitel 7, *Audio und Video einbetten*, erfahren Sie, wie Sie Audio und Video in HTML5 mit entsprechenden Ausweichlösungen einsetzen können.

Bessere Anwendungen

Entwickler haben alles Mögliche versucht, um umfangreichere, interaktivere Webanwendungen zu erstellen – von ActiveX-Elementen bis hin zu Flash. HTML5 bietet erstaunliche Funktionen, die in einigen Fällen Technologien von Drittanbietern vollkommen überflüssig machen.

Kommunikation mit anderen Dokumenten

Webbrowser verhindern, dass wir Skripten einer Domain dazu nutzen, Skripten auf einer anderen Domain zu verändern oder damit zu kommunizieren. Durch diese Einschränkung werden Endbenutzer vor Cross-Site Scripting geschützt, mit dem nichts ahnenden Besuchern von Webseiten schon allerhand fiese Dinge angetan wurden.

Allerdings führt das dazu, dass *überhaupt keine* Skripten funktionieren, selbst dann, wenn wir sie selbst schreiben und wissen, dass wir dem Inhalt vertrauen können. HTML5 bietet einen Workaround, der sowohl sicher als auch einfach zu implementieren ist. Wie Sie das zum Laufen bekommen, erfahren Sie im Abschnitt *Über Domains hinweg kommunizieren* auf Seite 206.

Web Sockets

HTML5 bietet Unterstützung für Web Sockets, mit denen Sie eine dauerhafte Verbindung zu einem Server herstellen können. Anstatt ständig vom Backend aus Fortschrittsaktualisierungen zu pollen, kann Ihre Webseite einen Socket abonnieren, und das Backend kann dann Benachrichtigungen an Ihre Benutzer pushen. Damit werden wir im Abschnitt *Mit Web Sockets chatten* auf Seite 213 ein bisschen spielen.

Clientseitige Speicherung

Wir tendieren dazu, HTML5 als Webtechnologie zu verstehen. Aber in Verbindung mit der Web Storage- und der WebSQL Database-API können wir Browseranwendungen entwickeln, bei denen die Daten dauerhaft und vollständig auf dem Clientrechner verbleiben. Wie Sie diese APIs verwenden, erfahren Sie in Kapitel 9, *Mit clientseitigen Daten arbeiten*, ab Seite 177.

Bessere Benutzeroberflächen

Die Benutzeroberfläche ist ein sehr wichtiger Teil jeder Webanwendung. Und tagtäglich springen wir durch brennende Reifen, um die Browser das zu tun zu lassen, was wir möchten. Für die Gestaltung einer Tabelle oder runder Ecken verwenden wir entweder JavaScript-Bibliotheken oder tonnenweise zusätzliches Markup. Mit HTML5 und CSS3 gehören solche Praktiken der Vergangenheit an.

Bessere Formulare

HTML5 bietet auch verbesserte Steuerelemente für Benutzeroberflächen. Lange Zeit waren wir gezwungen, JavaScript und CSS zu verwenden, um Slider, Kalender-Datepicker und Farbwähler zu bauen. All das gibt es in HTML5 als eigenständige Elemente, ebenso wie Dropdown-Menüs, Kontrollkästchen und Optionsfelder. In Kapitel 3, *Benutzerfreundliche Webformulare*, lernen Sie, wie Sie diese Elemente verwenden können. Auch wenn das noch nicht in jedem Browser so ganz funktioniert, sollten Sie es im Auge behalten – insbesondere dann, wenn Sie webbasierte Anwendungen entwickeln. Neben der erhöhten Benutzerfreundlichkeit auch ohne JavaScript-Bibliotheken gibt es noch einen weiteren Vorteil: mehr Barrierefreiheit. Screenreader und Browser können diese Steuerelemente so implementieren, dass auch Behinderte sie einfach anwenden können.

Verbesserte Barrierefreiheit

Die Beschreibung unserer Inhalte mit den neuen HTML5-Elementen erleichtert es Programmen wie Screenreadern, unsere Inhalte zu verarbeiten. So ist die Navigation einer Website beispielsweise viel leichter zu finden, wenn Sie nur nach dem `nav`-Tag zu suchen brauchen statt nach einem bestimmten `div` oder einer ungeordneten Liste. Fußzeilen, Seitenleisten und andere Inhalte können einfach neu angeordnet oder ausgelassen werden. Das Einlesen von Seiten wird allgemein unproblematischer, und damit werden die Erfahrungen für Menschen angenehmer, die auf unterstützende Technologien angewiesen sind. Außerdem kann über die neuen Elementattribute die Funktion von Elementen angegeben werden, sodass Bildschirmlesegeräte diese leichter verarbeiten können. In Kapitel 5, *Mehr Barrierefreiheit*, lernen Sie, wie Sie diese neuen Attribute einsetzen, damit sie von aktuellen Bildschirmlesegeräten verwendet werden können.

Fortschrittliche Selektoren

CSS3 bietet Selektoren, mit denen Sie Tabellenzeilen abwechselnd auswählen, alle aktivierten Kontrollkästchen oder sogar den letzten Absatz in einer Gruppe auswählen können. Sie erreichen mehr mit weniger Code und weniger Markup. Dadurch wird es auch viel leichter, HTML zu gestalten, das Sie nicht verändern können. In Kapitel 4, *Bessere Benutzeroberflächen mit CSS3*, lernen Sie, diese Selektoren effizient einzusetzen.

Visuelle Effekte

Schlagschatten für Texte und Bilder bringen Tiefe in eine Webseite. Verläufe sorgen für eine zusätzliche Dimension. Mit CSS3 können Sie Elementen Schatten und Verläufe hinzufügen, ohne auf Hintergrundbilder oder zusätzliches Markup zurückzugreifen. Sie können Transformationen anwenden, um Ecken abzurunden oder Elemente zu verzerren oder zu drehen. Wie das alles funktioniert, erfahren Sie in Kapitel 8, *Augenschmaus*, ab Seite 149.

1.2 Abwärtskompatibilität

Einer der entscheidenden Gründe für Sie, HTML5 bereits heute zu verwenden, ist, dass es in den meisten aktuellen Browsern funktioniert. Derzeit können Sie sogar in Internet Explorer 6 mit der Verwendung von HTML5 beginnen und Ihr Markup langsam migrieren. Es wird sogar beim Validierungsservice des W3C korrekt validiert (natürlich nur bedingt, da die Standards immer noch weiterentwickelt werden).

Wenn Sie schon mit HTML oder XML gearbeitet haben, haben Sie die Doctype-Deklaration bereits kennengelernt. Sie dient dazu, Validierern und Editoren mitzuteilen, welche Tags und Attribute verwendet werden können und wie das Dokument geformt sein soll. Und eine Menge Webbrowser stellen darüber fest, wie die Seite dargestellt werden soll. Ein gültiger Doctype lässt Browser oft Seiten im „Standards Mode“ darstellen.

Im Vergleich zu dem eher ausführlichen Doctype *XHTML 1.0 Transitional*, der für viele Websites verwendet wird

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
    "http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
```

ist der HTML5-Doctype lächerlich einfach:

```
html5_why/index.html
```

```
<!DOCTYPE html>
```

Schreiben Sie das in den oberen Teil Ihres Dokuments, und schon verwenden Sie HTML5.

Natürlich können Sie keines der neuen HTML5-Elemente verwenden, wenn Ihr Zielbrowser diese nicht unterstützt. Aber Ihr Dokument wird bereits als HTML5 validiert.



Joe fragt ...

**Ich mag aber die selbstschließenden Tags in XHTML.
Kann ich sie weiterhin verwenden?**

Natürlich können Sie! Viele Entwickler haben sich wegen der strikteren Anforderungen an das Markup in XHTML verliebt. XHTML-Dokumente verlangen, dass Attribute in Anführungszeichen eingfasst werden, dass Sie Inhaltstags selbst schließen, für Attributnamen Kleinbuchstaben verwenden, und haben so wohlgeformtes Markup in das World Wide Web eingeführt. Der Umstieg auf HTML5 bedeutet nicht notwendigerweise, dass Sie Ihre Gewohnheiten ändern müssen. HTML5-Dokumente sind valide, egal ob Sie lieber die HTML5- oder die XHTML-Syntax verwenden. Sie müssen jedoch die Auswirkungen selbstschließender Tags verstehen.

Die meisten Webserver liefern HTML-Seiten mit dem MIME-Type `text/html` aus, weil der Internet Explorer unfähig ist, den XHTML-Seiten zugeordneten MIME-Type `application/xml+xhtml` zu verarbeiten. Deshalb neigen Browser dazu, selbstschließende Tags zu entfernen, weil solche Tags vor HTML5 nicht als valides HTML angesehen wurden. Wenn Sie beispielsweise ein selbstschließendes `script`-Tag oberhalb eines `h2`-Tags verwenden

```
<script language="javascript" src="application.js" />
<h2>Hello</h2>
```

dann würde der Browser den schließenden `/` entfernen, und der Renderer würde anschließend davon ausgehen, dass sich `h2` innerhalb des `script`-Tags befindet, das nie geschlossen wird. Das ist der Grund, warum Sie häufig `script`-Tags explizit mit einem schließenden Tag finden, obwohl ein selbstschließendes Tag valides XHTML-Markup ist.

Seien Sie sich also solcher möglichen Probleme bewusst, wenn Sie selbstschließende Tags in Ihren HTML5-Dokumenten verwenden. Denn diese Dokumente werden mit dem MIME-Type `text/html` ausgeliefert. Mehr zu diesem und anderen Problemen können Sie unter <http://www.webdevout.net/articles/beware-of-xhtml#myths> erfahren.

1.3 Der Weg in die Zukunft ist steinig

Es gibt einige Hürden, die der weiteren Verbreitung von HTML5 und CSS3 im Wege stehen. Manche sind offensichtlich, andere nicht.

Internet Explorer

Der Internet Explorer verfügt derzeit über die größte Anwenderbasis, und die Versionen 8 und darunter bieten eine äußerst schwache Unterstützung für HTML5 und CSS3. Beim IE 9 sieht die Situation schon besser aus, aber er ist noch nicht weit verbreitet. Das bedeutet nicht, dass wir HTML5 und CSS3 nicht trotzdem für unsere Websites verwenden können. Wir bekommen es hin, dass unsere Websites im Internet Explorer funktionieren, sie müssen ja nicht genauso funktionieren wie die Versionen für Chrome und Firefox. Wir müssen lediglich Ausweichlösungen anbieten, sodass wir keine Benutzer verärgern oder Kunden verlieren.

Barrierefreiheit

Die Benutzer müssen mit unseren Websites interagieren können, ganz gleich ob sie seh- oder hörbehindert sind, ältere Browser, langsame Verbindungen oder mobile Geräte verwenden. In HTML5 werden einige neue Elemente eingeführt, zum Beispiel audio, video und canvas. Mit Audio und Video gab es immer schon Schwierigkeiten in puncto Barrierefreiheit, aber das canvas-Element stellt uns vor neue Herausforderungen. Mit diesem Element können wir Vektorbilder über JavaScript innerhalb des HTML-Dokuments erstellen. Hieraus ergeben sich Schwierigkeiten für Sehbehinderte, aber auch für die fünf Prozent der Webbenutzer, die JavaScript deaktiviert haben.³

Wir dürfen nicht mit den neuen Technologien davongallopieren und dabei die Barrierefreiheit aus den Augen verlieren. Wir müssen geeignete Ausweichlösungen für die neuen HTML5-Elemente bieten – ebenso wie für Menschen, die den Internet Explorer verwenden.

3 <http://visualrevenue.com/blog/2007/08/eu-and-us-javascript-disabled-index.html>

Kuchen und Zuckerguss

Ich mag Kuchen. Ich mag Torte lieber, aber Kuchen ist auch ziemlich gut. Am allerliebsten mag ich Kuchen mit Zuckerguss.

Wenn Sie Webanwendungen entwickeln, dürfen Sie nicht vergessen, dass die hübsche Benutzeroberfläche und die schicken JavaScripts lediglich der Zuckerguss auf dem Kuchen sind. Ihre Website kann auch ohne dieses ganze Zeug richtig gut sein. Und genau wie bei einem Kuchen brauchen Sie ein Fundament, auf das Sie den Zuckerguss auftragen können.

Ich habe einige Leute kennengelernt, die keinen Zuckerguss mögen. Sie kratzen ihn vom Kuchen ab. Ich habe auch Leute kennengelernt, die Webapplikationen aus verschiedenen Gründen ohne JavaScript verwenden.

Backen Sie für diese Menschen zuerst einen wirklich tollen Kuchen. Und tragen Sie erst dann den Zuckerguss auf.

Veraltete Tags

In HTML5 wurden eine Menge neuer Elemente eingeführt, aber die Spezifikation schafft auch einige gebräuchliche Elemente ab, die Sie eventuell in Ihren Webseiten verwenden.⁴ Diese sollten Sie beim Umstieg entfernen:

- `basefont`
- `big`
- `center`
- `font`
- `s`
- `strike`
- `tt`
- `u`

Einige dieser Tags sind ziemlich veraltet. Aber da draußen finden Sie viele Seiten, die mit visuellen Editoren wie zum Beispiel Dreamweaver bearbeitet werden und deshalb eine Menge `font`- und `center`-Tags enthalten.

⁴ <http://www.w3.org/TR/html5-diff/>

Neben gestalterischen Elementen wurde auch die Unterstützung von Frames entfernt. Frames waren immer sehr beliebt in Enterprise-Webanwendungen wie zum Beispiel PeopleSoft, Microsoft Outlook Web Access und sogar in speziell entwickelten Portalen. Trotz der weit verbreiteten Verwendung haben Frames so viele Schwierigkeiten im Hinblick auf Benutzerfreundlichkeit und Barrierefreiheit mit sich gebracht, dass man sie loswerden musste. Dementsprechend sind die folgenden Elemente verschwunden:

- `frame`
- `frameset`
- `noframes`

Sie sollten Möglichkeiten finden, das Layout Ihrer Benutzeroberflächen mit regulärem CSS und ein bisschen JavaScript auch ohne Frames zu gestalten. Falls Sie mit Frames dafür sorgen, dass Kopfzeile, Fußzeile und Navigation auf jeder Seite Ihrer Anwendung gleich angezeigt werden, können Sie das auch mit den Tools Ihres Webentwicklungs-Frameworks erreichen. Einige andere Elemente sind verschwunden, weil es bessere Optionen dafür gibt:

- `acronym` wird durch `abbr` ersetzt.
- `applet` wird durch `object` ersetzt.
- `dir` wird durch `ul` ersetzt.

Zusätzlich zu den veralteten Elementen sind auch viele Attribute nicht mehr zulässig. Dazu gehören Darstellungsattribute wie zum Beispiel:

- `align`
- `link`, `vlink`, `alink` und `text` für das `body`-Tag
- `bgcolor`
- `height` und `width`
- `scrolling` für das `iframe`-Element
- `valign`
- `hspace` und `vspace`
- `cellpadding`, `cellspacing` und `border` für `table`

Falls Sie für Ihre Links `target` verwenden, wie z.B. in:

```
<a href="http://www.google.com" target="_blank">
```

sollten Sie stattdessen JavaScript verwenden, da auch `target` abgeschafft wurde.

Das Attribut `profile` des `head`-Tags wird auch nicht mehr unterstützt. Das ist etwas, was Sie tendenziell in einer Menge WordPress-Vorlagen finden.

Zu guter Letzt ist das Attribut `longdesc` für `img`- und `iframe`-Elemente verschwunden. Für Verfechter der Barrierefreiheit ist das ein bisschen enttäuschend, da `longdesc` die allgemein anerkannte Möglichkeit war, Benutzern mit Bildschirmlesegeräten zusätzliche beschreibende Informationen zu bieten.

Falls Sie vorhaben, HTML5 für Ihre vorhandenen Websites zu verwenden, sollten Sie diese Elemente ausfindig machen und entfernen oder durch semantischere Varianten ersetzen. Validieren Sie Ihre Seiten unbedingt mit dem Validierungsservice des W3C⁵, der Ihnen dabei helfen wird, veraltete Tags und Attribute aufzuspüren.

Konkurrierende Unternehmensinteressen

Der Internet Explorer ist nicht der einzige Browser, dessen Entwickler die Einführung von HTML5 und CSS3 hinauszögern. Google, Apple und die Mozilla Foundation haben ebenfalls ihre eigenen Vorstellungen und kämpfen um die Vorherrschaft. Sie streiten über die Unterstützung von Video- und Audio-Codecs und bringen ihre Standpunkte in ihren Browser-Versionen klar zum Ausdruck. So spielt Safari beispielsweise MP3 über das `audio`-Element ab, aber keine ogg-Dateien. Firefox unterstützt dagegen ogg-Dateien, aber keine mp3-Dateien.

Irgendwann werden diese Unterschiede entfallen. Bis dahin können wir lediglich intelligente Entscheidungen darüber treffen, was wir selbst unterstützen – indem wir unsere Implementierungen auf die von unseren Zielgruppen verwendeten Browser beschränken oder alles mehrfach implementieren, für jeden Browser einmal – bis die Standards endgültig feststehen. Es ist aber alles nicht so schlimm, wie es sich anhört. Mehr darüber erfahren Sie in Kapitel 7, *Audio und Video einbetten*, ab Seite 131.

5 <http://validator.w3.org/>

HTML5 und CSS3 befinden sich noch in der Entwicklung

Die Spezifikationen sind noch nicht endgültig. Das bedeutet, dass sich alles innerhalb dieser Spezifikationen ändern kann. Firefox, Chrome und Safari bieten eine gute Unterstützung für HTML5. Aber wenn sich die Spezifikation ändert, werden sich die Browser mit ihr verändern, und dies kann wiederum zu veralteten und defekten Websites führen. Während dieses Buch geschrieben wurde, wurde die CSS3-Eigenschaft `box-shadow` aus der Spezifikation entfernt und wieder hinzugefügt. Das Protokoll Web Sockets wurde verändert, und die Client-Server-Kommunikation wurde dadurch zunichte gemacht.

Wenn Sie den Fortschritt von HTML5 und CSS3 mitverfolgen und immer auf dem Laufenden bleiben, ist alles gut. Eine ganze Menge der Dinge, die wir in diesem Buch diskutieren, wird lange funktionieren.

Wenn Sie über etwas stolpern, das in einem Ihrer Zielbrowser nicht funktioniert, verwenden Sie einfach JavaScript und Flash als Spachtelmasse, um diese Lücken zu füllen. Sie entwickeln Lösungen, die für alle Benutzer funktionieren. Und im Laufe der Zeit können Sie JavaScript und die anderen Ausweichlösungen nach und nach entfernen, ohne Ihre eigentlichen Implementierungen zu ändern.

Aber bevor Sie zu viel über die Zukunft nachdenken, fangen wir einfach an, mit HTML5 zu arbeiten. Eine ganze Menge neuer struktureller Tags warten darauf, Sie im nächsten Kapitel kennenzulernen. Und wir wollen sie doch nicht so lange warten lassen, oder?

Teil I

Bessere Benutzeroberflächen

Kapitel 2

Neue strukturelle Tags und Attribute

In den ersten Kapiteln dieses Buchs sprechen wir darüber, wie wir mit den Funktionen von HTML5 und CSS3 unseren Benutzern verbesserte Interfaces bieten können. Sie erfahren, wie Sie bessere Formulare erstellen, Tabellen einfacher gestalten und die Barrierefreiheit Ihrer Seiten für unterstützende Geräte verbessern. Außerdem erfahren Sie, wie Sie bei der Erzeugung von Inhalten die Usability von Stylesheets für den Druck optimieren, und Sie erforschen das „In-place-Editing“ mit dem neuen Attribut `contenteditable`. Zunächst sehen wir uns aber an, wie wir mit den neuen Elementen von HTML5 unsere Seiten besser strukturieren können.

Zu Beginn würde ich gern mit Ihnen über ein ernst zu nehmendes Problem sprechen, von dem heutzutage viele Webentwickler betroffen sind: Die *Divitis* greift um sich – ein chronisches Syndrom, das dazu führt, dass Webentwickler Elemente in zusätzliche `div`-Tags mit IDs wie zum Beispiel `banner`, `sidebar`, `article` und `footer` verpacken. Dieses Syndrom ist in hohem Maße ansteckend. Divitis verbreitet sich extrem schnell von Entwickler zu Entwickler. Und da `divs` für das bloße Auge unsichtbar sind, bleiben insbesondere leichte Fälle von Divitis unter Umständen jahrelang unbemerkt.

Ein häufiges Symptom von Divitis sieht so aus:

```
<div id="navbar_wrapper">
  <div id="navbar">
    <ul>
      <li><a href="/">Home</a></li>
      <li><a href="/">Home</a></li>
    </ul>
  </div>
</div>
```

Hier wurde eine ungeordnete Liste, die ihrerseits ja bereits ein Blockelement ist¹, in zwei `div`-Tags eingebettet, die auch wiederum Blockelemente sind. Die `id`-Attribute dieser Wrapper-Elemente erklären zwar, welche Aufgabe sie haben. Aber Sie können zumindest einen dieser Wrapper entfernen und zum gleichen Ergebnis kommen. Diese übermäßige Verwendung von Markup führt zu aufgeblähten Seiten, die schwierig zu gestalten und zu pflegen sind.

Es gibt jedoch Hoffnung: Die HTML5-Spezifikation liefert ein Heilmittel in Form von neuen semantischen Tags, die ihren Inhalt beschreiben. Weil so viele Entwickler Seitenleisten, Kopfzeilen, Fußzeilen und Abschnitte in ihren Designs verwenden, werden mit der HTML5-Spezifikation spezielle Tags eingeführt, um Seiten in solche logischen Abschnitte zu unterteilen. Machen wir uns mit diesen neuen Elementen an die Arbeit. Mit HTML5 können wir der Divitis noch in unserer Generation ein Ende setzen.

Neben den neuen strukturellen Tags werden wir auch das `meter`-Element besprechen und Ihnen zeigen, wie Sie in HTML5 mit den neuen benutzerdefinierten Attributen Daten in Elemente einbetten können, ohne dafür Klassen oder existierende Attribute zu missbrauchen. Mit anderen Worten werden wir herausfinden, wie wir das richtige Tag für die richtige Aufgabe einsetzen.

In diesem Kapitel erforschen wir die folgenden neuen Elemente und Funktionen:²

1 Sie erinnern sich, dass Blockelemente eine eigene Zeile einnehmen, während Inline-Elemente keinen Zeilenumbruch erzwingen.

2 In den folgenden Beschreibungen wird die Unterstützung durch die verschiedenen Browser in eckigen Klammern mit einem Kurzcode und der mindestens erforderlichen Versionsnummer angegeben. Die verwendeten Codes lauten: *C*: Google Chrome, *F*: Firefox, *IE*: Internet Explorer, *O*: Opera, *S*: Safari, *IOS*: iOS-Geräte mit Mobile Safari und *A*: Android-Browser.

<header>

Definiert den Kopfbereich einer Seite oder eines Abschnitts.

[C5, F3.6, IE8, S4, O10]

<footer>

Definiert die Fußzeile einer Seite oder eines Abschnitts.

[C5, F3.6, IE8, S4, O10]

<nav>

Definiert einen Navigationsbereich einer Seite oder eines Abschnitts.

[C5, F3.6, IE8, S4, O10]

<section>

Definiert einen logischen Abschnitt einer Seite oder einer Gruppe von Inhalten. [C5, F3.6, IE8, S4, O10]

<article>

Definiert einen Artikel oder ein in sich abgeschlossenes Inhaltselement.

[C5, F3.6, IE8, S4, O10]

<aside>

Definiert sekundäre oder ähnliche Inhalte. [C5, F3.6, IE8, S4, O10]

Benutzerdefinierte Datenattribute

Mit dem data-Muster lassen sich zu beliebigen Elementen benutzerdefinierte Attribute hinzufügen. [Alle Browser unterstützen das Einlesen dieser Attribute über die JavaScript-Methode `getAttribute()`.]

<meter>

Beschreibt einen Wert innerhalb eines Wertebereichs. [C5, F3.5, S4, O10]

<progress>

Steuerelement, das den Fortschritt zu einem bestimmten Ziel hin in Echtzeit anzeigt. [Zum Zeitpunkt der Veröffentlichung nicht unterstützt.]

1

Einen Blog mit semantischem Markup neu definieren

In einem Blog finden Sie jede Menge Inhalte, die nach strukturiertem Markup schreiben. Sie brauchen Kopfzeilen, Fußzeilen, mehrere Navigationsmöglichkeiten (Archive, Blogrolls und interne Links) und natürlich Artikel oder Beiträge. Basteln wir mit HTML5 ein Mock-up für die Startseite des Blogs von AwesomeCo, einem der großartigsten Unternehmen überhaupt.

In Abbildung 2.1 auf der nächsten Seite sehen Sie, worum es geht. Es handelt sich um eine typische Blog-Struktur, mit einer Hauptkopfzeile und einer horizontalen Navigation darunter. Im Hauptbereich erhält jeder Artikel eine Kopf- und eine Fußzeile. Artikel können außerdem einen Zitatkasten oder Sekundärinhalte enthalten. Zum Abschluss erhält die Seite eine Fußzeile für Kontakt- und Copyright-Informationen. An der Struktur an sich ist nichts außergewöhnlich. Allerdings bauen wir sie diesmal nicht aus einer Menge `div`-Tags auf, sondern verwenden spezielle Tags, um die jeweiligen Abschnitte zu beschreiben.

Wenn wir fertig sind, erhalten wir etwas, das in etwa wie Abbildung 2.2 auf Seite 28 aussieht.

Auf den Doctype kommt es an

Wir möchten die neuen Elemente von HTML5 verwenden. Deshalb müssen wir Browsern und Validierern mitteilen, welche Tags wir verwenden. Erstellen Sie eine neue Seite mit dem Namen `index.html`, und schreiben Sie die folgende einfache HTML5-Vorlage in die Datei.

`html5newtags/index.html`

```

1 <!DOCTYPE html>
2 <html lang="en-US">
3   <head>
4     <meta http-equiv="Content-Type" content="text/html; charset=utf-8" />
5     <title>AwesomeCo Blog</title>
6   </head>
7
8   <body>
9     </body>
10 </html>

```

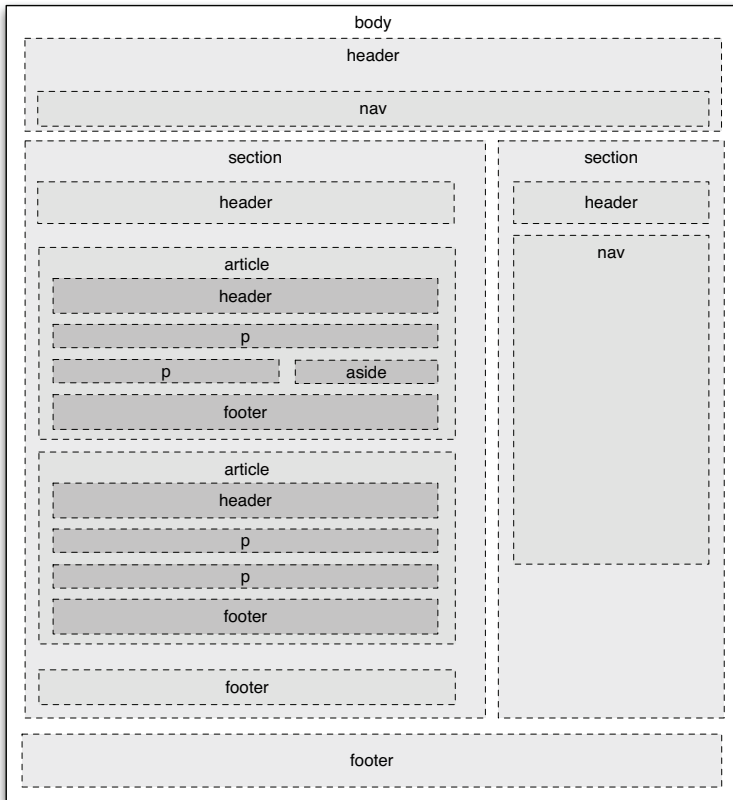


Abbildung 2.1: Die Blogstruktur bei Verwendung von semantischem HTML5-Markup

Sehen Sie sich mal den Doctype in Zeile 1 an. Mehr brauchen wir für den HTML5-Doctype nicht zu schreiben. Wenn Sie regelmäßig Webseiten erstellen, sind Sie wahrscheinlich eher mit diesen langen, schwer zu merkenden Doctypes für XHTML vertraut:

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
```

Zum Vergleich noch einmal der HTML5-Doctype:

```
<!DOCTYPE HTML>
```

Das ist doch viel einfacher und viel leichter zu merken!

Ein Doctype hat zweierlei Aufgaben. Erstens hilft er Validierern festzustellen, welche Validierungsregeln für die Validierung des Codes herangezogen werden müssen. Außerdem zwingt der Doctype den Internet Explorer 6, 7 und 8, in den „Standards Mode“ zu wechseln, was von

entscheidender Bedeutung ist, wenn Sie Webseiten erstellen, die in allen Browsern funktionieren sollen. Der HTML5-Doctype erfüllt beide Anforderungen und wird *sogar vom Internet Explorer 6* erkannt.



Abbildung 2.2: Das fertige Layout

Kopfzeilen

Kopfzeilen (nicht zu verwechseln mit Überschriften wie etwa h1, h2 und h3) können alle möglichen Arten von Inhalten enthalten – vom Unternehmenslogo bis hin zum Suchfeld. Die Kopfzeile unseres Blogs soll zunächst nur den Titel des Blogs enthalten.

html5newtags/index.html

```
1 <header id="page_header">
2   <h1>AwesomeCo Blog!</h1>
3 </header>
```

Sie sind nicht auf eine Kopfzeile pro Seite beschränkt. Jeder einzelne Abschnitt oder Artikel kann eine eigene Kopfzeile enthalten. Da kann es hilfreich sein, Ihre Elemente mit dem id-Attribut eindeutig zu kennzeichnen, wie ich das in Zeile 1 getan habe. Mit einer eindeutigen ID sind Elemente mit CSS einfach zu gestalten oder mit JavaScript ausfindig zu machen.

Semantisches Markup

Semantisches Markup dient dazu, Ihren Inhalt zu beschreiben. Wenn Sie bereits seit mehreren Jahren Webseiten entwickeln, teilen Sie Ihre Seiten wahrscheinlich in verschiedene Abschnitte wie header, footer und sidebar auf. Dadurch können Sie die einzelnen Abschnitte der Seite leichter identifizieren, wenn Sie Stylesheets und andere Formatierungen darauf anwenden.

Semantisches Markup macht es Maschinen und Menschen gleichermaßen leicht, die Bedeutung und den Kontext des Inhalts zu verstehen. Die neuen Markup-Tags von HTML5 wie header und nav sollen Ihnen genau dabei helfen.

Fußzeilen

Das footer-Element definiert Fußzeileninformationen für ein Dokument oder einen angrenzenden Abschnitt. Fußzeilen auf Websites kennen Sie bereits. Üblicherweise stehen darin Informationen wie das Copyright-Datum oder darüber, wem die Website gehört. Der Spezifikation entsprechend können wir auch mehrere Fußzeilen in einem Dokument haben. Das bedeutet also, dass wir auch innerhalb unserer Blog-Artikel Fußzeilen verwenden können.

Für den Moment definieren wir einfach eine einfache Fußzeile für unsere Seite. Da wir mehr als eine Fußzeile verwenden können, geben wir dieser genauso wie der Kopfzeile eine ID. Dadurch können wir diese bestimmte Fußzeile eindeutig identifizieren, wenn wir das Element und seine Kinder stylen möchten.

```
html5newtags/index.html
```

```
<footer id="page_footer">
  <p>&copy; 2010 AwesomeCo.</p>
</footer>
```

Die Fußzeile enthält lediglich ein Copyright-Datum. Jedoch können Fußzeilen genau wie Kopfzeilen auch andere Elemente enthalten, wie etwa Navigationselemente.

Navigation

Die Navigation ist für den Erfolg einer Website entscheidend. Ihre Besucher werden nicht lange bleiben, wenn sie nicht das finden, was sie suchen. Daher ist es absolut sinnvoll, dass die Navigation ein eigenes HTML-Tag erhält.

Fügen wir einen Navigationsabschnitt in die Kopfzeile unseres Dokuments ein. Wir erstellen Links auf die Homepage des Blogs, das Archiv, eine Liste mit den Verfassern von Beiträgen zum Blog und einen Link auf die Kontaktseite.

html5newtags/index.html

```

1 <header id="page_header">
2   <h1>AwesomeCo Blog!</h1>
3   <nav>
4     <ul>
5       <li><a href="/">Latest Posts</a></li>
6       <li><a href="archives">Archives</a></li>
7       <li><a href="contributors">Contributors</a></li>
8       <li><a href="contact">Contact Us</a></li>
9     </ul>
10  </nav>
11 </header>

```

Ihre Seite kann nicht nur mehrere Kopf- und Fußzeilen, sondern auch mehrere Navigationselemente enthalten. Die Navigation befindet sich häufig in der Kopfzeile und in der Fußzeile, die Sie nun explizit kennzeichnen können. Die Fußzeile unseres Blogs braucht Links auf die Homepage von AwesomeCo, eine Seite über das Unternehmen sowie Links auf die Nutzungsbedingungen und Datenschutzrichtlinien. Diese Links fügen wir in der Seite als eine weitere ungeordnete Liste im Element footer ein.

html5newtags/index.html

```

<footer id="page_footer">
  <p>&copy; 2010 AwesomeCo.</p>
  <nav>
    <ul>
      <li><a href="http://awesomeco.com/">Home</a></li>
      <li><a href="about">About</a></li>
      <li><a href="terms.html">Terms of Service</a></li>
      <li><a href="privacy.html">Privacy</a></li>
    </ul>
  </nav>
</footer>

```

Das Aussehen der beiden Navigationsleisten passen wir später mit CSS an. Achten Sie daher zunächst nicht so sehr auf die Optik. Die Aufgabe dieser neuen Elemente ist es, den Inhalt zu beschreiben, nicht aber, das Aussehen des Inhalt zu bestimmen.

Abschnitte und Artikel

Abschnitte sind die logischen Bereiche einer Seite. Deshalb gibt es nun das Element `section`, um das oft missbrauchte `div`-Tag bei der Beschreibung logischer Abschnitte einer Seite abzulösen.

```
html5newtags/index.html
```

```
<section id="posts">
</section>
```

Übertreiben Sie es aber nicht mit den Abschnitten. Setzen Sie sie ein, um Ihren Inhalt logisch zu gliedern. Hier haben wir einen Abschnitt erstellt, der alle Beiträge in einem Blog enthält. Jedoch soll nicht jeder Beitrag einen eigenen Abschnitt erhalten. Dafür gibt es ein besser geeignetes Tag.

Artikel

Das `article`-Tag ist das perfekte Element, um den Inhalt einer Webseite zu beschreiben. Mit den vielen Elementen auf einer Seite – Kopfzeilen, Fußzeilen, Navigationselemente, Werbung, Widgets, Blogrolls und Lesezeichen für soziale Medien – könnten Sie glatt vergessen, dass die Benutzer Ihre Website wegen des von Ihnen angebotenen Inhalts besuchen. Das `article`-Tag hilft Ihnen dabei, diesen Inhalt zu beschreiben.

Jeder unserer Artikel besteht aus einer Kopfzeile, dem eigentlichen Inhalt und einer Fußzeile. So definieren wir einen vollständigen Artikel:

```
html5newtags/index.html
```

```
<article class="post">
  <header>
    <h2>How Many Should We Put You Down For?</h2>
    <p>Posted by Brian on
      <time datetime="2010-10-01T14:39">October 1st, 2010 at 2:39PM</time>
    </p>
  </header>
  <p>
    The first big rule in sales is that if the person leaves empty-handed,
    they're likely not going to come back. That's why you have to be
    somewhat aggressive when you're working with a customer, but you have
    to make sure you don't overdo it and scare them away.
  </p>
```

```

<p>
  One way you can keep a conversation going is to avoid asking questions
  that have yes or no answers. For example, if you're selling a service
  plan, don't ever ask "Are you interested in our 3 or 5 year
  service plan?" Instead, ask "Are you interested in the 3
  year service plan or the 5 year plan, which is a better value?"
  At first glance, they appear to be asking the same thing, and while
  a customer can still opt out, it's harder for them to opt out of
  the second question because they have to say more than just "no."
</p>
<footer>
  <p><a href="comments"><i>25 Comments</i></a>...</p>
</footer>
</article>

```



Joe fragt ...

Was ist der Unterschied zwischen Artikeln und Abschnitten?

Stellen Sie sich einen Abschnitt als logischen Teil eines Dokuments vor. Einen Artikel können Sie sich als den eigentlichen Inhalt vorstellen, wie etwa einen Artikel in einer Zeitschrift, einen Beitrag in einem Blog oder eine aktuelle Meldung.

Die neuen Tags beschreiben genau den Inhalt, den sie enthalten. Abschnitte können mehrere Artikel enthalten, und Artikel können aus mehreren Abschnitten bestehen. Ein Abschnitt ist wie der Sportteil einer Zeitung. Der Sportteil enthält viele Artikel. Jeder dieser Artikel kann wiederum aus mehreren eigenständigen Abschnitten bestehen. Bestimmte Abschnitte wie Kopfzeilen und Fußzeilen erhalten eigene Tags. Ein Abschnitt ist ein allgemein gehaltenes Element, mit dem Sie andere Elemente logisch gruppieren können.

Bei semantischem Markup geht es darum, die Bedeutung Ihres Inhalts zu vermitteln.

In einem Artikel können wir die Elemente `header` und `footer` verwenden, wodurch es wesentlich einfacher wird, diese bestimmten Abschnitte zu beschreiben. Wir können unseren Artikel auch mit dem Element `section` in mehrere Abschnitte unterteilen.

Das `aside`-Tag und Seitenleisten

Manche Inhalte ergänzen etwas zum eigentlichen Inhalt, wie etwa Zitatkästen, Diagramme, weiterführende Gedanken oder entsprechende Links. Mit dem neuen `aside`-Tag können Sie diese Elemente kenntlich machen.

```
html5newtags/index.html
```

```
<aside>
  <p>
    &quot;Never give someone a chance to say no when selling your
    product.&quot;
  </p>
</aside>
```

Das Zitat schreiben wir in ein `aside`-Element. Das `aside`-Element verschachteln wir wiederum in den Artikel und platzieren es damit nahe beim zugehörigen Inhalt.

So sieht unser vollständiger Abschnitt mit dem `aside`-Element aus:

```
html5newtags/index.html
```

```
<section id="posts">
  <article class="post">
    <header>
      <h2>How Many Should We Put You Down For?</h2>
      <p>Posted by Brian on
        <time datetime="2010-10-01T14:39">October 1st, 2010 at 2:39PM</time>
      </p>
    </header>

    <aside>
      <p>
        &quot;Never give someone a chance to say no when selling your
        product.&quot;
      </p>
    </aside>

    <p>
      The first big rule in sales is that if the person leaves empty-handed,
      they're likely not going to come back. That's why you have to be
      somewhat aggressive when you're working with a customer, but you have
      to make sure you don't overdo it and scare them away.
    </p>
    <p>
      One way you can keep a conversation going is to avoid asking questions
      that have yes or no answers. For example, if you're selling a service
      plan, don't ever ask &quot;Are you interested in our 3 or 5 year service
      plan?&quot; Instead, ask &quot;Are you interested in the 3
      year service plan or the 5 year plan, which is a better value?&quot;
      At first glance, they appear to be asking the same thing, and while
      a customer can still opt out, it's harder for them to opt out of
      the second question because they have to say more than just
      &quot;no.&quot;
    </p>
    <footer>
      <p><a href="comments"><i>25 Comments</i></a>...</p>
    </footer>
  </article>
</section>
```

Nun müssen wir nur noch den Abschnitt für die Seitenleiste einfügen.

aside-Elemente sind keine Seitenleisten

Unser Blog enthält eine rechte Seitenleiste mit den Archiven für den Blog. Falls Sie jetzt glauben, wir könnten das `aside`-Tag verwenden, um die Seitenleiste unseres Blogs zu definieren, liegen Sie leider falsch. Sie *könnten* das zwar so machen, aber es widerspricht dem Sinn der Spezifikation. `aside` wurde entwickelt, um die Inhalte anzuzeigen, die zu einem Artikel gehören: die richtige Stelle also, um entsprechende Links, ein Glossar oder einen Zitatkasten anzuzeigen.

Als Markup für unsere Seitenleiste mit der Liste der bisherigen Archive verwenden wir ein weiteres `section`-Tag sowie ein `nav`-Tag:

```
html5newtags/index.html
```

```
<section id="sidebar">
  <nav>
    <h3>Archives</h3>
    <ul>
      <li><a href="2010/10">October 2010</a></li>
      <li><a href="2010/09">September 2010</a></li>
      <li><a href="2010/08">August 2010</a></li>
      <li><a href="2010/07">July 2010</a></li>
      <li><a href="2010/06">June 2010</a></li>
      <li><a href="2010/05">May 2010</a></li>
      <li><a href="2010/04">April 2010</a></li>
      <li><a href="2010/03">March 2010</a></li>
      <li><a href="2010/02">February 2010</a></li>
      <li><a href="2010/01">January 2010</a></li>
    </ul>
  </nav>
</section>
```

Damit haben wir schon unsere Blog-Struktur. Jetzt können wir damit beginnen, die neuen Elemente zu stylen.

Styling

Genau wie auf `div`-Tags können wir auch auf die neuen Elemente Stilregeln anwenden. Zunächst erstellen wir ein neues Stylesheet mit dem Namen `style.css`. Anschließend verknüpfen wir es mit unserem HTML-Dokument, indem wir im Header einen Link auf das Stylesheet einfügen:

```
html5newtags/index.html
```

```
<link rel="stylesheet" href="style.css" type="text/css">
```

Als Erstes zentrieren wir den Inhalt der Seite und legen einige grundlegenden Schriftarten fest:

html5newtags/style.css

```
body{
  width:960px;
  margin:15px auto;
  font-family: Arial, "MS Trebuchet", sans-serif;
}

p{
  margin:0 0 20px 0;
}

p, li{
  line-height:20px;
}
```

Dann definieren wir die Breite der Kopfzeile:

html5newtags/style.css

```
header#page_header{
  width:100%;
}
```

Für die Navigationslinks wandeln wir die Listen mit Aufzählungszeichen in horizontale Navigationsleisten um:

html5newtags/style.css

```
header#page_header nav ul, #page_footer nav ul{
  list-style: none;
  margin: 0;
  padding: 0;
}

#page_header nav ul li, footer#page_footer nav ul li{
  padding:0;
  margin: 0 20px 0 0;
  display:inline;
}
```

Der Abschnitt posts muss gefloatet werden und eine feste Breite erhalten. Außerdem müssen wir auch den Callout im Artikel floaten. Bei der Gelegenheit vergrößern wir gleich noch die Schrift für den Callout.

html5newtags/style.css

```
section#posts{
  float: left;
  width: 74%;
}

section#posts aside{
  float: right;
  width: 35%;
}
```

Messwerte und Fortschrittsbalken

Wenn Sie ein Spendenbarometer oder einen Fortschrittsbalken für den Upload in einer Webanwendung implementieren möchten, sollten Sie einen Blick auf die Elemente `meter` und `progress` werfen, die mit HTML5 eingeführt wurden.

Mit dem Element `meter` können wir einen festen Punkt auf einer Messskala mit einem Mindest- und einem Höchstwert semantisch beschreiben. Damit Ihr Messwert mit der Spezifikation harmonisiert, sollten Sie ihn nicht für beliebige Minimal- oder Maximalwerte wie Höhe oder Gewicht verwenden, außer Sie haben einen bestimmten Grenzwert festgelegt. Wenn wir beispielsweise auf einer Website zum Sammeln von Spenden zeigen möchten, wie weit wir noch von dem Ziel \$5.000 entfernt sind, lässt sich das so beschreiben:

`html5_meter/index.html`

```
<section id="pledge">
  <header>
    <h3>Our Fundraising Goal</h3>
  </header>
  <meter title="USD" id="pledge_goal"
    value="2500" min="0" max="5000">
    $2500.00
  </meter>
  <p>Help us reach our goal of $5000!</p>
</section>
```

Das Element `progress` ist einem Messwert sehr ähnlich, wurde aber dafür entwickelt, einen aktiven Fortschritt darzustellen, zum Beispiel beim Hochladen einer Datei. Ein Messwert zeigt dagegen eine Messung an, die nicht mehr verändert wird, wie etwa einen Schnappschuss des Speicherplatzes, der auf einem Server einem bestimmten Benutzer noch zur Verfügung steht. Das Markup für einen Fortschrittsbalken ist dem eines `meter`-Elements aber sehr ähnlich:

`html5_meter/progress.html`

```
<progress id="progressbar" max=100><span>0</span>%</progress>
```

Die Elemente `meter` und `progress` lassen sich noch in keinem Browser darstellen. Sie können jedoch mit JavaScript die Werte aus dem `meter`-Element auslesen und selbst abbilden, das Element `meter` oder `progress` also dafür verwenden, die Daten semantisch zu beschreiben. Ein Beispiel dafür finden Sie in den Beispieldateien dieses Buchs für das `meter`-Element.

```
margin-left: 5%;
font-size: 20px;
line-height: 40px;
}
```

Außerdem müssen wir die Seitenleiste floaten und eine feste Breite dafür festlegen:

```
html5newtags/style.css
```

```
section#sidebar{
  float: left;
  width: 25%;
}
```

Und wir müssen die Fußzeile definieren. Wir löschen die Floats in der Fußzeile, sodass sie im unteren Teil der Seite zu liegen kommt.

```
html5newtags/style.css
```

```
footer#page_footer{
  clear: both;
  width: 100%;
  display: block;
  text-align: center;
}
```

Das sind nur die wichtigsten Stilregeln in Kürze. Ich bin mir sicher, dass Sie das alles noch viel, viel besser aussehen lassen können.

Ausweichlösung

Das funktioniert bereits alles wunderbar in Firefox, Chrome und Safari. Allerdings werden die Leute im Management nicht allzu glücklich sein, wenn sie das Chaos sehen, das der Internet Explorer aus unserer Seite macht. Der Inhalt wird zwar wunderbar dargestellt. Aber da der IE diese Elemente nicht kennt, kann er auch keine Stilregeln darauf anwenden. Die Seite erinnert deshalb eher an die Mitte der Neunziger-Jahre.

Mit dem IE können wir diese Elemente nur stylen, wenn wir sie mit JavaScript als Teil des Dokuments definieren. Wie sich herausstellt, ist das wirklich einfach. Wir fügen den Code in den Abschnitt `head` der Seite ein, sodass er ausgeführt wird, bevor der Browser irgendwelche Elemente rendert. Außerdem schreiben wir den Code in einen *bedingten Kommentar* – eine besondere Art von Kommentar, die nur der Internet Explorer verarbeitet.

```
html5newtags/index.html
```

```
<!--[if lt IE 9]>
<script type="text/javascript">
  document.createElement("nav");
  document.createElement("header");
  document.createElement("footer");
  document.createElement("section");
  document.createElement("aside");
  document.createElement("article");
</script>
<![endif]-->
```

Solche Kommentare gelten nur für Versionen des Internet Explorer, die älter als Version 9.0 sind. Wenn wir die Seite jetzt noch einmal laden, wird sie richtig dargestellt.

Allerdings machen wir uns dadurch von JavaScript abhängig. Das sollten Sie bedenken. Für den verbesserten Aufbau und die Lesbarkeit des Dokuments lohnt sich das durchaus. Außerdem gibt es keine Probleme bezüglich der Barrierefreiheit, da der Inhalt nach wie vor angezeigt und von einem Bildschirmlesegerät vorgelesen werden kann. Die Darstellung wirkt lediglich hoffnungslos altmodisch für Benutzer, die JavaScript absichtlich deaktiviert haben.

Mit diesem Ansatz können Sie auch wunderbar zusätzliche Elemente unterstützen oder verstehen, wie Sie eine entsprechende Unterstützung schreiben können. Die brillante Lösung HTMLShiv³ von Remy Sharp treibt die Dinge noch wesentlich weiter, ist aber wahrscheinlich eher dann angebracht, wenn Sie eine Ausweichlösung für viele zusätzliche Elemente integrieren möchten.

3 <http://code.google.com/p/html5shiv/>

2

Pop-up-Fenster mit benutzerdefinierten Datenattributen

Wenn Sie schon mal eine Webanwendung entwickelt haben, die über JavaScript Dokumentinformationen ausliest, dann wissen Sie, dass dafür manchmal ein bisschen Hackerei und Parsen erforderlich sind. Letztendlich fügen Sie zusätzliche Informationen in Event-Handler ein oder missbrauchen die Attribute `rel` oder `class`, damit es funktioniert. Dank der Einführung benutzerdefinierter Datenattribute sind diese Zeiten nun jedoch vorbei.

Benutzerdefinierte Datenattribute beginnen mit dem Präfix `data-` und werden von Validierern für HTML5-Dokumente ignoriert. Sie können benutzerdefinierte Datenattribute zu jedem beliebigen Element hinzufügen – ganz egal, ob es sich um Metadaten zu einem Foto, Längen- und Breitengradkoordinaten oder, wie in diesen Tipp, um die Maße eines Pop-up-Fensters handelt. Das Beste daran ist, dass Sie benutzerdefinierte Datenattribute schon jetzt in beinahe jedem Browser verwenden und sie ganz leicht mit JavaScript ausgelesen können.

Das Verhalten vom Inhalt trennen oder warum onclick böse ist

Im Laufe der Jahre haben Pop-up-Fenster einen schlechten Ruf bekommen – oft auch zu Recht. Sie werden eingesetzt, um Werbung anzuzeigen, um nichts ahnende Websurfer dazu zu bringen, Spyware oder Viren zu installieren, oder sogar, um persönliche Informationen weiterzugeben, die später verkauft werden. Es überrascht daher nicht weiter, dass die meisten Browser über einen Pop-up-Blocker verfügen.

Pop-ups an sich sind aber gar nichts Schlechtes. Entwickler von Webanwendungen verwenden Pop-up-Fenster häufig, um die Online-Hilfe, zusätzliche Optionen oder andere wichtige Funktionen der Benutzeroberfläche anzuzeigen. Damit Pop-ups weniger nerven, müssen wir sie lediglich entsprechend unaufdringlich implementieren. Wenn Sie sich die Seite der Personalabteilung von AwesomeCo ansehen, finden Sie mehrere Links, über die Richtlinien in Pop-up-Fenstern angezeigt werden. Die meisten sehen so aus:

```
html5_popups_with_custom_data/original_example_1.html
```

```
<a href='#'
  onclick="window.open('holiday_pay.html',WinName,'width=300,height=300');">
  Holiday pay
</a>
```

Das ist ein gebräuchlicher Ansatz, um Links zum Öffnen von Pop-ups zu erstellen. Genau genommen ist das der Ansatz, den JavaScript-Neulinge lernen, um Pop-up-Fenster zu öffnen. Diese Methode bringt jedoch einige Probleme mit sich, derer wir uns annehmen sollten, bevor wir weitermachen.

Mehr Barrierefreiheit

Das Ziel des Links wurde nicht festgelegt! Wenn JavaScript deaktiviert ist, führt der Link den Benutzer nicht zur entsprechenden Seite. Das ist ein großes Problem, um das wir uns sofort kümmern müssen. Lassen Sie *niemals* das Attribut `href` aus, und vergeben Sie dafür *unter keinen Umständen* einen Wert wie im vorherigen Beispiel. Geben Sie die Adresse der Ressource an, die normalerweise angezeigt werden soll.

html5_popups_with_custom_data/original_example_2.html

```
<a href='holiday_pay.html'
  onclick="window.open(this.href,WinName, 'width=300,height=300');">
  Holiday pay
</a>
```

Der JavaScript-Code liest dann seinerseits das Ziel des Links aus dem `href`-Attribut des verknüpften Elements aus.

Als erster Schritt zu einer barrierefreien Seite muss die gesamte Funktionalität auch *ohne* JavaScript gewährleistet sein.

onclick beseitigen

Trennen Sie das Verhalten vom Inhalt, genauso wie Sie auch die Darstellungsinformationen in separaten und verknüpften Stylesheets ablegen. Auf den ersten Blick ist `onclick` einfach zu verwenden. Aber wenn Sie sich eine Seite mit 50 Links vorstellen, werden Sie verstehen, dass die `onclick`-Methode schnell außer Kontrolle gerät. Sie wiederholen dasselbe JavaScript immer und immer wieder. Und wenn Sie diesen Code serverseitig generieren, erhöhen Sie lediglich die Anzahl von JavaScript-Events und machen das so entstandene HTML viel umfangreicher als nötig.

Weisen Sie stattdessen jedem Anker auf der Seite eine entsprechende Klasse zu:

html5_popups_with_custom_data/original_example_3.html

```
<a href="holiday_pay" class="popup">Holiday Pay</a>
```

```
html5_popups_with_custom_data/original_example_3.html
```

```
var links = $("a.popup");

links.click(function(event){
    event.preventDefault();
    window.open($(this).attr('href'));
});
```

Mit einem jQuery-Selektor erfassen wir alle Elemente der Klasse `popup`. Anschließend fügen wir einen Listener zum `click`-Event der jeweiligen Elemente hinzu. Der an die `click`-Methode übergebene Code wird ausgeführt, wenn jemand auf den Link klickt. Die Methode `preventDefault` unterbindet dabei das standardmäßige Verhalten für das `click`-Event. In diesem Fall verhindert sie, dass der Browser dem Link folgt und eine neue Seite anzeigt.

Was wir dabei im Vergleich zum ursprünglichen Beispiel verlieren, sind allerdings die Informationen zu Größe und Position des Fensters. Wir möchten aber, dass auch ein Designer, der sich nicht allzu gut mit JavaScript auskennt, für jeden Link die Maße des Fensters festlegen kann.

Benutzerdefinierte Datenattribute zu Hilfe!

Bei der Erstellung von JavaScript-gestützten Anwendungen kommt dieses Problem häufig vor. Wie wir gesehen haben, möchten wir die gewünschte Höhe und Breite im Code ablegen. Der Ansatz mit `onclick` hat aber einige Nachteile. Stattdessen können wir diese Informationen als Attribute des jeweiligen Elements einbetten. Hierfür müssen wir lediglich den Link folgendermaßen aufbauen:

```
html5_popups_with_custom_data/popup.html
```

```
<a href="help/holiday_pay.html"
  data-width="600"
  data-height="400"
  title="Holiday Pay"
  class="popup">Holiday pay</a>
```

Jetzt ändern wir einfach das `click`-Event, das wir gerade geschrieben haben, so, dass es die Optionen aus den benutzerdefinierten Datenattributen des Links ausliest und an die Methode `window.open` übergibt:

```
html5_popups_with_custom_data/popup.html
```

```
$(function(){
  $(".popup").click(function(event){
    event.preventDefault();
    var href = $(this).attr("href");
    var width = $(this).attr("data-width");
    var height = $(this).attr("data-height");
    var popup = window.open(href, "popup",
      "height=" + height + ",width=" + width + "");
  });
});
```

Das ist alles! Der Link wird nun in einem neuen Fenster geöffnet.

Eine kleine Warnung

In diesem Beispiel haben wir benutzerdefinierte Datenattribute verwendet, um zusätzliche Informationen an das clientseitige Skript zu übergeben. Das ist ein cleverer Ansatz, um ein bestimmtes Problem zu lösen und eine mögliche Verwendung dieser Attribute zu zeigen. Zwar wird unser Markup dadurch mit Darstellungsinformationen vermischt, dafür sehen Sie aber, wie leicht es ist, mit JavaScript in Ihre Seite eingebettete Werte auszulesen.

Ausweidlösung

Diese Attribute funktionieren schon jetzt auch in älteren Browsern, solange diese JavaScript unterstützen. Der Browser hat keine Schwierigkeiten mit benutzerdefinierten Datenattributen. Außerdem ist Ihr Dokument auch valide, da Sie den Doctype HTML5 verwenden und Attribute, die mit `data-` beginnen, daher ignoriert werden.

Die Zukunft

Sobald diese neuen Tags und Attribute auf breiter Basis unterstützt werden, können wir einige interessante Dinge damit anstellen. So können wir beispielsweise die Navigation und Fußzeilen von Artikeln kennzeichnen und in Stylesheets für den Druck ganz einfach deaktivieren.

```
nav, article>footer{display:none}
```

Mit Skriptsprachen können wir schnell alle Artikel auf einer Seite oder einer Website erkennen. Und noch besser: Wir zeichnen Inhalte mit Tags aus, die diese korrekt beschreiben, und können so bessere Stylesheets und besseres JavaScript schreiben.

Benutzerdefinierte Datenattribute bieten Entwicklern die Flexibilität, alle Arten von Informationen in ihr Markup einzubetten. Wir verwenden sie wieder in Kapitel 6, *Zeichnen mit dem canvas-Element*, auf Seite 113.

In Verbindung mit JavaScript können Sie mit benutzerdefinierten Datenattributen festlegen, ob ein `form`-Tag über Ajax übertragen werden soll. Dafür müssen Sie lediglich einem beliebigen `form`-Tag `data-remote=true` zuweisen – so macht es zum Beispiel das *Ruby on Rails*-Framework.

Außerdem können Sie damit Datum und Uhrzeit in der Zeitzone Ihrer Benutzer anzeigen, auch wenn sich die Seite im Cache befindet. Dazu geben Sie das Datum in der HTML-Seite einfach als UTC an und konvertieren es clientseitig in das lokale Format der Benutzer. Mit diesen Attributen können Sie real greifbare und brauchbare Daten in Ihre Seiten einbetten und sich darauf verlassen, dass immer mehr Frameworks und Bibliotheken diese nutzen. Ich bin mir sicher, Sie finden dafür jede Menge großartiger Anwendungsmöglichkeiten in Ihrer eigenen Arbeit.

Und außerdem können wir damit der Divitis ein für alle Mal den Gar-aus machen!

Kapitel 3

Benutzerfreundliche Webformulare

Falls Sie jemals eine komplexere Benutzeroberfläche gestaltet haben, wissen Sie, was die einfachen HTML-Formularelemente für eine Einschränkung darstellen. Sie müssen sich mit Textfeldern, Auswahlménüs, Optionsschaltflächen, Kontrollkästchen und manchmal sogar mit den noch klobigeren Mehrfach-Auswahllisten herumärgern, die Sie dann auch noch Ihren Benutzern erklären müssen. („Halten Sie die Strg-Taste gedrückt, und klicken Sie auf die gewünschten Einträge. Außer Sie verwenden einen Mac. In diesem Fall drücken Sie die Cmd-Taste.

Also tun Sie, was alle guten Webentwickler tun – Sie nehmen Prototype, jQuery oder basteln Ihre eigenen Steuerelemente und Funktionen aus einer Kombination von HTML, CSS und JavaScript. Aber wenn Sie sich ein Formular mit Slidern, Kalendersteuerelementen, Spinboxen, Autovervollständigungsfeldern und visuellen Editoren ansehen, merken Sie schnell, was Sie sich da für einen Albtraum eingebrockt haben. Sie müssen sicherstellen, dass die von Ihnen eingefügten Steuerelemente keine Konflikte mit den anderen Steuerelementen von Ihnen oder aus anderen JavaScript-Bibliotheken auf der Seite verursachen. Sie können Stunden damit verwenden, einen Kalender-Picker zu implementieren, nur um dann festzustellen, dass die Prototype-Bibliothek Probleme hat, weil jQuery die Funktion `$()` übernommen hat. Also verwenden Sie die Methode `noConflict()` von jQuery. Allerdings stellen Sie dann fest, dass der von Ihnen verwendete Farbwähler nicht mehr funktioniert, weil das Plugin nicht sorgfältig genug programmiert wurde.

Falls Sie jetzt lächeln, liegt das daran, dass Sie das alles schon erlebt haben. Falls Sie vor Wut kochen, dann aus demselben Grund. Es gibt jedoch Hoffnung. In diesem Kapitel erstellen wir einige Webformulare mit neuen Arten von Formularfeldern. Und wir implementieren dabei auch Autofokus und Platzhaltertexte.

Zum Abschluss zeigen wir Ihnen, wie Sie mit dem neuen Attribut `contenteditable` jedes beliebige HTML-Feld in ein Steuerelement umwandeln, das Ihre Benutzer bearbeiten können.

Im Einzelnen behandeln wir die folgenden Funktionen:¹

E-Mail-Feld [`<input type="email">`]

Zeigt ein Formularfeld für E-Mail-Adressen an. *[O10.1, IOS]*

URL-Feld [`<input type="url">`]

Zeigt ein Formularfeld für URLs an. *[O10.1, IOS]*

Telefonnummern-Feld [`<input type="tel">`]

Zeigt ein Formularfeld für Telefonnummern an. *[O10.1, IOS]*

Suchfeld [`<input type="search">`]

Zeigt ein Formularfeld für Suchschlüsselwörter an. *[C5, S4, O10.1, IOS]*

Slider (Bereich) [`<input type="range">`]

Zeigt ein Slider-Steuerelement an. *[C5, S4, O10.1]*

Zahl [`<input type="number">`]

Zeigt ein Formularfeld für Zahlen an, häufig als Spinbox.
[C5, S5, O10.1, IOS]

Datumsfelder [`<input type="date">`]

Zeigt ein Formularfeld für Datumswerte an. Unterstützt `date`, `month` oder `week`. *[C5, S5, O10.1]*

Datumswerte mit Uhrzeit [`<input type="datetime">`]

Zeigt ein Formularfeld für Datumswerte mit Uhrzeit an. Unterstützt `datetime`, `datetime-local` oder `time`. *[C5, S5, O10.1]*

Farbe [`<input type="color">`]

Zeigt ein Feld zum Auswählen von Farben an. *[C5, S5]* (Chrome 5 und Safari 5 verstehen das `color`-Feld, zeigen aber kein spezielles Steuerelement dafür an.)

¹ In den folgenden Beschreibungen wird die Unterstützung durch die verschiedenen Browser in eckigen Klammern mit einem Kurzcode und der mindestens erforderlichen Versionsnummer angegeben. Die verwendeten Codes lauten: *C*: Google Chrome, *F*: Firefox, *IE*: Internet Explorer, *O*: Opera, *S*: Safari, *IOS*: iOS-Geräte mit Mobile Safari und *A*: Android-Browser.

Autofokus [`<input type="text" autofocus>`]

Bietet die Möglichkeit, einem bestimmten Formularelement automatisch den Fokus zu erteilen. *[C5, S4]*

Unterstützung für Platzhalter

[`<input type="email" placeholder="me@example.com">`]

Ermöglicht die Anzeige von Platzhaltertext in einem Formularfeld.
[C5, S4, F4]

Unterstützung für In-Place-Editing

[`<p contenteditable>loremipsum</p>`]

Unterstützung für das In-Place-Editing von Inhalten im Browser.
[C4, S3.2, IE6, O10.1]

Fangen wir damit an, etwas über die extrem nützlichen neuen Formularfeldtypen zu lernen.

3**Daten mit neuen Eingabefeldern beschreiben**

In HTML5 werden mehrere neue Typen von Eingabefeldern eingeführt, mit denen Sie den Datentyp besser beschreiben können, den Ihre Benutzer eingeben sollen. Zusätzlich zu den standardmäßigen Textfeldern, Optionsschaltflächen und Kontrollkästchen können Sie Elemente wie E-Mail-Felder, Kalender, Farbwähler, Spinboxen und Slider verwenden. Browser bieten den Benutzern für diese neuen Felder besser funktionierende Steuerelemente an – auch ohne JavaScript. Mobile Geräte und virtuelle Tastaturen für Tablet-PCs und Touchscreens können für diese Feldtypen unterschiedliche Tastaturlayouts anzeigen. So zeigt beispielsweise der iPhone-Browser Mobile Safari ein anderes Tastaturlayout an, wenn Benutzer Daten vom Typ URL oder email eingeben, wodurch Sonderzeichen wie `@`, `.`, `:` und `/` leichter zugänglich sind.

Verbessertes Formular für unser AwesomeCo-Projekt

AwesomeCo arbeitet an einer neuen Webanwendung für das Projektmanagement, die es Entwicklern und Managern erleichtern soll, bezüglich der aktuellen Projekte auf dem neuesten Stand zu bleiben. Jedes Projekt hat einen Namen, eine Kontakt-E-Mail-Adresse sowie eine Test-URL, damit Manager während der Entwicklungsphase eine Vorschau der Website ansehen können. Es gibt auch Felder für das Startdatum, für die Priorität sowie für die geschätzte Stundenzahl bis zur Fertigstellung des Projekts. Außerdem möchte der Entwicklungsmanager die Möglichkeit haben, jedem Projekt eine Farbe zuzuweisen, damit er in Berichten das jeweilige Projekt schnell erkennen kann.

Bauen wir also mit den neuen HTML5-Feldern ein Gerüst der Seite für die Projekteinstellungen.

Das grundlegende Formular

Wir erstellen ein einfaches HTML-Formular, das eine POST-Anfrage sendet. Nachdem es mit dem Namensfeld nichts Besonderes auf sich hat, verwenden wir dafür ein text-Feld.

html5forms/index.html

```

<form method="post" action="/projects/1">

  <fieldset id="personal_information">
    <legend>Project Information</legend>
    <ol>
      <li>
        <label for="name">Name</label>
        <input type="text" name="name" autofocus id="name">
      </li>
      <li>
        <input type="submit" value="Submit">
      </li>
    </ol>
  </fieldset>

</form>

```

Beachten Sie, dass wir die Beschriftungen für das Formular in eine geordnete Liste schreiben. Beschriftungen sind von grundlegender Bedeutung bei der Erstellung barrierefreier Formulare. Das Attribut `for` referenziert die `id` des entsprechenden Formularelements. Dies hilft Bildschirmlesegeräten bei der Erkennung der Felder auf einer Seite. Die geordnete Liste bietet eine gute Möglichkeit, die Felder aufzulisten, ohne auf eine komplizierte Tabelle oder `div`-Strukturen zurückzugreifen. Außerdem können Sie das Markup auf diese Weise auch in der Reihenfolge schreiben, in der Ihre Besucher die Felder ausfüllen sollen.

Slider mithilfe von `range` erstellen

Slider werden häufig verwendet, damit Benutzer einen numerischen Wert bequem vergrößern oder verkleinern können. Dies ist eine ausgezeichnete Lösung, damit Manager schnell die Priorität eines Projekts visualisieren und anpassen können. Einen Slider implementieren Sie mit dem Typ `range`.

html5forms/index.html

```

<label for="priority">Priority</label>
<input type="range" min="0" max="10"
      name="priority" value="0" id="priority">

```

Fügen Sie diesen Code genau wie das vorherige Feld in einem `li`-Element in das Formular ein.

Chrome und Opera implementieren beide ein Slider-Widget, das folgendermaßen aussieht:

Priority



Beachten Sie, dass wir den Wertebereich des Sliders mit min und max festgelegt haben. Dadurch wird auch der Wertebereich des Formularfelds eingegrenzt.

Zahlen und Spinboxen

Wir verwenden sehr häufig Zahlen. Und obwohl das Eingeben von Zahlen relativ einfach ist, können Spinboxen kleinere Anpassungen erleichtern. Eine Spinbox ist ein Steuerelement mit Pfeilen, über die der Wert im Feld erhöht oder verringert werden kann. Wir verwenden eine Spinbox für die geschätzte Stundenzahl. Auf diese Weise können die Stunden einfach angepasst werden.

html5forms/index.html

```
<label for="estimated_hours">Estimated Hours</label>
<input type="number" name="estimated_hours"
       min="0" max="1000"
       id="estimated_hours">
```

Opera unterstützt das Spinbox-Steuerelement. Es sieht folgendermaßen aus:

Estimated Hours



Standardmäßig können auch Werte direkt in die Spinbox eingetippt werden. Ebenso wie beim range-Slider können wir einen Mindest- und einen Maximalwert festlegen. Der so festgelegte Wertebereich gilt jedoch nicht für Werte, die direkt in das Feld eingetippt werden.

Beachten Sie auch, dass Sie die Größe der Schritte beim Erhöhen oder Verringern des Wertes festlegen können, indem Sie den Parameter step angeben. Dieser beträgt standardmäßig 1, kann aber ein beliebiger numerischer Wert sein.

Datumswerte

Die Erfassung des Startdatums für das Projekt ist ziemlich wichtig, wir möchten es daher so einfach wie möglich gestalten. Der input type date ist dafür die perfekte Wahl.

html5forms/index.html

```
<label for="start_date">Start date</label>
<input type="date" name="start_date" id="start_date"
      value="2010-12-01">
```

Als dieses Buch geschrieben wurde, war Opera der einzige Browser, der einen vollwertigen Kalender-Picker unterstützt.

Hier ein Beispiel für die Implementierung:

The screenshot shows a web form with the following elements:

- Start date:** A date picker showing '2010-12-0' with a dropdown arrow.
- Email contact:** A text input field.
- Staging URL:** A text input field.
- Project color:** A text input field.
- Submit:** A button.

The date picker is open, showing a calendar for December 2010. The calendar has a table structure with columns for days of the week (Mon, Tue, Wed, Thu, Fri, Sat, Sun) and rows for weeks. The date 2010-12-01 is selected.

Week	Mon	Tue	Wed	Thu	Fri	Sat	Sun
48	29	30	1	2	3	4	5
49	6	7	8	9	10	11	12
50	13	14	15	16	17	18	19
51	20	21	22	23	24	25	26
52	27	28	29	30	31	1	2
1	3	4	5	6	7	8	9

At the bottom of the calendar, there are two buttons: 'Today' and 'None'.

Safari 5.0 zeigt ein Feld, das dem number-Feld ähnlich ist und über Pfeile verfügt, mit denen das Datum erhöht oder verringert werden kann. Wird es leer gelassen, steht der Standardwert auf „1582“. Andere Browser rendern ein Textfeld.

E-Mail

Die HTML5-Spezifikation legt fest, dass der input type email für die Anzeige einer einzelnen E-Mail-Adresse oder einer Liste von E-Mail-Adressen entwickelt wurde. Er ist daher die perfekte Wahl für unser E-Mail-Feld.

html5forms/index.html

```
<label for="email">Email contact</label>
<input type="email" name="email" id="email">
```

Auf mobilen Geräten kommt der größtmögliche Nutzen dieser Art von Formularfeld zum Tragen, da das Layout der virtuellen Tastatur angepasst und so die Eingabe von E-Mail-Adressen entsprechend vereinfacht wird.

URL

Es gibt auch einen eigenen Feldtyp für URLs. Das ist besonders praktisch, wenn Ihre Besucher ein iPhone verwenden. Denn dann wird ein ganz anderes Tastaturlayout mit Hilfsschaltflächen für die schnelle Eingabe von Webadressen angezeigt, ganz ähnlich wie die Tastatur zur Eingabe einer URL in die Adressleiste von Mobile Safari. Das Feld für die Eingabe der Test-URL lässt sich mit dem folgenden Code einfach einfügen:

```
html5forms/index.html
```

```
<label for="url">Staging URL</label>
<input type="url" name="url" id="url">
```

Virtuelle Tastaturen zeigen für diesen Feldtyp ebenfalls ein anderes Layout an.

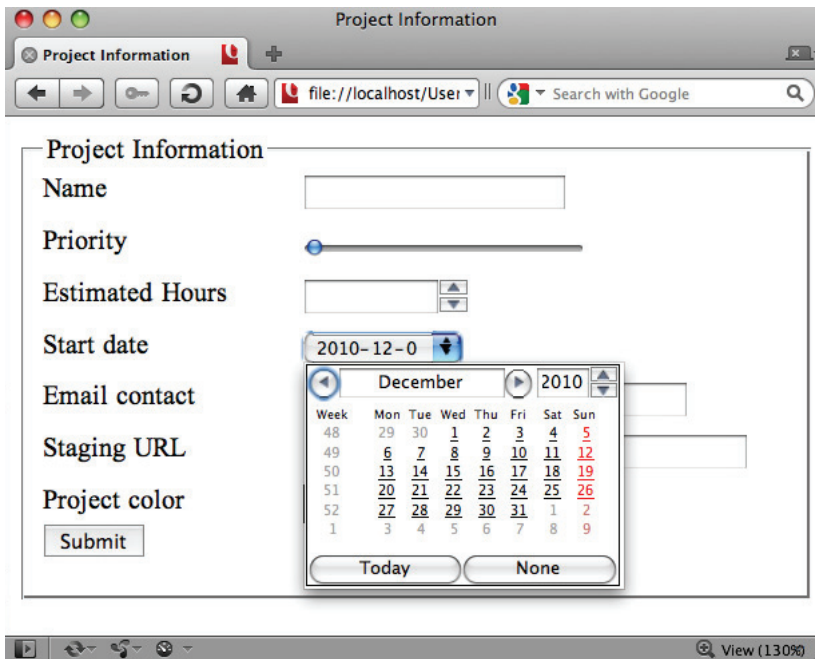


Abbildung 3.1: Manche Formularsteuerelemente werden von Opera bereits unterstützt.

Farbe

Zu guter Letzt müssen wir noch eine Möglichkeit finden, Farbcodes einzugeben. Dafür verwenden wir den Typ `color`.

html5forms/index.html

```
<label for="project_color">Project color</label>
<input type="color" name="project_color" id="project_color">
```

Als dieses Buch geschrieben wurde, zeigte kein Browser ein Farbwähler-Steuererelement an, aber das soll Sie nicht davon abhalten, es zu verwenden. Sie schreiben korrekten Markup für die Beschreibung Ihres Inhalts, und das wird sich in Zukunft als praktisch erweisen. Vor allem, wenn Sie Ausweichlösungen bereitstellen müssen.

Opera unterstützt bereits jetzt die meisten der neuen Steuerelemente, wie Sie in Abbildung 3.1 sehen können. Öffnen Sie die Seite dagegen in Firefox, Safari oder Google Chrome, werden Sie keinen nennenswerten Unterschied erkennen. Das müssen wir ändern.

Ausweichlösungen

Browser, die die neuen Typen nicht verstehen, verwenden stattdessen den Typ `text`. Ihre Formulare bleiben also trotzdem verwendbar. Außerdem können Sie eines der jQueryUI- oder YUI-Widgets mit dem Feld verknüpfen, um es entsprechend anzupassen. Wenn im Laufe der Zeit immer mehr Browser diese Steuerelemente unterstützen, können Sie die JavaScript-Krücken nach und nach entfernen.

Den Farbwähler ersetzen

Den Farbwähler können wir mit jQuery und den Attributselektoren von CSS3 einfach suchen und ersetzen. Wir suchen alle `input`-Felder vom Typ `color` und wenden das jQuery-Plugin „SimpleColor“ darauf an.

html5forms/index.html

```
if (!hasColorSupport()){
  $('input[type=color]').simpleColor();
}
```

Da wir die neuen Formulartypen in unserem Markup verwenden, brauchen wir keinen zusätzlichen Klassennamen oder anderes Markup einfügen, um Farbwähler zu kennzeichnen. Attributselektoren und HTML5 ergänzen sich prima.

Wir möchten das Farbwähler-Plugin nicht verwenden, wenn der Browser es nativ unterstützt. Also ermitteln wir mit JavaScript, ob der Browser Eingabefelder unterstützt, deren `type` den Wert `color` hat.

html5forms/index.html

```

1 function hasColorSupport(){
2     input = document.createElement("input");
3     input.setAttribute("type", "color");
4     var hasColorType = (input.type !== "text");
5     // Lösung für teilweise Implementierung in Safari/Chrome
6     if(hasColorType){
7         var testString = "foo" ;
8         input.value=testString;
9         hasColorType = (input.value != testString);
10    }
11    return(hasColorType);
12 }

```

Zunächst erstellen wir mit JavaScript ein Element und legen das type-Attribut auf den Wert color fest. Anschließend lesen wir das type-Attribut wieder aus, um festzustellen, ob wir das Attribut erfolgreich setzen konnten. Wenn wir den Wert color zurückbekommen, wird dieser Typ unterstützt. Falls nicht, müssen wir unser Skript anwenden.

Interessant wird es in Zeile 6. Safari 5 und Google Chrome 5 haben den Typ color teilweise implementiert. Sie unterstützen das Feld, zeigen aber kein Farb-Widget an, und so erhalten wir trotzdem nur ein Textfeld. Daher legen wir in unserer Prüfmethode einen Wert für das Eingabefeld fest und überprüfen, ob der Wert erhalten bleibt. Falls nein, können wir davon ausgehen, dass der Browser einen Farbwähler implementiert hat, da sich das Eingabefeld nicht wie ein Textfeld verhält.

Der vollständige Code zum Ersetzen des Farbwählers sieht folgendermaßen aus:

html5forms/index.html

```

if (!hasColorSupport()){
    $('input[type=color]').simpleColor();
}

```

Diese Lösung funktioniert, ist aber ein bisschen wackelig. Sie richtet sich nur an bestimmte Browser und funktioniert auch nur für das color-Steuerelement. Andere Steuerelemente haben wieder ganz andere Macken, mit denen Sie sich ebenfalls auskennen müssen. Glücklicherweise gibt es eine Alternative.

Modernizr

Die Modernizr-Bibliothek² kann die Unterstützung für viele HTML5- und CSS3-Funktionen überprüfen. Sie implementiert zwar nicht die fehlende Funktionalität, bietet aber mehrere Mechanismen für die Ermittlung von Formularfeldern, die ähnlich wie unsere Lösungen, aber zuverlässiger funktionieren.

Bevor Sie Modernizr in Ihren Projekten einsetzen, sollten Sie sich etwas Zeit nehmen zu verstehen, wie er funktioniert. Ob Sie den Code nun selbst geschrieben haben oder nicht, sobald Sie ihn in Ihrem Projekt einsetzen, sind Sie auch dafür verantwortlich. Modernizr war nicht von Anfang an so weit, mit der teilweisen Unterstützung des Farbfelds in Safari klarzukommen. Wenn die nächste Version von Chrome oder Firefox herauskommt, müssen Sie unter Umständen selbst eine Lösung basteln. Wer weiß, vielleicht können Sie ja genau diese Lösung zu Modernizr beitragen!

Ausweichlösungen für Steuerelemente wie den Datums-Picker und den Slider implementieren Sie auf dieselbe Weise. Slider und Datums-Picker sind als Komponenten in der jQuery UI-Bibliothek enthalten.³ Sie binden die jQuery UI-Bibliothek in die Seite ein und überprüfen, ob der Browser das Steuerelement nativ unterstützt. Falls nein, verwenden Sie stattdessen die JavaScript-Version.

Nach und nach können Sie die JavaScript-Steuerelemente ausrangieren und ausschließlich auf die Steuerelemente des Browsers zurückgreifen. Aufgrund der Komplexität bei der Ermittlung dieser Typen wird Ihnen Modernizr eine große Hilfe sein. Allerdings werden wir im Rest dieses Buchs auch weiterhin unsere eigenen Prüftechniken schreiben, damit Sie sehen können, wie sie funktionieren.

Neben den neuen Formularfeldtypen werden in HTML5 auch einige andere Attribute für Formularfelder eingeführt, die dabei helfen können, die Benutzerfreundlichkeit zu verbessern. Sehen wir uns als Nächstes autofocus an.

² <http://www.modernizr.com/>

³ <http://jqueryui.com/>

4

Mit autofocus zum ersten Feld springen

Sie können die Dateneingabe deutlich beschleunigen, wenn Sie schon beim Laden der Seite den Cursor des Benutzers im ersten Feld des Formulars platzieren. Viele Suchmaschinen machen das mit JavaScript, aber in HTML5 ist diese Funktionalität jetzt Teil der Sprache.

Dafür müssen Sie lediglich das Attribut `autofocus` zu einem beliebigen Formularfeld hinzufügen, genau wie wir das schon auf der Profilseite im Abschnitt *Daten mit neuen Eingabefeldern beschreiben* auf Seite 48 gemacht haben.

```
html5forms/index.html
```

```
<label for="name">Name</label>
<input type="text" name="name" autofocus id="name">
```

Damit das Attribut `autofocus` zuverlässig funktioniert, dürfen Sie es nur einmal pro Seite verwenden. Wenn Sie es auf mehr als ein Feld pro Seite anwenden, setzt der Browser den Cursor auf das zuletzt „autofokussierte“ Formularfeld.

Ausweichlösung

Wir überprüfen mit JavaScript, ob das `autofocus`-Attribut vorhanden ist. Wenn der Browser keine Unterstützung für `autofocus` bietet, setzen wir den Fokus mit jQuery. Das ist bestimmt die einfachste Ausweichlösung, die es gibt.

```
html5forms/autofocus.js
```

```
function hasAutofocus() {
    var element = document.createElement('input');
    return 'autofocus' in element;
}

$(function(){
    if(!hasAutofocus()){
        $('input[autofocus=true]').focus();
    }
});
```

Fügen Sie einfach dieses JavaScript in Ihre Seite ein, und Sie haben jederzeit autofocus-Unterstützung, wenn Sie sie brauchen.

Mit autofocus erleichtern Sie Ihren Benutzern die Arbeit mit Ihren Formularen, unmittelbar nachdem sie geladen wurden. Aber Sie möchten ihnen auch etwas klarer sagen, welche Art von Informationen sie eingeben sollen. Sehen wir uns daher als Nächstes das `placeholder`-Attribut an.

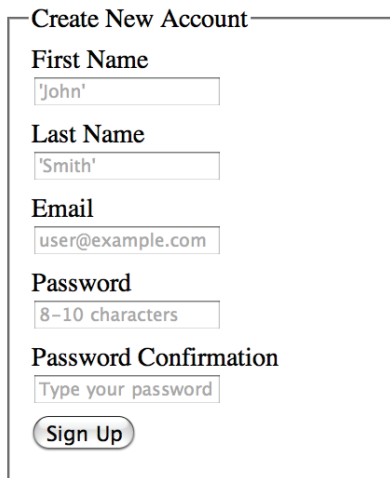
5

Platzhaltertext für Hinweise nutzen

Mit Platzhaltertext können Sie Benutzern Anweisungen geben, womit sie das entsprechende Feld ausfüllen sollen. Abbildung 3.2 zeigt ein Registrierungsformular mit Platzhaltertexten. Dieses Formular bauen wir jetzt nach.

Ein einfaches Registrierungsformular

Benutzer der Support-Website von AwesomeCo müssen zunächst ein Konto erstellen. Eines der größten Probleme bei der Registrierung ist, dass Benutzer immer wieder versuchen, unsichere Kennwörter zu wählen. Wir werden unsere Benutzer mit Platzhaltertext an die Hand nehmen und ihnen mitteilen, welche Anforderungen wir an Kennwörter haben. Der Konsistenz halber werden wir auch die anderen Felder mit Platzhaltertexten versehen.



Create New Account

First Name
John

Last Name
Smith

Email
user@example.com

Password
8-10 characters

Password Confirmation
Type your password

Sign Up

Abbildung 3.2: Platzhalter helfen Benutzern zu verstehen, was Sie von ihnen erwarten.

Zum Hinzufügen von Platzhaltertexten müssen Sie lediglich für jedes Eingabefeld das `placeholder`-Attribut angeben:

```
html5placeholder/index.html
```

```
<input id="email" type="email"
      name="email" placeholder="user@example.com">
```

Das Markup für unser Formular mit Platzhaltertexten für jedes Feld sieht in etwa so aus:

```
html5placeholder/index.html
```

```
<form id="create_account" action="/signup" method="post">
  <fieldset id="signup">
    <legend>Create New Account</legend>
    <ol>
      <li>
        <label for="first_name">First Name</label>
        <input id="first_name" type="text"
          autofocus="true"
          name="first_name" placeholder="'John'">
      </li>
      <li>
        <label for="last_name">Last Name</label>
        <input id="last_name" type="text"
          name="last_name" placeholder="'Smith'">
      </li>
      <li>
        <label for="email">Email</label>
        <input id="email" type="email"
          name="email" placeholder="user@example.com">
      </li>
      <li>
        <label for="password">Password</label>
        <input id="password" type="password" name="password" value=""
          autocomplete="off" placeholder="8-10 characters" />
      </li>
      <li>
        <label for="password_confirmation">Password Confirmation</label>
        <input id="password_confirmation" type="password"
          name="password_confirmation" value=""
          autocomplete="off" placeholder="Type your password again" />
      </li>
      <li><input type="submit" value="Sign Up"></li>
    </ol>
  </fieldset>
</form>
```

Autovervollständigung verhindern

Vielleicht ist Ihnen aufgefallen, dass wir für die Kennwortfelder des Formulars das Attribut `autocomplete` angegeben haben. In HTML5 wird das Attribut `autocomplete` eingeführt, um Webbrowser mitzuteilen, dass sie nicht versuchen sollen, die Daten für dieses Feld automatisch auszufüllen. Einige Browser merken sich Daten, die die Benutzer zuvor eingetippt haben. Aber in manchen Fällen möchten wir den Browser anweisen, das lieber nicht zu tun.

Da wir wieder einmal eine geordnete Liste als Container für unsere Formularfelder verwenden, schreiben wir ein bisschen CSS, damit das Formular besser aussieht.

```
html5placeholder/style.css
```

```
fieldset{
  width: 216px;
}

fieldset ol{
  list-style:
  none;
  padding:0;
  margin:2px;
}

fieldset ol li{
  margin:0 0 9px 0;
  padding:0;
}

/* Jedes Eingabefeld erhält eine eigene Zeile */
fieldset input{
  display:block;
}
```

Safari, Opera und Chrome zeigen nun in den Formularfeldern einen hilfreichen Text für die Benutzer an. Als Nächstes bringen wir Firefox und den Internet Explorer dazu, ebenfalls mitzuspielen.

Ausweichlösung

Mit JavaScript können Sie Platzhaltertext ohne größeren Aufwand in Formularfelder schreiben. Sie testen den Wert jedes Formularfelds. Und wenn er leer ist, belegen Sie den Wert mit dem Platzhalter. Erhält das Formularfeld den Fokus, löschen Sie den Wert wieder. Und wenn das Feld den Fokus verliert, überprüfen Sie den Wert erneut. Falls er sich geändert hat, lassen Sie ihn unverändert. Und falls der Wert leer ist, ersetzen Sie ihn wieder durch den Platzhaltertext.

Die Unterstützung für placeholder ermitteln Sie genauso wie die Unterstützung für autofocus.

```
html5placeholder/index.html
```

```
function hasPlaceholderSupport() {
  var i = document.createElement('input');
  return 'placeholder' in i;
}
```

Dann schreiben Sie das JavaScript, um die Änderungen zu verarbeiten. Um das zum Laufen zu bekommen, verwenden wir eine Lösung, die auf der Arbeit von Andrew January⁴ und anderen basiert. Wir füllen die Werte aller Formularfelder mit dem Text, der im Attribut `placeholder` abgelegt ist. Wählen die Benutzer ein Feld aus, entfernen wir den Text aus dem Feld. Das Ganze verpacken wir in ein jQuery-Plugin, damit wir dieses Verhalten einfach auf unser Formular anwenden können. Im Kasten auf Seite 63 erfahren Sie, wie Plugins funktionieren.

html5placeholder/jquery.placeholder.js

```

1 (function($){
2
3   $.fn.placeholder = function(){
4
5     function valueIsPlaceholder(input){
6       return ($(input).val() == $(input).attr("placeholder"));
7     }
8     return this.each(function() {
9
10      $(this).find(":input").each(function(){
11
12        if($(this).attr("type") == "password"){
13
14          var new_field = $("

```

4 Das ursprüngliche Skript finden Sie unter <http://www.morethannothing.co.uk/wp-content/uploads/2010/01/pplaceholder.js>. Es unterstützt aber keine Kennwörterfelder im IE.

```

38         }else{
39
40             // Wert wird durch den Platzhaltertext ersetzt.
41             // Durch den optionalen Parameter "reload" wird
42             // verhindert, dass FF und IE die Werte der Felder
43             // zwischenspeichern.
44             function showPlaceholder(input, reload){
45                 if( $(input).val() == "" ||
46                     ( reload && valueIsPlaceholder(input) ) ){
47                     $(input).val($(input).attr("placeholder"));
48                 }
49             };
50
51             $(this).focus(function(){
52                 if($(this).val() == $(this).attr("placeholder")){
53                     $(this).val("");
54                 };
55             });
56
57             $(this).blur(function(){
58                 showPlaceholder($(this), false)
59             });
60
61             showPlaceholder(this, true);
62         };
63     });
64
65     // Verhindern, dass Formulare die Standardwerte übermitteln
66     $(this).submit(function(){
67         $(this).find(":input").each(function(){
68             if($(this).val() == $(this).attr("placeholder")){
69                 $(this).val("");
70             }
71         });
72     });
73
74 });
75 };
76
77 })(jQuery);

```

In diesem Plugin stehen einige interessante Dinge, die Sie wissen sollten. In Zeile 46 laden wir den Platzhaltertext erneut in die Felder, wenn diese keinen Wert haben oder die Seite neu geladen wurde. Firefox und andere Browser speichern die Formularwerte. Wir belegen das `value`-Attribut mit dem Platzhalter und möchten auf keinen Fall, dass dieser versehentlich zum Wert des Benutzers wird. Wenn wir die Seite laden, übergeben wir `true` an diese Methode, wie Sie in Zeile 61 sehen.

jQuery-Plugins

Sie können jQuery um eigene Plugins erweitern. Dazu fügen Sie eigene Methoden zur jQuery-Funktion hinzu, und schon steht Ihr Plugin nahtlos allen Entwicklern zur Verfügung, die Ihre Bibliothek einbinden. Hierzu ein triviales Beispiel, das eine JavaScript-Alertbox anzeigt:

```
jQuery.fn.debug = function() {
    return this.each(function(){
        alert(this.html());
    });
};
```

Wenn Sie nun für jeden Absatz einer Seite ein Pop-up-Feld anzeigen möchten, geht das so:

```
$("p").debug();
```

jQuery-Plugins sind so konzipiert, dass sie eine Sammlung von jQuery-Objekten durchlaufen. Gleichzeitig liefern sie diese Objektsammlung auch zurück, damit mehrere Methoden miteinander verkettet werden können. Da unser debug-Plugin ebenfalls die jQuery-Sammlung zurückliefert, können wir in derselben Zeile gleich noch mit der `css`-Methode von jQuery die Textfarbe der Absätze ändern:

```
$("p").debug().css("color", "red");
```

Wir werden im Laufe dieses Buchs mehrere Male jQuery-Plugins für Ausweichlösungen einsetzen, um unseren Code klar zu strukturieren. Weitere Informationen dazu können Sie auf der Dokumentationswebsite von jQuery erfahren.^a

a. <http://docs.jquery.com/Plugins/Authoring>

Kennwortfelder verhalten sich ein bisschen anders als andere Formularfelder. Also müssen wir auch anders damit umgehen. Werfen Sie einen Blick auf Zeile 12. Wir überprüfen, ob ein Kennwortfeld vorhanden ist, und müssen seinen Typ in ein reguläres Textfeld ändern, damit der Wert nicht mit Sternchen maskiert wird. Manche Browser melden Fehler, wenn Sie versuchen, Kennwortfelder zu konvertieren. Also müssen wir die Kennwortfelder durch Textfelder ersetzen. Wir werden die Felder austauschen, während die Benutzer damit interagieren.

Durch diesen Trick werden die Werte der Formulare verändert. Im Zweifel möchten Sie aber verhindern, dass die Platzhalter zum Server gelangen. Nachdem wir den Platzhalter-Code nur einsetzen können, wenn JavaScript aktiviert ist, können wir auch mit JavaScript die Übermittlung des Formulars überprüfen. In Zeile 66 fangen wir daher die Übermittlung des Formulars ab und löschen alle Werte aus allen Eingabefeldern, die den Platzhalterwerten entsprechen.

Nachdem wir jetzt alles als Plugin zusammengefasst haben, können wir das Plugin auf der Seite aufrufen, indem wir es mit dem Formular verknüpfen:

```
html5placeholder/index.html
```

```
$(function(){
  function hasPlaceholderSupport() {
    var i = document.createElement('input');
    return 'placeholder' in i;
  }

  if(!hasPlaceholderSupport()){
    $("#create_account").placeholder();
    // Ende der Ausweichlösung für Platzhalter

    $('input[autofocus=true]').focus();
  };
});
```

Damit haben wir eine ziemlich gute Lösung, die Ihnen Platzhaltertexte als brauchbare Option für Ihre Webanwendungen zur Verfügung stellt, egal welchen Browser Ihre Benutzer verwenden.

6

In-Place-Editing mit `contenteditable`

Wir suchen ständig nach Möglichkeiten, Menschen die Interaktion mit unseren Anwendungen zu erleichtern. Manchmal möchten wir, dass Benutzer auf unserer Website persönliche Informationen eingeben können, ohne zu einem anderen Formular wechseln zu müssen. Bisher lässt sich In-Place-Editing nur implementieren, indem wir die Klicks in Textbereiche überwachen und diese Bereiche dann durch Textfelder ersetzen. Die Felder senden den geänderten Text anschließend über Ajax zurück an den Server. In HTML5 übernimmt das Attribut `contenteditable` den Teil mit der Dateneingabe automatisch. Wir müssen zwar immer noch ein wenig JavaScript schreiben, um die Daten an den Server zurückzuschicken und sie zu speichern, aber wir brauchen keine versteckten Formulare mehr und können uns das Hin- und Herschalten sparen.

In einem aktuellen Projekt von AwesomeCo können Benutzer das Profil ihres Kontos bearbeiten. Es zeigt ihren Namen, die Stadt, die Postleitzahl und die E-Mail-Adresse. Peppen wir nun die Profilseite mit In-Place-Editing so auf, dass wir eine Oberfläche wie in Abbildung 3.3 auf der nächsten Seite erhalten.

Bevor wir damit anfangen, sollten Sie Folgendes wissen: Eine Funktionalität mit JavaScript zu implementieren, ohne zuvor eine rein serverseitige Lösung zu implementieren, widerspricht allem, woran ich bei der Entwicklung barrierefreier Webanwendungen glaube. Wir machen das nur so, weil ich mich auf die Funktionen des Attributs `contenteditable` konzentrieren möchte. *Dies ist kein Code für eine Produktionsumgebung.* Sie sollten immer – und ich meine *immer* – zuerst eine Lösung entwickeln, für die kein JavaScript erforderlich ist. Und erst *dann* sollten Sie die Version schreiben, die nur mit Skripten funktioniert. Und außerdem sollten Sie unbedingt automatisierte Tests für beide Varianten schreiben, damit Sie Fehler zuverlässig aufspüren können, wenn Sie eine Version ändern und die andere nicht.

Das Profilformular

In HTML5 wird das Attribut `contenteditable` für beinahe jedes Element eingeführt. Sie müssen dieses Attribut lediglich zu einem Element hinzufügen, um es in ein editierbares Feld zu verwandeln.

User information

Name	Hugh Mann
City	Anytown
State	OH
Postal Code	92110
Email	boss@awesomecompany.com

Abbildung 3.3: In-Place-Editing leicht gemacht

html5_content_editable/show.html

```
<h1>User information</h1>
<div id="status"></div>
<ul>
  <li>
    <b>Name</b>
    <span id="name" contenteditable="true">Hugh Mann</span>
  </li>
  <li>
    <b>City</b>
    <span id="city" contenteditable="true">Anytown</span>
  </li>
  <li>
    <b>State</b>
    <span id="state" contenteditable="true">OH</span>
  </li>
  <li>
    <b>Postal Code</b>
    <span id="postal_code" contenteditable="true">92110</span>
  </li>
  <li>
    <b>Email</b>
    <span id="email" contenteditable="true">boss@awesomecompany.com</span>
  </li>
</ul>
```

Natürlich können wir das Formular auch mit ein bisschen CSS stylen. Wir verwenden CSS3-Selektoren, um die editierbaren Felder dadurch hervorzuheben, dass sie die Farbe ändern, wenn die Benutzer den Mauszeiger darüber bewegen oder sie auswählen.

html5_content_editable/show.html

```
1 ul{list-style:none;}
2
3 li{clear:both;}
4
5 li>b, li>span{
6   display: block;
7   float: left;
8   width: 100px;
9 }
```

```

10  li>span{
11      width:500px;
12      margin-left: 20px;
13  }
14
15  li>span[contenteditable=true]:hover{
16      background-color: #ffc;
17  }
18
19  li>span[contenteditable=true]:focus{
20      background-color: #ffa;
21      border: 1px shaded #000;
22  }

```

Das war's schon für das Frontend. Die Benutzer können die Daten auf der Seite bequem verändern. Jetzt müssen wir sie nur noch speichern.

Die Daten ablegen

Zwar können die Benutzer die Daten nun ändern, die Änderungen gehen aber verloren, wenn sie die Seite neu laden oder verlassen. Wir brauchen also eine Möglichkeit, diese Änderungen an unser Backend zu übermitteln. Mit jQuery ist das einfach. Wenn Sie schon mal mit Ajax gearbeitet haben, wird das für Sie nichts Neues sein.

html5_content_editable/show.html

```

$(function(){
    var status = $("#status");
    $("span[contenteditable=true]").blur(function(){
        var field = $(this).attr("id");
        var value = $(this).text();
        $.post("http://localhost:4567/users/1",
            field + "=" + value,
            function(data){
                status.text(data);
            }
        );
    });
});

```

Wir fügen für jedes span-Element, dessen contenteditable-Attribut den Wert true hat, einen Event-Listener ein. Anschließend müssen wir die Daten nur noch an unser serverseitiges Skript übermitteln.

Ausweidlösung

Wir haben eine ganze Reihe von Dingen getan, die für einen Teil unserer Zielgruppe nicht funktionieren. Zum einen sind wir auf JavaScript

angewiesen, um die veränderten Ergebnisse auf dem Server zu speichern. Das ist gar nicht gut. Außerdem verwenden wir die Pseudoklasse `focus`, um die Felder hervorzuheben, wenn sie den Fokus erhalten. Das wiederum unterstützen einige Versionen des IE nicht. Kümmern wir uns zuerst um die Funktionalität. Die visuellen Effekte kommen später dran.

Bearbeitungsseiten erstellen

Anstatt uns zu viele Sorgen über die verschiedenen Situationen zu machen, in denen Benutzer unsere Technik nicht nutzen können, bieten wir ihnen einfach die Möglichkeit, eine andere Seite mit einem entsprechenden Formular aufzurufen. Natürlich müssen wir dafür mehr Code schreiben, aber stellen Sie sich die folgenden Szenarien vor:

- Benutzer deaktivieren JavaScript und verwenden den Internet Explorer 7.
- Benutzer verwenden keinen HTML5-kompatiblen Browser.
- Benutzer verwenden den aktuellen Firefox mit HTML5-Unterstützung, haben aber JavaScript deaktiviert, weil sie JavaScript nicht mögen (das kommt immer wieder vor, und zwar häufiger, als Sie denken).

Unterm Strich ist es am sinnvollsten, ein Formular zu erstellen, das einen POST auf dieselbe Aktion durchführt, die sich auch um Ajax kümmert. Wie Sie das tun, bleibt Ihnen überlassen. Aber viele Frameworks bieten die Möglichkeit, anhand der `accept-Header` den Anfragetyp zu ermitteln und festzustellen, ob es sich bei der Anfrage um einen POST oder einen XMLHttpRequest handelt. Auf diese Weise bleibt der serverseitige Code „DRY“.⁵ Wenn der Browser `contenteditable` und JavaScript unterstützt, verstecken wir einfach den Link auf das Formular.

Dementsprechend erstellen wir jetzt eine neue Seite mit dem Namen `edit.html` und schreiben ein standardmäßiges Bearbeitungsformular, das einen POST auf dieselbe Aktion durchführt wie unsere Ajax-Version.

```
html5_content_editable/edit.html
```

```
<!DOCTYPE html>
<html lang="en-US">
```

5 DRY steht für „Don't Repeat Yourself“ (Wiederhol' dich nicht). Diese Devise wurde von Dave Thomas und Andy Hunt in *Der Pragmatische Programmierer* [HT03] geprägt.

```

<head>
  <meta http-equiv="Content-Type" content="text/html; charset=utf-8" />
  <title>Editing Profile</title>
  <link href="style.css" rel="stylesheet" media="screen">
</head>
<body>
  <form action="/users/1" method="post" accept-charset="utf-8">
    <fieldset id="your_information">
      <legend>Your Information</legend>
      <ol>
        <li>
          <label for="name">Your Name</label>
          <input type="text" name="name" value="" id="name">
        </li>
        <li>
          <label for="city">City</label>
          <input type="text" name="city" value="" id="city">
        </li>
        <li>
          <label for="state">State</label>
          <input type="text" name="state" value="" id="state">
        </li>
        <li>
          <label for="postal_code">Postal Code</label>
          <input type="text" name="postal_code" value="" id="postal_code">
        </li>
        <li>
          <label for="email">Email</label>
          <input type="email" name="email" value="" id="email">
        </li>
      </ol>

    </fieldset>

    <p><input type="submit" value="Save"></p>
  </form>

</body>
</html>

```

Fügen Sie anschließend in `show.html` einen Link auf diese Seite ein.

```
html5_content_editable/show.html
```

```

<h1>User information</h1>
<section id="edit_profile_link">
  <p><a href="edit.html">Edit Your Profile</a></p>
</section>
<div id="status"></div>

```

Nachdem wir den Link eingefügt haben, müssen wir nur noch unser Skript ein bisschen anpassen. Wir möchten den Link auf die Bearbeitungsseite nur dann ausblenden und die Ajax-Unterstützung aktivieren, wenn der Browser editierbare Inhalte unterstützt.

```
html5_content_editable/show.html
```

```
if(document.getElementById("edit_profile_link").contentEditable != null){
```

Zusammen mit dieser Prüfung sieht unser Skript folgendermaßen aus:

```
html5_content_editable/show.html
```

```
$(function(){
  if(document.getElementById("edit_profile_link").contentEditable != null){
    $("#edit_profile_link").hide();
    var status = $("#status");
    $("span[contenteditable=true]").blur(function(){
      var field = $(this).attr("id");
      var value = $(this).text();
      $.post("http://localhost:4567/users/1",
        field + "=" + value,
        function(data){
          status.text(data);
        }
      );
    });
  }
});
```

Damit haben unsere Benutzer die Möglichkeit, eine Standardoberfläche oder die schnellere „In-Place“-Variante zu verwenden. Nachdem Sie jetzt wissen, wie diese Oberfläche implementiert wird, müssen Sie aber dennoch daran denken, zuerst die Ausweichlösung zu implementieren. Im Gegensatz zu anderen Lösungen entfällt die Funktionalität hier vollständig, wenn Sie die Ausweichlösung vergessen.

Die Zukunft

Wenn Sie derzeit einen JavaScript-basierten Datepicker in Ihre Website einfügen, müssen Ihre Benutzer zunächst lernen, wie er funktioniert. Wenn Sie schon mal online Flugtickets gekauft und Hotels reserviert haben, kennen Sie bereits die verschiedenen Ansätze, nach denen benutzerdefinierte Formularsteuerelemente auf Websites implementiert werden. Das ist genau wie mit Geldautomaten – die Oberfläche ist oft unterschiedlich genug, um Sie auszubremsten.

Malen Sie sich dagegen aus, wie es wäre, wenn jede Website das HTML5-Datumsfeld verwenden und der Browser die Oberfläche zeichnen würde. Auf jeder Website, die Sie besuchen, würde genau derselbe Datepicker angezeigt werden. Bildschirmlesegeräte oder Screenreader könnten sogar einen Standardmechanismus implementieren, um auch

für sehbehinderte Menschen die Eingabe von Datumswerten einfach zu gestalten. Und stellen Sie sich vor, wie nützlich placeholder-Text und autofocus sein könnten, sobald sie überall eingesetzt werden. Platzhaltertexte könnten Screenreadern dabei helfen, die Formularfelder zu erklären. Und autofocus könnte den Menschen dabei helfen, auch ohne Maus einfacher zu navigieren – was einerseits praktisch für sehbehinderte Menschen ist, aber auch für Benutzer mit motorischen Einschränkungen, die unter Umständen keine Maus bedienen können.

Die Möglichkeit für Entwickler, beliebige Elemente in editierbare Bereiche umzuwandeln, erleichtert das In-Place-Editing und könnte auch den Aufbau von Oberflächen für Content-Management-Systeme verändern.

Im modernen Web spielt Interaktivität die Hauptrolle, und Formulare sind ein wesentlicher Teil dieser Interaktivität. Die mit HTML5 eingeführten Verbesserungen geben uns eine ganze Reihe neuer Tools an die Hand, mit denen wir unseren Benutzern helfen können.

Kapitel 4

Bessere Benutzeroberflächen mit CSS3

Viel zu lange mussten wir als Entwickler auf CSS-Hacks zurückgreifen, damit der Code die gewünschten Effekte bringt. Wir haben mit JavaScript oder serverseitigem Code Tabellen gestreift dargestellt und unsere Formulare scharf- oder weichgezeichnet. Wir mussten unsere Tags mit zusätzlichen `class`-Attributen zumüllen, damit wir die 50 Formularfelder auch auseinanderhalten konnten, die wir stylen wollten.

Damit ist jetzt Schluss! CSS3 bietet einige erstaunliche Selektoren, die einen Teil dieser Arbeit trivial erscheinen lassen. Falls Sie es vergessen haben sollten: Ein Selektor ist ein Muster, mit dem Sie Elemente in HTML-Dokumenten finden können, um Stilregeln darauf anzuwenden. Wir werden nun mit diesen neuen Selektoren eine Tabelle stylen. Anschließend sehen wir uns an, wie wir mit einigen anderen CSS3-Funktionen Stylesheets für den Druck unserer Website verbessern und unseren Inhalt auf mehrere Spalten verteilen können.

Folgende CSS-Funktionen werden wir in diesem Kapitel unter die Lupe nehmen:¹

¹ In den folgenden Beschreibungen wird die Unterstützung durch die verschiedenen Browser in eckigen Klammern mit einem Kurzcode und der mindestens erforderlichen Versionsnummer angegeben. Die verwendeten Codes lauten: *C*: Google Chrome, *F*: Firefox, *IE*: Internet Explorer, *O*: Opera, *S*: Safari, *IOS*: iOS-Geräte mit Mobile Safari und *A*: Android-Browser.

:nth-of-type [p:nth-of-type(2n+1){color:red;}]

Sucht alle n Elemente eines bestimmten Typs.

[C2, F3.5, S3, IE9, O9.5, IOS3, A2]

:first-child [p:first-child{color:blue;}]

Sucht das erste Kindelement.

[C2, F3.5, S3, IE9, O9.5, IOS3, A2]

:nth-child [p:nth-child(2n+1){color:red;}]

Sucht vorwärts nach einem bestimmten Kindelement.

[C2, F3.5, S3, IE9, O9.5, IOS3, A2]

:last-child [p:last-child{color:blue;}]

Sucht das letzte Kindelement. [C2, F3.5, S3, IE9, O9.5, IOS3, A2]

:nth-last-child [p:nth-last-child(2){color:red;}]

Sucht rückwärts nach einem bestimmten Kindelement.

[C2, F3.5, S3, IE9, O9.5, IOS3, A2]

:first-of-type [p:first-of-type{color:blue;}]

Sucht das erste Element des angegebenen Typs.

[C2, F3.5, S3, IE9, O9.5, IOS3, A2]

:last-of-type [p:last-of-type{color:blue;}]

Sucht das letzte Element des angegebenen Typs.

[C2, F3.5, S3, IE9, O9.5, IOS3, A2]

Unterstützung für Spalten [#content{column-count:2;

column-gap:20px; column-rule:1px solid#ddccb5;}]

Unterteilt einen Inhaltsbereich in mehrere Spalten.

[C2, F3.5, S3, O9.5, IOS3, A2]

:after [span.weight:after{ content: "lbs"; color:#bbb;}]

Wird in Verbindung mit content verwendet, um Inhalte nach dem angegebenen Element einzufügen. [C2, F3.5, S3, IE8, O9.5, IOS3, A2]

Media Queries [media="only all and(max-width:480)"]

Damit können Stile aufgrund von Geräteeinstellungen angewendet werden. [C3, F3.5, S4, IE9, O10.1, IOS3, A2]

7

Tabellen mit Pseudoklassen stylen

Eine *Pseudoklasse* ist eine Möglichkeit, in CSS Elemente anhand von Informationen auszuwählen, die entweder außerhalb des Dokuments liegen oder nicht mit herkömmlichen Selektoren ausgedrückt werden können. Wahrscheinlich haben Sie bereits Pseudoklassen wie zum Beispiel `:hover` verwendet, um die Farbe eines Links zu ändern, wenn der Benutzer den Mauszeiger darüber bewegt. CSS3 bietet mehrere neue Pseudoklassen, die das Auffinden von Elementen wesentlich vereinfachen.

Verbesserte Abrechnungen

AwesomeCo verwendet für ausgelieferte Produkte das Abrechnungssystem eines Drittherstellers. Eine der größten Einnahmequellen von AwesomeCo sind Merchandise-Artikel wie Kugelschreiber, Tassen, T-Shirts und alles andere, auf das man ein Logo drucken kann. Sie wurden gebeten, die Abrechnungen lesbarer zu gestalten. Im Moment generieren die Entwickler eine standardmäßige HTML-Tabelle, die wie in Abbildung 4.1 auf der folgenden Seite aussieht.

Es ist eine ziemlich gewöhnliche Abrechnung mit Preisen, Mengenangaben, Zeilensummen, einer Zwischensumme, Versandkosten sowie einer Gesamtsumme für die Bestellung. Die Rechnung wäre besser lesbar, wenn jede zweite Zeile in einer anderen Farbe dargestellt würde. Außerdem wäre es hilfreich, wenn die Gesamtsumme in einer anderen Farbe angezeigt würde, damit sie mehr heraussticht.

So sieht der Code für die Tabelle aus. Kopieren Sie ihn in eine eigene Datei, damit Sie damit arbeiten können.

```
css3advancedselectors/table.html
```

```
<table>
  <tr>
    <th>Item</th>
    <th>Price</th>
    <th>Quantity</th>
    <th>Total</th>
  </tr>
  <tr>
    <td>Coffee mug</td>
    <td>$10.00</td>
```

```

        <td>5</td>
        <td>$50.00</td>
    </tr>
    <tr>
        <td>Polo shirt</td>
        <td>$20.00</td>
        <td>5</td>
        <td>$100.00</td>
    </tr>
    <tr>
        <td>Red stapler</td>
        <td>$9.00</td>
        <td>4</td>
        <td>$36.00</td>
    </tr>
    <tr>
        <td colspan="3">Subtotal</td>
        <td>$186.00</td>
    </tr>
    <tr>
        <td colspan="3">Shipping</td>
        <td>$12.00</td>
    </tr>
    <tr>
        <td colspan="3">Total Due</td>
        <td>$198.00</td>
    </tr>
</table>

```

Item	Price	Quantity	Total
Coffee mug	\$10.00	5	\$50.00
Polo shirt	\$20.00	5	\$100.00
Red stapler	\$9.00	4	\$36.00
Subtotal			\$186.00
Shipping			\$12.00
Total Due			\$198.00

Abbildung 4.1: Für die aktuelle Rechnung wird eine ungestylte HTML-Tabelle verwendet.

Als Erstes beseitigen wir den schrecklichen Tabellenrahmen:

```
css3advancedselectors/table.css
```

```

table{
    width: 600px;
    border-collapse: collapse;
}

```

```
th, td{
  border: none;
}
```

Außerdem stylen wir die Überschrift mit weißem Text auf schwarzem Hintergrund:

```
css3advancedselectors/table.css
```

```
th{
  background-color:
    #000; color: #fff;
}
```

Wenn wir diese Stilregeln anwenden, sieht die Tabelle folgendermaßen aus:

Item	Price	Quantity	Total
Coffee mug	\$10.00	5	\$50.00
Polo shirt	\$20.00	5	\$100.00
Red stapler	\$9.00	4	\$36.00
Subtotal			\$186.00
Shipping			\$12.00
Total Due			\$198.00

Nachdem wir die Rahmen und Abstände der Tabelle ein bisschen aufgeräumt haben, können wir nun mit Pseudoklassen die einzelnen Zeilen und Spalten stylen. Beginnen wir damit, die Tabelle gestreift darzustellen.

Gestreifte Zeilen mit :nth-of-type

Wir alle kennen Tabellen mit „Zebrastreifen“ – jenen horizontalen Linien, an denen sich die Benutzer entlanghangeln können. Diese Art der Gestaltung nehmen wir am besten in CSS vor, in der Präsentationsschicht. Üblicherweise mussten wir dafür zusätzliche Klassennamen für unsere Tabellenzeilen einfügen, wie etwa „gerade“ und „ungerade“. Wir möchten aber das Markup unserer Tabelle nicht auf diese Weise beschmutzen, weil die HTML5-Spezifikation uns ermuntert, spezielle Klassennamen für die Präsentation zu vermeiden. Mit einigen neuen Selektoren bekommen wir, was wir möchten, auch ohne unser Markup zu ändern. Und wir können so die Darstellung wirklich vom Inhalt trennen.

Der Selektor `nth-of-type` sucht jedes *n-te* Element eines bestimmten Typs anhand einer Formel oder von Schlüsselwörtern. Wir sehen uns die Variante mit der Formel bald genauer an. Zunächst konzentrieren wir uns aber auf die Schlüsselwörter, da diese schneller zu verstehen sind..

Wir möchten jede zweite Zeile der Tabelle mit einer anderen Farbe darstellen. Die einfachste Möglichkeit besteht darin, jede Tabellenzeile mit einer ungeraden Nummer zu suchen und mit einer Hintergrundfarbe zu versehen. Anschließend machen wir dasselbe mit den Zeilen mit geraden Nummern. CSS3 bietet die Schlüsselwörter `even` und `odd`, mit denen genau das möglich ist.

```
css3advancedselectors/table.css
```

```
tr:nth-of-type(even){
    background-color: #F3F3F3;
}
tr:nth-of-type(odd) {
    background-color:#ddd;
}
```

Dieser Selektor sagt also: „Suche jede Tabellenzeile mit einer geraden Nummer (`even`), und färbe sie ein. Suche anschließend alle Tabellenzeilen mit einer ungeraden Nummer (`odd`), und färbe diese ein.“ Dadurch bekommen wir unsere Zebrastreifen, ohne auf Skripten oder zusätzliche Klassennamen für die jeweiligen Zeilen zurückzugreifen.

Nach Anwendung der Stilregeln sieht unsere Tabelle so aus:

Item	Price	Quantity	Total
Coffee mug	\$10.00	5	\$50.00
Polo shirt	\$20.00	5	\$100.00
Red stapler	\$9.00	4	\$36.00
Subtotal			\$186.00
Shipping			\$12.00
Total Due			\$198.00

Jetzt beschäftigen wir uns mit der Ausrichtung der Spalten in der Tabelle.

Spaltentexte ausrichten mit `:nth-child`

Standardmäßig werden alle Spalten unserer Abrechnungstabelle linksbündig dargestellt. Wir möchten aber bis auf die erste Spalte alle Spal-

ten rechtsbündig darstellen. Auf diese Weise sind die Spalten für die Preise und Mengen rechtsbündig und leichter lesbar. Dafür können wir `nth-child` verwenden. Aber zunächst müssen wir verstehen, wie das funktioniert.

Der Selektor `nth-child` sucht nach Kindelementen eines Elements und kann genauso wie `nth-of-type` mit Schlüsselwörtern oder einer Formel verwendet werden.

Die Formel lautet $a+b$, wobei b der Versatz ist und a ein Vielfaches davon. Diese Beschreibung ist ohne Zusammenhang nicht gerade hilfreich, daher sehen wir uns das mal für unsere Tabelle an.

Wenn wir alle Tabellenzeilen auswählen möchten, können wir diesen Selektor verwenden:

```
table tr:nth-child(n)
```

Wir verwenden kein Vielfaches und auch keinen Versatz.

Wenn wir alle Zeilen der Tabelle außer der ersten Zeile mit den Spaltenüberschriften auswählen möchten, verwenden wir den folgenden Selektor mit einem Versatz:

```
table tr:nth-child(n+2)
```

Wenn wir jede zweite Zeile unserer Tabelle auswählen möchten, verwenden wir ein Vielfaches: $2n$.

```
table tr:nth-child(2n)
```

Wenn Sie jede dritte Reihe auswählen möchten, verwenden Sie $3n$.

Sie können den Versatz auch verwenden, um weiter unten in der Tabelle zu beginnen. Der folgende Selektor wählt ab der vierten Zeile jede zweite Zeile aus:

```
table tr:nth-child(2n+4)
```

Entsprechend können wir mit der folgenden Regel jede Zeile *außer* der ersten auswählen:

```
css3advancedselectors/table.css
```

```
td:nth-child(n+2){
  text-align: right;
}
```

Damit nimmt unsere Tabelle allmählich Form an:

Item	Price	Quantity	Total
Coffee mug	\$10.00	5	\$50.00
Polo shirt	\$20.00	5	\$100.00
Red stapler	\$9.00	4	\$36.00
Subtotal			\$186.00
Shipping			\$12.00
Total Due			\$198.00

Jetzt stylen wir die letzte Zeile der Tabelle.

Die letzte Zeile fett darstellen mit :last-child

Die Abrechnung sieht schon ziemlich gut aus, aber einer der Manager möchte, dass die unterste Zeile der Tabelle fetter dargestellt wird, damit sie besser ins Auge sticht. Auch dafür können wir die Regel `last-child` verwenden, die das letzte Kind einer Gruppe erfasst.

Eine von Webentwicklern häufig verwendete Technik besteht darin, Absätze mit einem unteren Rand zu versehen, damit sie gleichmäßige Abstände auf der Seite haben. Dies kann manchmal zu einem unerwünschten unteren Rand am Ende einer Gruppe führen. Wenn sich die Absätze beispielsweise innerhalb einer Seitenleiste oder einer Callout-Box befinden, möchten wir den unteren Rand des letzten Absatzes entfernen, damit kein überflüssiger Leerraum zwischen dem unteren Rand des letzten Absatzes und dem Rahmen der Box entsteht. Der Selektor `last-child` ist das perfekte Werkzeug, um den unteren Rand vom letzten Absatz zu entfernen.

```
p{ margin-bottom: 20px }
#sidebar p:last-child{ margin-bottom: 0; }
```

Dieselbe Technik wenden wir an, um die Inhalte der letzten Zeile fett darzustellen:

```
css3advancedselectors/table.css
```

```
tr:last-child{
  font-weight: bolder;
}
```

Dasselbe machen wir nun mit der letzten Spalte der Tabelle. Dadurch werden die Summen der jeweiligen Zeile ebenfalls hervorgehoben.

css3advancedselectors/table.css

```
td:last-child{
  font-weight: bolder;
}
```

Zu guter Letzt wählen wir für die Gesamtsumme eine größere Schriftgröße, indem wir `last-child` zusammen mit Nachkommenselektoren verwenden. Wir suchen die letzte Spalte der letzten Zeile und stylen sie so:

css3advancedselectors/table.css

```
tr:last-child td:last-child{
  font-size: 24px;
}
```

Item	Price	Quantity	Total
Coffee mug	\$10.00	5	\$50.00
Polo shirt	\$20.00	5	\$100.00
Red stapler	\$9.00	4	\$36.00
Subtotal			\$186.00
Shipping			\$12.00
Total Due			\$198.00

Wir sind beinahe fertig, aber wir müssen noch ein paar Dinge mit den letzten Zeilen der Tabelle machen.

Rückwärts zählen mit `:nth-last-child`

Wir möchten die Tabellenzeile für den Versand hervorheben, wenn es einen Rabatt für den Versand gibt. Wir verwenden `nth-last-child`, um diese Zeile schnell zu finden. Im Abschnitt *Spaltentexte ausrichten mit `:nth-child`* auf Seite 78 haben Sie bereits gesehen, wie Sie mit `nth-child` und der Formel `an+b` bestimmte Kindelemente auswählen können. Der Selektor `nth-last-child` funktioniert genauso, außer dass er mit dem letzten Kind beginnt und rückwärts durch die Kindelemente zählt. Auf diese Weise ist es einfach, das vorletzte Element einer Gruppe zu erfassen. Und wie sich herausstellt, müssen wir genau das mit unserer Rechnungstabelle tun..

Unsere Versandzeile finden wir also mit diesem Code:

css3advancedselectors/table.css

```
tr:nth-last-child(2){
  color: green;
}
```

Diesmal geben wir ein bestimmtes Kind an, und zwar das vorletzte.

Es gibt noch eine letzte Sache, die wir mit dieser Tabelle tun sollten. Zuvor haben wir alle Spalten außer der ersten rechtsbündig formatiert. Das sieht zwar für die Tabellenzeilen mit den Positionsbeschreibungen und Preisen wunderbar aus. In den letzten drei Tabellenzeilen wirkt die erste Spalte allerdings ein bisschen seltsam, wenn sie linksbündig ausgerichtet wird. Formatieren wir also in den letzten drei Zeilen die erste Spalte ebenfalls rechtsbündig. Das erreichen wir mit `nth-last-child` in Verbindung mit einem negativen Wert für `n` und einem positiven Wert für `a` in unserer Formel:

```
css3advancedselectors/table.css
```

```
tr:nth-last-child(-n+3) td{
    text-align: right;
}
```

Sie können sich das als Bereichsselektor vorstellen ... wir verwenden einen Versatz von 3. Nachdem es sich um `nth-last-child` handelt, erfassen wir so alle Elemente vor dem Versatz. Würden wir stattdessen `nth-child` verwenden, würde diese Formel jede Zeile ab dem Versatz auswählen.

Unsere frisch gestaltete Tabelle sieht in Abbildung 4.2 schon viel besser aus. Und wir mussten dafür das zugrunde liegende Markup kein bisschen ändern. Viele der hier verwendeten Selektoren stehen Benutzern des Internet Explorer noch nicht zur Verfügung, also brauchen wir einen Workaround.

Ausweichlösung

Die aktuellen Versionen von Opera, Firefox, Safari und Chrome verstehen diese Selektoren, aber der Internet Explorer in den Versionen 8.0 und davor ignoriert sie völlig. Sie brauchen eine gute Ausweichlösung und müssen eine Entscheidung treffen.

Item	Price	Quantity	Total
Coffee mug	\$10.00	5	\$50.00
Polo shirt	\$20.00	5	\$100.00
Red stapler	\$9.00	4	\$36.00
Subtotal			\$186.00
Shipping			\$12.00
Total Due			\$198.00

Abbildung 4.2: Unsere gestylte Tabelle – mit Streifen und Formatierung, ausschließlich in CSS3 umgesetzt

HTML-Code ändern

Die naheliegendste Lösung, die immer funktioniert, besteht darin, den zugrunde liegenden Code zu ändern. Sie könnten allen Zellen der Tabelle Klassen zuweisen und auf jede Klasse einfache CSS-Regeln anwenden. Das wäre die schlechteste Wahl, weil dadurch Darstellung und Inhalt gemischt werden, und wir ja gerade CSS3 verwenden, um das zu vermeiden. Eines Tages würden wir das überflüssige Markup nicht mehr brauchen, und es wäre lästig, es wieder zu beseitigen.

JavaScript verwenden

Die jQuery-Bibliothek versteht bereits die meisten von uns verwendeten CSS3-Selektoren, also könnten wir schnell eine Methode schreiben, um die Tabelle auf diese Weise zu stylen. Es gibt aber eine einfachere Lösung.

Keith Clark hat eine tolle kleine Bibliothek mit dem Namen IE-css3² geschrieben, die den Internet Explorer um die Unterstützung von CSS3-Selektoren erweitert. Wir müssen also lediglich einige Skripten zu unserer Seite hinzufügen.

Die Bibliothek IE-CSS3 kann auf jQuery, Prototype oder verschiedene andere Bibliotheken zurückgreifen. Aber ich bevorzuge die DOM Assistant-Bibliothek³, weil sie die beste Unterstützung für die hier verwendeten Pseudoklassen bietet.

Laden Sie diese beiden Bibliotheken herunter, und binden Sie sie in Ihr Dokument ein. Da wir sie nur für den IE benötigen, können Sie sie in einen bedingten Kommentar schreiben, sodass sie nur von IE-Benutzern verwendet werden.

Item	Price	Quantity	Total
Coffee mug	\$10.00	5	\$50.00
Polo shirt	\$20.00	5	\$100.00
Red stapler	\$9.00	4	\$36.00
Subtotal			\$186.00
Shipping			\$12.00
Total Due			\$198.00

Abbildung 4.3: Unsere Tabelle sieht im Internet Explorer toll aus.

2 <http://www.keithclark.co.uk/labs/ie-css3/>

3 <http://www.domassistant.com/>

css3advancedselectors/table.html

```
<!--[if (gte IE 5.5)&(lte IE 8)]>
  <script type="text/javascript"
    src="js/DOMAssistantCompressed-2.8.js"></script>
  <script type="text/javascript"
    src="js/ie-css3.js"></script>
<![endif]-->
```

Wie Sie in Abbildung 4.3 feststellen werden, sieht auch im Internet Explorer alles wunderbar aus, wenn Sie diese Skripten in die Seite einfügen.

Obwohl die Benutzer dafür JavaScript aktiviert haben müssen, dient der Tabellenstil vor allem dazu, den Inhalt leichter lesbar zu machen. Fehlt das Styling, kann trotzdem jeder die Rechnung lesen.

Das Stylen von Elementen ist mit CSS3 wesentlich einfacher, insbesondere wenn wir keine Möglichkeit haben, das jeweilige HTML zu ändern. Wenn Sie Benutzeroberflächen stylen, sollten Sie die semantische Hierarchie und diese neuen Selektoren verwenden, bevor Sie zusätzliches Markup schreiben. Dadurch ist Ihr Code viel einfacher zu pflegen.

8

Links ausdrucken mit :after und content

Mit CSS können Sie nicht nur vorhandene Elemente stylen, sondern auch Inhalte in ein Dokument einfügen. Es gibt einige Fälle, in denen die Erzeugung von Inhalten mit CSS sinnvoll ist. So wäre es zum Beispiel naheliegend, beim Ausdruck einer Seite die URLs von Hyperlinks neben dem jeweiligen Text mit auszugeben. Wenn Sie ein Dokument auf dem Bildschirm ansehen, können Sie einfach den Mauszeiger über den Link bewegen und sehen in der Statusleiste, wohin der Link führt. Auf dem Ausdruck einer Seite haben Sie dagegen keine Ahnung, wohin die Links verweisen.

AwesomeCo arbeitet an einer neuen Seite für Formulare und Richtlinien. Ein Mitglied des Gremiums für die Neugestaltung besteht darauf, bei jedem Treffen ein Exemplar der Website auszudrucken. Dieser Kollege möchte genau wissen, wohin die Links auf der Seite führen, damit er feststellen kann, ob sie verändert werden müssen. Mit ein bisschen CSS können wir diese Funktionalität in IE 8, Firefox, Safari und Chrome hinbekommen. Und mit proprietärem JavaScript funktioniert es auch im IE 6 und 7.

Die Seite selbst enthält derzeit nichts außer einer Liste von Links. Bei Gelegenheit wird sie in eine entsprechende Vorlage eingesetzt.

```
css3_print_links/index.html
```

```
<ul>
  <li>
    <a href="travel/index.html">Travel Authorization Form</a>
  </li>
  <li>
    <a href="travel/expenses.html">Travel Reimbursement Form</a>
  </li>
  <li>
    <a href="travel/guidelines.html">Travel Guidelines</a>
  </li>
</ul>

</body>
```

Auf einem Ausdruck dieser Seite können Sie bisher nicht erkennen, wohin diese Links führen. Das werden wir ändern.

Das CSS

Wenn wir Stylesheets in eine Seite einbinden, können wir den Medientyp angeben, für den die Stilregeln gelten sollen. Meistens verwenden wir den Typ `screen`. Wir können aber auch den Typ `print` angeben, um ein Stylesheet zu definieren, das nur für den Ausdruck der Seite geladen wird (oder wenn der Benutzer die Druckvorschau aufruft).

```
css3_print_links/index.html
```

```
<link rel="stylesheet" href="print.css" type="text/css" media="print">
```

Anschließend erstellen wir das Stylesheet `print.css` mit dieser einfachen Regel:

```
css3_print_links/print.css
```

```
a:after {
  content: " (" attr(href) ") ";
}
```

Dadurch wird hinter den Text jedes Links in Klammern der Wert des Attributs `href` hinzugefügt. Wenn Sie die Seite in einem modernen Browser ausdrucken, sieht das so aus:

Forms and Policies

- [Travel Authorization Form \(travel/index.html\)](#)
- [Travel Reimbursement Form \(travel/expenses.html\)](#)
- [Travel Guidelines \(travel/guidelines.html\)](#)

Wenn Sie das Ganze in Aktion sehen möchten, ohne Papier zu verbrauchen, können Sie die Druckvorschau des Browsers verwenden, die ebenfalls dieses Stylesheet aufruft.

Damit haben wir alles im Griff, außer den Internet Explorer 6 und 7. Sollen wir uns darum als Nächstes kümmern?

Ausweichlösung

Der Internet Explorer verfügt über einige JavaScript-Events, von denen ich mir wünsche, dass alle Browser sie übernehmen: `onbeforeprint` und `onafterprint`. Mithilfe dieser Events können wir den Text des Hyperlinks ändern, wenn der Ausdruck angestoßen wird, und anschließend die Änderung wieder rückgängig machen, wenn der

Druck abgeschlossen ist. Unsere Benutzer werden den Unterschied nicht bemerken.⁴

Wir müssen lediglich eine Datei mit dem Namen `print.js` anlegen und diesen Code einfügen:

css3_print_links/print.js

```

1  $(function() {
2      if (window.onbeforeprint !== undefined) {
3          window.onbeforeprint = ShowLinks;
4          window.onafterprint = HideLinks;
5      }
6  });
7
8  function ShowLinks() {
9      $("a").each(function() {
10         $(this).data("linkText",$(this).text());
11         $(this).append(" (" + $(this).attr("href") + ")");
12     });
13 }
14
15 function HideLinks() {
16     $("a").each(function() {
17         $(this).text($(this).data("linkText"));
18     });
19 }

```

Anschließend binden wir die Datei in unsere Seite ein. Wir brauchen diese Lösung lediglich für den IE 6 und 7, deshalb können wir einen bedingten Kommentar verwenden. Der Code setzt jQuery voraus, also müssen wir unbedingt auch die jQuery-Bibliothek einbinden.

css3_print_links/index.html

```

<script
  charset="utf-8"
  src='http://ajax.googleapis.com/ajax/libs/jquery/1.4.2/jquery.min.js'
  type='text/javascript'>
</script>
<!--[if lte IE 7]>
<script type="text/javascript" src="print.js"></script>
<![endif]-->
</head>
<body>
  <h1>Forms and Policies</h1>

  <ul>
    <li>
      <a href="travel/index.html">Travel Authorization Form</a>

```

4 Eine gute Erklärung dieser Technik finden Sie unter <http://beckelman.net/post/2009/02/16/Use-jQuery-toShow-a-Link28099s-Address-After-its-Text-When-Printing-In-IE6-and-IE7.aspx>.

```
</li>
<li>
  <a href="travel/expenses.html">Travel Reimbursement Form</a>
</li>
<li>
  <a href="travel/guidelines.html">Travel Guidelines</a>
</li>
</ul>
```

Sobald das JavaScript eingebunden ist, werden die URLs von Links auf allen Browsern ausgedruckt. Sie können dieses Stylesheet für den Druck als Grundlage für ein umfassenderes verwenden und zum Beispiel dieses Verhalten nur auf bestimmte Links anwenden.

Im Printbereich werden seit Langem Spalten verwendet – und von vielen Webdesignern neidisch beäugt. Schmale Spalten erleichtern das Lesen, und seitdem Bildschirme immer breiter werden, suchen Entwickler nach Möglichkeiten, die Spalten auf eine angenehme Breite zu begrenzen. Schließlich möchte niemand Textzeilen quer über den ganzen Monitor lesen – genauso wie niemand Textzeilen über die gesamte Breite einer Zeitung lesen möchte. In den vergangenen zehn Jahren gab es einige ziemlich smarte Lösungen. Aber keine ist so einfach wie die Methode der CSS3-Spezifikation.

Spalten spalten

AwesomeCo veröffentlicht jeden Monat einen Newsletter für seine Mitarbeiter. Das Unternehmen verwendet zufällig ein beliebtes webbasiertes E-Mail-System. E-Mail-basierte Newsletter sehen bei Weitem nicht so gut aus und sind schwer zu pflegen. AwesomeCo hat daher beschlossen, den Newsletter auf die Intranetseite zu stellen und den Mitarbeitern nur eine E-Mail mit einem Link zu schicken, damit Sie den Newsletter im Browser ansehen können. Ein Gerüst dieses Newsletters sehen Sie in Abbildung 4.4 auf der nächsten Seite.

Die neue Kommunikationdirektorin, die Erfahrung im Printbereich hat, möchte, dass der Newsletter mehr wie ein echter Newsletter aussieht – mit zwei Spalten statt einer.

Falls Sie jemals versucht haben, Text mit `divs` und `floats` in mehrere Spalten aufzuteilen, wissen Sie, wie schwer das sein kann. Die erste große Hürde besteht darin, dass Sie den Text von Hand aufteilen müssen. In DTP-Software wie zum Beispiel InDesign können Sie die Textfelder so miteinander verknüpfen, dass der Text vom ersten Feld automatisch in das nächste fließt, wenn das erste Textfeld vollständig mit Text gefüllt ist. Exakt so etwas gibt es für das Web noch nicht. Aber wir haben etwas, das wirklich sehr gut und einfach funktioniert. Wir können den Inhalt eines Elements auf mehrere gleich breite Spalten aufteilen.

Wir beginnen mit dem Markup für den Newsletter. Das HTML ist ziemlich einfach. Da sich der Inhalt ohnehin ändern wird, sobald wir es geschrieben haben, verwenden wir einfach Blindtext. Falls Sie sich fragen, warum wir nicht die neuen HTML5-Markup-Elemente wie zum

Beispiel section für den Newsletter verwenden: Das liegt daran, dass unsere Ausweidlösung mit diesen Elementen nicht mit dem Internet Explorer kompatibel ist.



Abbildung 4.4: Der einspaltige Newsletter ist schwer zu lesen, weil er sehr breit ist.

css3columns/condensed_newsletter.html

```
<body>
  <div id="header">
    <h1>AwesomeCo Newsletter</h1>
    <p>Volume 3, Issue 12</p>
  </div>
  <div id="newsletter">
    <div id="director_news">
      <div>
        <h2>News From The Director</h2>
      </div>
      <div>
        <p>
          Lorem ipsum dolor...
        </p>
        <div class="callout">
          <h4>Being Awesome</h4>
          <p>
            "Lorem ipsum dolor sit amet..."
          </p>
        </div>
        <p>
          Duis aute irure...
        </p>
      </div>
    </div>
  </div>
</div>
```

```

<div id="awesome_bits">
  <div>
    <h2>Quick Bits of Awesome</h2>
  </div>
  <div>
    <p>
      Lorem ipsum...
    </p>
    <p>
      Duis aute irure...
    </p>
  </div>
</div>

<div id="birthdays">
  <div>
    <h2>Birthdays</h2>
  </div>
  <div>
    <p>
      Lorem ipsum dolor...
    </p>
    <p>
      Duis aute irure...
    </p>
  </div>
</div>

</div>
<div id="footer">
  <h6>Send newsworthy things to
    <a href="mailto:news@aweseomco.com">news@awesomeco.com</a>
  </h6>
</div>
</body>

```

Um den Newsletter auf zwei Spalten aufzuteilen, müssen wir lediglich Folgendes in unser Stylesheet einfügen:

```
css3columns/newsletter.html
```

```

#newsletter{
  -moz-column-count: 2;
  -webkit-column-count: 2;
  -moz-column-gap: 20px;
  -webkit-column-gap: 20px;
  -moz-column-rule: 1px solid #ddccb5;
  -webkit-column-rule: 1px solid #ddccb5;
}

```

Wie Sie in Abbildung 4.5 erkennen, sieht unser Newsletter schon viel besser aus. Wenn wir zusätzlichen Inhalt einfügen, verteilt der Browser den Inhalt automatisch gleichmäßig auf die Spalten. Beachten Sie auch, wie die gefloateten Elemente in die Spalten fließen, in die sie gehören.



Abbildung 4.5: Unser neuer zweispaltiger Newsletter



Joe fragt ...

Kann ich für jede Spalte eine andere Breite angeben?

Nein, alle Spalten müssen dieselbe Breite haben. Ich war zunächst auch etwas überrascht, also habe ich die Spezifikation gründlich überprüft. Als ich dieses Buch geschrieben habe, war keine Möglichkeit vorgesehen, verschiedene Spaltenbreiten anzugeben.

Wenn man aber darüber nachdenkt, wie Spalten herkömmlicherweise verwendet werden, ist das auch sinnvoll. Spalten sind genauso wenig wie Tabellen als Trick gedacht, um schnell eine Seitenleiste zu basteln. Spalten sollen das Lesen von langen Textpassagen erleichtern, und gleich breite Spalten sind nun mal perfekt dafür.

Ausweidlösung

CSS3-Spalten funktionieren im Internet Explorer der Version 8 und davor leider nicht. Also verwenden wir den jQuery-Plugin Columnizer⁵ als Ausweidlösung. Mit Columnizer können wir unseren Inhalt mit dem folgenden einfachen Code gleichmäßig aufteilen:

⁵ <http://welcome.totheinter.net/columnizer-jquery-plugin/>

css3columns/newsletter.html

```
$("#newsletter").columnize({ columns: 2 });
```

Menschen ohne JavaScript müssen mit einer Textspalte auskommen, können den Inhalt aber trotzdem lesen, weil wir ihn linear aufgebaut haben. Auch sie sind also versorgt. Allerdings können wir mit JavaScript die Unterstützung bestimmter Elemente durch den Browser ermitteln: Wenn wir eine existierende CSS-Eigenschaft abfragen, erhalten wir einen Leerstring. Erhalten wir dagegen einen Nullwert, ist die Eigenschaft nicht verfügbar.

css3columns/newsletter.html

```
<script
  charset="utf-8"
  src='http://ajax.googleapis.com/ajax/libs/jquery/1.4.2/jquery.min.js'
  type='text/javascript'>
</script>

<script
  charset="utf-8"
  src="javascripts/autocolumn.js"
  type='text/javascript'>
</script>

<script type="text/javascript">
  function hasColumnSupport(){
    var element = document.documentElement;
    var style = element.style;
    if (style){
      return typeof style.columnCount == "string" ||
        typeof style.MozColumnCount == "string" ||
        typeof style.WebkitColumnCount == "string" ||
        typeof style.KhtmlColumnCount == "string";
    }
    return null;
  }

  $(function(){
    if(!hasColumnSupport()){
      $("#newsletter").columnize({ columns: 2 });
    }
  });
</script>
```

AwesomeCo Newsletter

Volume 3, Issue 12

News From The Director

Lorem ipsum dolor sit amet, consectetur adipiscing elit, sed do eiusmod tempor incididunt ut labore et dolore magna aliqua. Ut enim ad minim veniam, quis nostrud exercitation ullamco laboris nisi ut aliquip ex ea commodo consequat.

Duis aute irure dolor in reprehenderit in voluptate velit esse cillum dolore eu fugiat nulla pariatur. Excepteur sint occaecat cupidatat non proident, sunt in culpa qui officia deserunt mollit anim id est laborum.

Quick Bits of Awesome

Lorem ipsum dolor sit amet, consectetur adipiscing elit, sed do eiusmod tempor incididunt ut labore et dolore magna aliqua. Ut

Being Awesome

"Lorem ipsum dolor sit amet, consectetur adipiscing elit, sed do eiusmod tempor incididunt ut labore et dolore magna aliqua. Ut enim ad minim veniam."

enim ad minim veniam, quis nostrud exercitation ullamco laboris nisi ut aliquip ex ea commodo consequat.

Duis aute irure dolor in reprehenderit in voluptate velit esse cillum dolore eu fugiat nulla pariatur. Excepteur sint occaecat cupidatat non proident, sunt in culpa qui officia deserunt mollit anim id est laborum.

Birthdays

Lorem ipsum dolor sit amet, consectetur adipiscing elit, sed do eiusmod tempor incididunt ut labore et dolore magna aliqua. Ut enim ad minim veniam, quis nostrud exercitation ullamco laboris nisi ut aliquip ex ea commodo consequat.

Duis aute irure dolor in reprehenderit in voluptate velit esse cillum dolore eu fugiat nulla pariatur. Excepteur sint occaecat cupidatat non proident, sunt in culpa qui officia deserunt mollit anim id est laborum.

Send newsworthy things to news@awesomeco.com.

Abbildung 4.6: Unsere Internet Explorer-Version funktioniert, braucht aber einige Anpassungen.

Wir prüfen also zunächst, ob Spalten unterstützt werden. Falls nicht, wenden wir unser Plugin an.

Laden Sie die Seite im Internet Explorer neu, und der Newsletter wird zweiseitig angezeigt. Das Ergebnis ist zwar nicht perfekt (siehe Abbildung 4.6). Aber mit ein bisschen CSS oder JavaScript können Sie die Elemente zurechtbiegen, die noch nicht optimal aussehen. Diese Übung überlasse ich Ihnen. Nutzen Sie bedingte Kommentare wie im Abschnitt *JavaScript verwenden* auf Seite 83, um sich bei Bedarf bestimmte Versionen des Internet Explorer vorzuknöpfen.

Die Aufteilung von Inhalten auf mehrere Spalten kann diese besser lesbar machen. Sollte Ihre Seite aber länger sein, finden es Ihre Benutzer unter Umständen nervig, wieder ganz an den Anfang scrollen zu müssen. Setzen Sie diese Funktion daher mit Vorsicht ein.

10

Benutzeroberflächen für mobile Geräte mit Media Queries

Wir können schon seit einer ganzen Weile medienspezifische Stylesheets definieren. Aber wie Sie bei unserem Druck-Stylesheet im Abschnitt *Links ausdrucken mit :after und content* auf Seite 85 gesehen haben, waren wir bisher auf den Ausgabebetyp beschränkt. Mit den Media Queries⁶ von CSS3 können wir die Darstellung einer Seite sogar an die Bildschirmgröße unserer Benutzer anpassen. Webentwickler haben schon seit Jahren mit JavaScript die Bildschirmgröße ihrer Benutzer ermittelt. Aber jetzt können wir damit anfangen, das ausschließlich mit Stylesheets zu tun. Mit Media Queries können wir Folgendes ermitteln:

- Auflösung
- Ausrichtung (Hoch- oder Querformat)
- Breite und Höhe des Geräts
- Breite und Höhe des Browserfensters

Daher machen es uns Media Queries recht einfach, spezielle Stylesheets für mobile Benutzer zu entwickeln. Wie sich herausstellt, haben die Führungskräfte von AwesomeCo gerade alle ihre Blackberrys gegen brandneue iPhones ausgetauscht. Der Marketingdirektor möchte unbedingt eine iPhone-taugliche Version der Blogvorlage sehen, die wir im Abschnitt *Einen Blog mit semantischem Markup neu definieren* auf Seite 26 erstellt haben. Das bekommen wir recht schnell hin.

Unser derzeitiger Blog hat ein zweiseitiges Layout mit einem Hauptinhaltsbereich und einer Seitenleiste. Am einfachsten können wir das auf einem iPhone lesbarer darstellen, indem wir die gefloateten Elemente entfernen, damit die Seitenleiste unterhalb des Hauptinhalts erscheint. Auf diese Weise müssen die Leser nicht zur Seite scrollen.

6 <http://www.w3.org/TR/css3-mediaqueries/>



Joe fragt ...

Was ist mit dem Medientyp „Handheld“?

Der Medientyp Handheld wurde entwickelt, damit wir Stylesheets gezielt für mobile Geräte erstellen können. Aber die meisten mobilen Geräte möchten uns das „echte Internet“ zeigen: Sie ignorieren den Medientyp und verwenden einfach das Stylesheet für den Medientyp screen.

Damit das funktioniert, fügen wir den folgenden Code am Ende des Stylesheets für den Blog ein:

css3mediaquery/style.css

```
@media only screen and (max-device-width: 480px) {
  body{
    width:460px;
  }

  section#sidebar, section#posts{
    float: none;
    width: 100%;
  }
}
```

Sie können sich den Code, den wir in die geschweiften Klammern der Media Query geschrieben haben, wie ein eigenes Stylesheet vorstellen, das nur aufgerufen wird, wenn die Bedingungen der Query erfüllt sind. In diesem Fall ändern wir die Größe des Body der Seite und machen aus dem zweispaltigen Layout ein einspaltiges.

Wir können Media Queries auch beim Einbinden des Stylesheets verwenden, dann können wir unser Stylesheet für mobile Geräte in einer eigenen Datei ablegen:

```
<link rel="stylesheet" type="text/css"
  href="CSS/mobile.css" media="only screen and (max-device-width: 480px)">
```

Damit ist unser Blog sofort auf iPhones besser lesbar. Mit diesem Ansatz können Sie auch Stylesheets für andere Anzeigen erstellen, wie etwa Kioske, Tablet-PCs und andere Bildschirme verschiedenster Größen, damit Ihr Inhalt auf möglichst vielen Geräten lesbar ist.

Ausweichlösung

Media Queries werden von Firefox, Chrome, Safari, Opera und dem Internet Explorer 9 unterstützt. Als Ausweichlösung müssen Sie auf JavaScript zurückgreifen, um zusätzliche Stylesheets für das jeweilige Gerät zu laden. Unser Beispiel ist für iPhones gedacht, daher brauchen wir keine Ausweichlösung – unser Inhalt ist auch ohne Media Query lesbar.

Falls Sie aber mit Media Queries in anderen Browsern experimentieren möchten: Es gibt ein jQuery-Plugin⁷ für die grundlegende Unterstützung von Media Queries in anderen Browsern. Die Möglichkeiten sind insofern eingeschränkt, als dass das Plugin nur mit verknüpften Stylesheets funktioniert und nur `min-width` und `max-width` in Pixel unterstützt. Aber selbst mit diesen Einschränkungen können Sie damit sehr gut Benutzeroberflächen für verschiedene Fenstergrößen erstellen.

Die Zukunft

Mit den Dingen, über die wir in diesem Kapitel gesprochen haben, können wir zwar unsere Benutzerflächen verbessern. Unsere Produkte sind aber auch für Benutzer verwendbar, deren Browser die neuen Funktionen nicht unterstützen. Die Leute können die Daten in unserer Tabelle auch lesen, wenn sie keine Streifen hat. Die Formulare funktionieren auch ohne runde Ecken, genauso wie der Newsletter mit nur einer Spalte. Es ist aber gut zu wissen, dass diese Effekte auch auf der Präsentationsebene ganz ohne JavaScript oder serverseitige Lösungen möglich sind.

Bis auf den Internet Explorer unterstützen mittlerweile fast alle Browser die neuen Selektoren. Sie können aber davon ausgehen, dass sie mit der Zeit auch von IE unterstützt werden, insbesondere die Pseudoklassen. Wenn die endgültige Spezifikation veröffentlicht wird, entfallen die browserspezifischen Präfixe wie `moz` und `webkit`. Dann können Sie auch Ihre Ausweichlösungen abschaffen.

7 <http://plugins.jquery.com/project/MediaQueries>

Kapitel 5

Mehr Barrierefreiheit

Viele der neuen Elemente in HTML5 helfen Ihnen dabei, Inhalte genauer zu beschreiben. Dies wird umso entscheidender, wenn andere Programme Ihren Code übersetzen. Manche Menschen verwenden zum Beispiel Bildschirmlesegeräte oder Screenreader, um sich die grafischen Inhalte des Bildschirms in Text übersetzen und laut vorlesen zu lassen. Screenreader interpretieren den Text auf dem Bildschirm und das entsprechende Markup, um Links, Bilder und andere Elemente zu erkennen. Diese Geräte haben erstaunliche Fortschritte gemacht, hinken aber immer ein bisschen hinter den neuesten Trends her. Live-Bereiche auf Seiten, in denen durch Polling oder Ajax-Anfragen der Inhalt verändert wird, sind schwierig zu erkennen. Aufwendige Seiten sind teilweise schwierig zu navigieren, da der Screenreader eine Menge Text vorlesen muss.

„Accessibility for Rich Internet Applications (WIA-ARIA)“¹ ist eine Spezifikation zur Verbesserung der Barrierefreiheit von Webseiten, insbesondere von Webanwendungen. Sie ist besonders nützlich, wenn Sie Anwendungen mit JavaScript-Steuerelementen und Ajax entwickeln. Manche Teile der WIA-ARIA-Spezifikation wurden in HTML5 eingeführt, während andere außen vor bleiben und die HTML5-Spezifikation ergänzen können. Viele Bildschirmlesegeräte verwenden bereits Funktionen der WIA-ARIA-Spezifikation, dazu gehören JAWS, Window Eyes und sogar die integrierte Voice Over-Funktion von Apple. ARIA führt auch zusätzliches Markup ein, das unterstützende Technologien verwenden können, um aktualisierbare Bereiche zu erkennen.

1 <http://www.w3.org/WAI/intro/aria.php>

In diesem Kapitel sehen wir uns an, wie HTML5 die Erfahrung jener Besucher verbessern kann, die unterstützende Geräte verwenden. Das Beste daran ist:

Für die in diesem Kapitel beschriebenen Techniken benötigen wir keine Ausweidlösung, da viele Screenreader jetzt schon diese Technologien nutzen können.

Dazu gehören:²

Das role-Attribut [`<div role="document">`]

Gibt die Aufgabe eines Elements für den Screenreader an.
[C3, F3.6, S4, IE8, O9.6]

aria-live [`<div aria-live="polite">`]

Kennzeichnet einen Bereich, der automatisch aktualisiert wird, möglicherweise durch Ajax. [F3.6 (Windows), S4, IE8]

aria-atomic [`<div aria-live="polite" aria-atomic="true">`]

Gibt an, ob der gesamte Inhalt eines Live-Bereichs oder nur die veränderten Elemente vorgelesen werden sollen.
[F3.6 (Windows), S4, IE8]

² In den folgenden Beschreibungen wird die Unterstützung durch die verschiedenen Browser in eckigen Klammern mit einem Kurzcode und der mindestens erforderlichen Versionsnummer angegeben. Die verwendeten Codes lauten: C: Google Chrome, F: Firefox, IE: Internet Explorer, O: Opera, S: Safari, IOS: iOS-Geräte mit Mobile Safari und A: Android-Browser.

Die meisten Websites haben eine typische Struktur gemeinsam: Es gibt einen Kopfbereich, einen Navigationsbereich, den Hauptinhalt und eine Fußzeile. Genauso ist auch der Code der meisten Websites aufgebaut: linear. Leider bedeutet das, dass ein Bildschirmlesegerät unter Umständen die gesamte Website auch in dieser Reihenfolge vorlesen muss. Da sich auf den meisten Websites derselbe Kopfbereich und dieselbe Navigation auf jeder Seite wiederholen, müssen die Benutzer sich diese Elemente jedes Mal wieder anhören, um von einer Seite zur nächsten zu navigieren.

Die bevorzugte Abhilfe besteht in einem versteckten Link „Navigation überspringen“, den Screenreader laut vorlesen und der auf einen Anker im Hauptinhalt verweist. Allerdings ist diese Funktionalität nicht von Haus aus integriert – und nicht jeder weiß, wie das funktioniert, oder denkt überhaupt daran.

Mit dem neuen HTML5-Attribut `role` können wir jedem Element auf Ihrer Seite eine „Zuständigkeit“ zuweisen. Bildschirmlesegeräte können dann ganz einfach die Seite einlesen und diese Zuständigkeiten kategorisieren und einen einfachen Index für die Seite erstellen. So werden beispielsweise alle Elemente mit der Rolle `navigation` auf der Seite gesucht und den Benutzern erklärt, damit sie schnell durch die Anwendung navigieren können.

Diese Rollen stammen aus der WIA-ARIA-Spezifikation³ und wurden in die HTML5-Spezifikation integriert. Es gibt zwei bestimmte Rollenklassifizierungen, die Sie bereits jetzt einsetzen können: Landmark- und Dokument-Rollen.

Landmark-Rollen

Landmark-Rollen kennzeichnen „interessante Punkte“ auf Ihrer Website, wie etwa Banner, den Suchbereich oder die Navigation, die Screenreader schnell erkennen können.

³ <http://www.w3.org/WAI/PF/aria/roles>

Rolle	Verwendung
banner	Kennzeichnet den Bannerbereich Ihrer Seite.
search	Kennzeichnet den Suchbereich Ihrer Seite.
navigation	Kennzeichnet Navigationselemente auf Ihrer Seite.
main	Kennzeichnet die Stelle, an der der Hauptinhalt Ihrer Seite beginnt.
contentinfo	Gibt an, wo Informationen über den Inhalt vorhanden sind, zum Beispiel Copyright-Informationen und das Veröffentlichungsdatum.
complementary	Kennzeichnet Inhalte auf einer Seite, die den Hauptinhalt ergänzen, aber auch eine eigenständige Bedeutung haben.
application	Kennzeichnet einen Seitenbereich, der statt einem Webdokument eine Anwendung enthält.

Wir können einige dieser Rollen auf die Blog-Vorlage von AwesomeCo anwenden, die wir im Abschnitt *Einen Blog mit semantischem Markup neu definieren* auf Seite 26 erarbeitet haben.

Für den Kopfbereich vergeben wir die Rolle banner:

```
html5_aria/blog/index.html
```

```
<header id="page_header" role="banner">
  <h1>AwesomeCo Blog!</h1>
</header>
```

Wir müssen lediglich `role="banner"` zu dem vorhandenen `header`-Tag hinzufügen.

Unsere Navigation können wir auf dieselbe Weise kennzeichnen:

```
html5_aria/blog/index.html
```

```
<nav role="navigation">
  <ul>
    <li><a href="/">Latest Posts</a></li>
    <li><a href="/archives">Archives</a></li>
    <li><a href="/contributors">Contributors</a></li>
    <li><a href="/contact">Contact Us</a></li>
  </ul>
</nav>
```

Die HTML5-Spezifikation besagt, dass manche Elemente standardmäßige Rollen haben, die nicht überschrieben werden können. Das Element `nav` muss die Rolle `navigation` haben und daher technisch gesehen nicht damit gekennzeichnet werden. Nicht alle Bildschirmlesegeräte akzeptieren diesen Standard, aber viele verstehen die ARIA-Rollen.

Unsere Haupt- und Seitenleistenbereiche lassen sich wie folgt kennzeichnen:

```
html5_aria/blog/index.html
```

```
<section id="posts" role="main">
</section>
```

```
html5_aria/blog/index.html
```

```
<section id="sidebar" role="complementary">

<nav>
  <h3>Archives</h3>
  <ul>
    <li><a href="2010/10">October 2010</a></li>
    <li><a href="2010/09">September 2010</a></li>
    <li><a href="2010/08">August 2010</a></li>
    <li><a href="2010/07">July 2010</a></li>
    <li><a href="2010/06">June 2010</a></li>
    <li><a href="2010/05">May 2010</a></li>
    <li><a href="2010/04">April 2010</a></li>
    <li><a href="2010/03">March 2010</a></li>
    <li><a href="2010/02">February 2010</a></li>
    <li><a href="2010/01">January 2010</a></li>
  </ul>
</nav>
</section> <!-- Seitenleiste -->
```

Die Informationen zur Veröffentlichung und zum Copyright in unserem Fußbereich kennzeichnen wir mit der Rolle `contentinfo`:

```
html5_aria/blog/index.html
```

```
<footer id="page_footer" role="contentinfo">
  <p>&copy; 2010 AwesomeCo.</p>
</footer> <!-- Fußzeile -->
```

Wenn es in unserem Blog eine Suchfunktion gäbe, könnten wir diesen Bereich ebenso kennzeichnen. Nachdem wir jetzt die Orientierungspunkte gekennzeichnet haben, gehen wir einen Schritt weiter und identifizieren einige der Dokumentslemente.

Rollen für die Dokumentstruktur

Die Rollen für die Dokumentstruktur helfen Screenreadern dabei, die Teile mit statischen Inhalten leichter zu erkennen, wodurch sich der Inhalt für die Navigation leichter strukturieren lässt.

Rolle	Verwendung
<code>document</code>	Kennzeichnet einen Bereich mit Dokumentinhalt, im Gegensatz zu Anwendungsinhalten.
<code>article</code>	Kennzeichnet eine Zusammenstellung, die einen unabhängigen Dokumentteil darstellt.
<code>definition</code>	Kennzeichnet die Definition eines Begriffs oder Themas.
<code>directory</code>	Kennzeichnet eine Liste von Referenzen auf eine Gruppe, wie etwa ein Inhaltsverzeichnis. Für statischen Inhalt.
<code>heading</code>	Kennzeichnet eine Überschrift für einen Abschnitt einer Seite.
<code>img</code>	Kennzeichnet einen Abschnitt, der Bildelemente enthält. Dabei kann es sich ebenso um Bildelemente wie um Beschriftungen und beschreibenden Text handeln.
<code>list</code>	Kennzeichnet eine Gruppe nicht interaktiver Listenelemente.
<code>listitem</code>	Kennzeichnet ein einzelnes Mitglied einer Gruppe nicht interaktiver Listenelemente.
<code>math</code>	Kennzeichnet einen mathematischen Ausdruck.
<code>note</code>	Kennzeichnet Inhalt, der ausgeklammert ist oder den Hauptinhalt der Ressource ergänzt.
<code>presentation</code>	Kennzeichnet Inhalt, der der Darstellung dient und daher von unterstützenden Technologien ignoriert werden kann.
<code>row</code>	Kennzeichnet eine Zeile von Zellen in einem Gitter.
<code>rowheader</code>	Kennzeichnet eine Zelle mit Header-Informationen für eine Zeile in einem Gitter.

Viele Dokumentrollen sind implizit durch HTML-Tags definiert, wie etwa `article` und `header`. Allerdings ist die Rolle `document` nicht implizit definiert. Und das ist eine sehr wichtige Rolle, insbesondere für Anwendungen, in denen dynamischer und statischer Inhalt gemischt werden. So kann zum Beispiel in einem webbasierten E-Mail-Client dem Element, das den Body der E-Mail enthält, die Rolle `document` zugewiesen werden. Das ist nützlich, weil Bildschirmlesegeräte oft verschiedene

Methoden für die Navigation mit der Tastatur anbieten. Wenn der Fokus des Screenreaders auf einem Anwendungselement liegt, müssen Tastenanschläge unter Umständen an die Webanwendung durchgereicht werden. Befindet sich der Fokus dagegen auf statischem Inhalt, können die Tastenbelegungen des Screenreaders andere Aufgaben übernehmen.

In unserem Blog können wir die Rolle `document` auf das `body`-Element anwenden:

```
html5_aria/blog/index.html
```

```
<body role="document">
```



Joe fragt ...

Brauchen wir diese Landmark-Rollen für Elemente wie `nav` und `header`?

Die Landmark-Rollen erscheinen im ersten Moment vielleicht redundant, aber sie bieten die Flexibilität, die Sie dann benötigen, wenn Sie die neuen Elemente nicht verwenden können.

Mit der Rolle `search` können Sie Ihre Benutzer nicht nur zu dem Seitenbereich führen, der das Suchfeld enthält, sondern auch zu Links auf eine Sitemap, zu einer Dropdown-Liste mit Quick-Links oder zu anderen Elementen, die Ihren Benutzern helfen können, Informationen schnell zu finden. Sie sind also nicht darauf beschränkt, Ihre Benutzer lediglich zum Suchfeld selbst zu führen.

Mit der Spezifikation werden eine Menge mehr Rollen eingeführt als neue Elemente und Formularsteuerelemente.

Dies hilft, dafür zu sorgen, dass Bildschirmlesegeräte Ihre Seite als statischen Inhalt behandeln.

Ausweichlösung

Diese Rollen können bereits mit den neuesten Browsern und den neuesten Screenreadern verwendet werden. Sie können also ab sofort damit arbeiten. Browser, die die Rollen nicht unterstützen, ignorieren sie einfach. Sie können damit also auch nur den Menschen helfen, die sie verwenden können.

12

Barrierefreie aktualisierbare Bereiche erstellen

Wir machen heutzutage eine Menge mit Ajax in unseren Webanwendungen. Typischerweise lösen wir irgendeinen visuellen Effekt aus, um den Benutzer darauf hinzuweisen, dass sich etwas auf der Seite verändert hat. Personen, die einen Screenreader verwenden, sind aber aus naheliegenden Gründen nicht dazu in der Lage, solche visuellen Hinweise zu erkennen. Die WIA-ARIA-Spezifikation bietet eine Alternativlösung, die derzeit in IE, Firefox und Safari mit verschiedenen beliebten Bildschirmlesegeräten funktioniert.

Der Kommunikationsdirektor von AwesomeCo möchte eine neue Homepage. Sie soll Links auf die Abschnitte „Services“, „Contact“ und „About“ enthalten. Er besteht darauf, dass die Seite nicht gescrollt werden soll, weil „viele Leute es hassen, zu scrollen“. Er möchte, dass Sie einen Prototyp für die Seite mit einem horizontalen Menü erstellen, über das per Klick der Hauptinhalt der Seite geändert werden kann. Das ist einfach zu implementieren, und mit dem Attribut `aria-live` können wir sogar etwas tun, das bisher nicht so ohne Weiteres möglich war – eine derartige Oberfläche screenreaderfreundlich implementieren.

Erstellen wir eine einfache Oberfläche wie in Abbildung 5.1. Wir stellen den gesamten Inhalt auf die Homepage. Falls JavaScript verfügbar ist, blenden wir alles bis auf den ersten Eintrag aus. Wir lassen die Navigationslinks mit Seitenankern auf den jeweiligen Abschnitt verweisen und verwenden jQuery, um diese Links in Events umzuwandeln, die den Hauptinhalt austauschen. Leute mit JavaScript sehen, was unser Direktor möchte, und Menschen ohne JavaScript können trotzdem den gesamten Inhalt der Seite sehen.

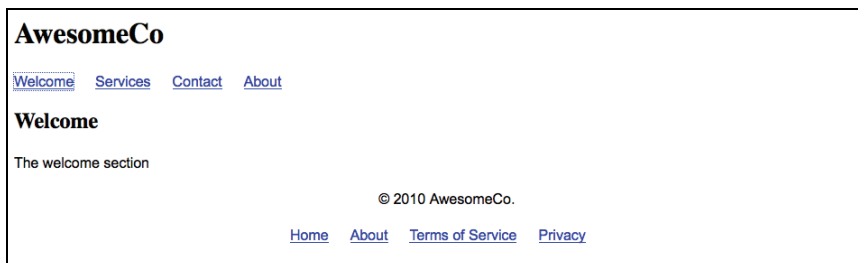


Abbildung 5.1: Mock-up der Homepage mit Verwendung von jQuery zum Austauschen des Hauptinhalts

Die Seite erstellen

Wir beginnen damit, eine einfache HTML5-Seite zu erstellen und fügen unseren Abschnitt „Welcome“ ein, der standardmäßig angezeigt wird, wenn Benutzer die Seite besuchen. Hier sehen Sie den Code für die Seite mit der Navigationsleiste und den Jump-Links:

html5_aria/homepage/index.html

```
<!DOCTYPE html>
<html lang="en-US">
  <head>
    <meta http-equiv="Content-Type" content="text/html; charset=utf-8" />
    <title>AwesomeCo</title>
    <link rel="stylesheet" href="style.css" type="text/css">
  </head>
  <body>
    <header id="header">
      <h1>AwesomeCo </h1>
      <nav>
        <ul>
          <li><a href="#welcome">Welcome</a></li>
          <li><a href="#services">Services</a></li>
          <li><a href="#contact">Contact</a></li>
          <li><a href="#about">About</a></li>
        </ul>
      </nav>
    </header>
    <section id="content"
      role="document" aria-live="assertive" aria-atomic="true">

      <section id="welcome">
        <header>
          <h2>Welcome</h2>
        </header>
        <p>The welcome section</p>
      </section>
    </section>
    <footer id="footer">
      <p>&copy; 2010 AwesomeCo.</p>
      <nav>
        <ul>
          <li><a href="http://awesomeco.com/">Home</a></li>
          <li><a href="about">About</a></li>
          <li><a href="terms.html">Terms of Service</a></li>
          <li><a href="privacy.html">Privacy</a></li>
        </ul>
      </nav>
    </footer>

  </body>
</html>
```

Der Abschnitt „Welcome“ erhält die ID `welcome`, die dem Anker in der Navigationsleiste entspricht. Die übrigen Seitenabschnitte können wir auf dieselbe Weise deklarieren.

```
html5_aria/homepage/index.html
```

```
<section id="services">
  <header>
    <h2>Services</h2>
  </header>
  <p>The service section</p>
</section>

<section id="contact">
  <header>
    <h2>Contact</h2>
  </header>
  <p>The contact section</p>
</section>

<section id="about">
  <header>
    <h2>About</h2>
  </header>
  <p>The about section</p>
</section>
```

Unsere vier Inhaltsbereiche verpacken wir in das folgende Markup:

```
html5_aria/homepage/index.html
```

```
<section id="content"
  role="document" aria-live="assertive" aria-atomic="true">
```

Die Attribute in dieser Zeile teilen Screenreadern mit, dass dieser Seitenbereich aktualisiert wird.

Höfliche, aber bestimmte Aktualisierung

Es gibt zwei verschiedene Methoden, um den Benutzer bei der Verwendung von `aria-live` auf Änderungen der Seite hinzuweisen. Die Methode `polite` wurde entwickelt, um den Arbeitsfluss des Benutzers nicht zu unterbrechen. Wenn der Screenreader beispielsweise einen Satz vorliest, während ein Bereich der Seite aktualisiert wird und der Modus `polite` gewählt ist, liest das Bildschirmlesegerät den aktuellen Satz zu Ende. Ist dagegen der Modus `assertive` festgelegt, wird für diesen Inhalt eine hohe Priorität angenommen: Der Screenreader hört auf, den aktuellen Satz vorzulesen und beginnt mit dem Vorlesen des neuen Inhalts. Es ist wirklich wichtig, dass Sie sich bei der Entwicklung Ihrer Website für die richtige Art der Unterbrechung entscheiden. Der übermäßige Einsatz von `assertive` kann dazu führen, dass Ihre Benutzer die Orientierung verlieren und verwirrt werden. Verwenden Sie daher `assertive` nur, wenn es absolut notwendig ist. In unserem Fall ist es die richtige Entscheidung, weil wir den anderen Inhalt ausblenden.

Alles vorlesen lassen

Der zweite Parameter, `aria-atomic=true`, weist den Screenreader an, den gesamten Inhalt des veränderten Abschnitts zu lesen. Wenn wir dafür den Wert `false` angeben, weisen wir damit den Screenreader an, nur veränderte Knoten vorzulesen. Wir ersetzen den gesamten Inhalt, deshalb ist es sinnvoll, das Bildschirmlesegerät anzuweisen, auch alles vorzulesen. Wenn wir dagegen nur ein einzelnes Listenelement ersetzen oder etwas mit Ajax zu einer Tabelle hinzufügen, wäre `false` die richtige Entscheidung.

Bereiche ausblenden

Zum Ausblenden der Bereiche müssen wir ein bisschen JavaScript schreiben und in unsere Seite einfügen. Wir erstellen hierzu eine Datei mit dem Namen `application.js` und binden diese Datei ebenso wie die jQuery-Bibliothek in unsere Seite ein.

`html5_aria/homepage/index.html`

```
<script type="text/javascript"
  charset="utf-8"
  src="http://ajax.googleapis.com/ajax/libs/jquery/1.4.2/jquery.min.js">
</script>

<script type="text/javascript"
  charset="utf-8"
  src="javascripts/application.js">
</script>
```

Unsere Datei `application.js` enthält ein einfaches Skript:

`css3transitions/style.css`

```
1 // Unterstützung der HTML5-Strukturelemente für IE 6, 7 und 8
2 document.createElement("header");
3 document.createElement("footer");
4 document.createElement("section");
5 document.createElement("aside");
6 document.createElement("article");
7 document.createElement("nav");
8
9 $(function(){
10
11   $("#services, #about, #contact").hide().addClass("hidden");
12   $("#welcome").addClass("visible");
13
14   $("nav ul").click(function(event){
15
16     target = $(event.target);
17     if(target.is("a")){
18       event.preventDefault();
19       if ( $(target.attr("href")).hasClass("hidden") ){
20         $(".visible").removeClass("visible")
```

```

21         .addClass("hidden")
22         .hide();
23     $(target.attr("href"))
24         .removeClass("hidden")
25         .addClass("visible")
26         .show();
27     };
28 };
29
30 });
31
32 });

```

In Zeile 11 blenden wir die Abschnitte „services“, „about“ und „contact“ aus. Außerdem wenden wir darauf die Klasse `hidden` an und weisen in der nächsten Zeile dem Abschnitt `welcome` die Klasse `visible` zu. Auf diese Weise ist es wirklich einfach festzustellen, welche Abschnitte beim Umschalten aktiviert oder deaktiviert werden müssen.

In Zeile 14 fangen wir sämtliche Klicks auf die Navigationsleiste ab und ermitteln in Zeile 17, welches Element angeklickt wurde. Wenn der Benutzer auf einen Link geklickt hat, prüfen wir, ob der entsprechende Abschnitt versteckt ist. Über das `href`-Attribut des geklickten Links können wir mit jQuery-Selektoren den entsprechenden Abschnitt finden, wie Sie in Zeile 19 sehen.

Ist der Abschnitt ausgeblendet, zeigen wir den gewählten Abschnitt an und blenden alle anderen aus. Das war's. Der Screenreader sollte die Änderungen dieses Bereichs feststellen.

Ausweidlösung

Wie auch die Rollen kann diese Lösung sofort mit den neuesten Versionen von Screenreadern verwendet werden. Wenn Sie sich an einige bewährte Regeln halten, zum Beispiel an den Verzicht auf aufdringliches JavaScript, haben wir eine einfache Implementierung, die für ein durchaus breites Publikum funktioniert. Ältere Browser und Bildschirmlesegeräte ignorieren die zusätzlichen Attribute, sodass es auch keine Gefahr darstellt, wenn wir sie in unseren Markup einfügen.

Die Zukunft

HTML5 und die WIA-ARIA-Spezifikation haben den Weg für ein wesentlich barrierefreieres Web geebnet. Mit der Möglichkeit, veränderliche Bereiche der Seite zu kennzeichnen, können Entwickler umfangreichere JavaScript-Anwendungen entwickeln, ohne sich allzu viele Gedanken über Barrierefreiheit machen zu müssen.

Teil II

Neue Perspektiven und Klänge

Kapitel 6

Zeichnen mit dem canvas-Element

Im zweiten Teil dieses Buchs gehen wir von der Besprechung von Strukturen und Oberflächen dazu über, uns anzusehen, wie wir mit HTML5 und CSS3 zeichnen, mit Multimediateien arbeiten und unsere eigenen Benutzeroberflächenelemente erstellen können. Zu Beginn verbringen wir einige Zeit damit, mit dem neuen HTML5-Element canvas Grafiken zu erstellen.

Wenn Sie in einer Webanwendung ein Bild verwenden möchten, öffnen Sie üblicherweise die Grafiksoftware Ihrer Wahl, erstellen ein Bild und betten es mit dem `img`-Tag in Ihre Seite ein. Brauchen Sie eine Animation, verwenden Sie Flash. Mit dem in HTML5 neu hinzugekommenen canvas-Element können Entwickler dagegen im Browser Bilder und Animationen mit JavaScript programmieren. Mit canvas können wir einfache bis komplexe Formen oder sogar Graphen und Diagramme erstellen, ohne auf serverseitige Bibliotheken, Flash oder andere Plugins zurückzugreifen. Rein zufällig werden wir in diesem Kapitel¹ beides tun.

```
<canvas> [<audio src="drums.mp3"></audio>]
```

Unterstützt die Erstellung vektorbasierter Grafiken mit JavaScript. *[C4, F3, IE9, S3.2, O10.1, IOS3.2, A2]*

¹ In den folgenden Beschreibungen wird die Unterstützung durch die verschiedenen Browser in eckigen Klammern mit einem Kurzcode und der mindestens erforderlichen Versionsnummer angegeben. Die verwendeten Codes lauten: C: Google Chrome, F: Firefox, IE: Internet Explorer, O: Opera, S: Safari, IOS: iOS-Geräte mit Mobile Safari und A: Android-Browser.

Zunächst machen wir uns mit der Verwendung des canvas-Elements mit JavaScript vertraut, indem wir einige einfache Formen zeichnen und eine Version des AwesomeCo-Logos erstellen. Anschließend arbeiten wir mit einer Graphen-Bibliothek, die speziell dafür entwickelt wurde, mit dem canvas-Element Balkendiagramme für Browser-Statistiken zu zeichnen. Außerdem behandeln wir noch einige besondere Herausforderungen bei den Ausweichlösungen, die sich daraus ergeben, dass das canvas-Element eher eine Programmierschnittstelle als ein Element ist.

13

Logos zeichnen

Das canvas-Element ist genauso wie das script-Element ein Container, sozusagen eine leere Tafel, auf der wir zeichnen können. So definieren Sie ein canvas-Element mit einer bestimmten Breite und Höhe:

```
html5canvasgraph/canvas_simple_drawing.html
```

```
<canvas id="my_canvas" width="150" height="150">
  Ausweidlösung hierhin
</canvas>
```

Leider können Sie die Breite und Höhe eines canvas-Elements nicht mit CSS anpassen, ohne die Inhalte zu verzerren. Insofern müssen Sie sich also schon bei der Deklaration für die Maße Ihres canvas-Elements entscheiden.

Wir verwenden JavaScript, um Formen darauf zu zeichnen. Selbst wenn Sie für Browser ohne canvas-Element alternativen Inhalt bereitstellen, müssen Sie dennoch verhindern, dass dieser durch den JavaScript-Code verändert wird. Suchen Sie das canvas-Element anhand seiner ID, und prüfen Sie, ob der Browser die getContext-Methode des canvas-Elements unterstützt.

```
html5canvasgraph/canvas_simple_drawing.html
```

```
var canvas = document.getElementById('my_canvas');
if (canvas.getContext){
  var context = canvas.getContext('2d');

}else{
  // Versteckten Inhalt des <canvas>-Elements anzeigen oder
  // den Browser den darin enthaltenen Text anzeigen lassen.
}
```

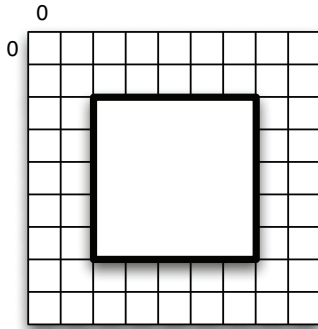
Wenn die getContext-Methode etwas zurückliefert, rufen wir den 2D-Kontext des canvas-Elements auf, damit wir Objekte hinzufügen können. Wenn wir keinen Kontext erhalten, müssen wir eine Möglichkeit finden, den Alternativinhalt anzuzeigen. Da wir wissen, dass das canvas-Element nur mit JavaScript funktioniert, bauen wir von Anfang an ein Framework für die Ausweidlösung auf.

Sobald Sie über den Kontext des canvas-Elements verfügen, können Sie Elemente dazu hinzufügen. Ein rotes Rechteck fügen Sie zum Beispiel ein, indem Sie die Füllfarbe festlegen und dann ein Rechteck erstellen:

html5canvasgraph/canvas_simple_drawing.html

```
context.fillStyle = "rgb(200,0,0)";
context.fillRect (10, 10, 100, 100);
```

Der 2D-Kontext des canvas-Elements ist ein Raster, dessen standardmäßiger Ursprung die linke obere Ecke ist. Zum Zeichnen einer Form geben Sie die X- und Y-Koordinaten sowie Breite und Höhe an.



Jede Form wird in einer eigenen Ebene gezeichnet, sodass Sie beispielsweise auch drei Rechtecke mit einem Versatz von je 10 Pixel erstellen können:

html5canvasgraph/canvas_simple_drawing.html

```
context.fillStyle = "rgb(200,0,0)";
context.fillRect (10, 10, 100, 100);
context.fillStyle = "rgb(0,200,0)";
context.fillRect (20, 20, 100, 100);

context.fillStyle = "rgb(0,0,200)";
context.fillRect (30, 30, 100, 100);
```

Die Dreiecke werden dann auch entsprechend übereinander geschichtet:



Da Sie nun die Grundlagen bereits verstehen, stellen wir gleich das Logo für AwesomeCo zusammen. Wie Sie in Abbildung 6.1 sehen, ist es relativ simpel.



Abbildung 6.1: Das Awesomeco-Logo

Das Logo zeichnen

Das Logo besteht aus einem Schriftzug, einem gewinkelten Pfad, einem Quadrat und einem Dreieck. Erstellen wir ein neues HTML5-Dokument mit einem canvas-Element und einer JavaScript-Funktion zum Zeichnen des Logos, die ermittelt, ob wir das 2D-Canvas verwenden können.

html5canvasgraph/logo.html

```
var drawLogo = function(){
    var canvas = document.getElementById('logo');
    var context = canvas.getContext('2d');
};
```

Diese Methode rufen wir aber erst auf, nachdem wir geprüft haben, ob das canvas-Element existiert:

html5canvasgraph/logo.html

```
$(function(){
    var canvas = document.getElementById('logo');
    if (canvas.getContext){
        drawLogo();
    }
});
```

Beachten Sie, dass wir auch hier wieder die jQuery-Funktion verwenden, um sicherzustellen, dass das Event ausgelöst wird, wenn das Dokument bereit ist. Wir suchen ein Element mit der ID `logo` auf der Seite, also sollten wir dafür sorgen, dass unser Dokument auch ein canvas-Element enthält. Denn nur so können wir es finden und unsere Prüfung durchführen.

html5canvasgraph/logo.html

```
<canvas id="logo" width="900" height="80">
  <h1>AwesomeCo</h1>
</canvas>
```

Als Nächstes fügen wir den Text „AwesomeCo“ zum canvas-Element hinzu.

Text einfügen

Um Text zum canvas-Element hinzuzufügen, müssen wir eine Schrift, eine Schriftgröße sowie die Ausrichtung wählen und anschließend den Text auf die entsprechenden Koordinaten im Raster anwenden. So fügen wir den Text „AwesomeCo“ in unser canvas-Element ein:

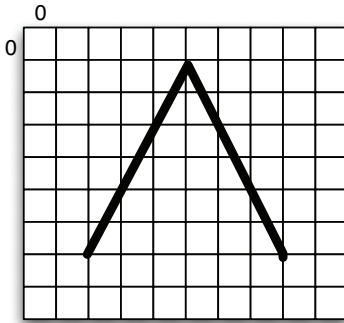
html5canvasgraph/logo.html

```
context.font = 'italic 40px sans-serif';
context.textBaseline = 'top';
context.fillText('AwesomeCo', 60, 0);
```

Wir definieren den Schrifttyp und legen die Grundlinie bzw. die vertikale Ausrichtung fest, bevor wir den Text in das canvas-Element einfügen. Wir verwenden die `fillText`-Methode und erhalten Text, der mit der Füllfarbe gefüllt ist und den wir 60 Pixel nach rechts platzieren, damit wir Platz für den großen dreieckigen Pfad haben, den wir als Nächstes zeichnen.

Linien zeichnen

Linien können wir auf dem canvas-Element ganz einfach mit einer Runde „Malen nach Zahlen“ zeichnen: Wir geben einen Anfangspunkt im Raster an und legen dann zusätzliche Punkte im Raster fest, die mit Punkten verbunden werden:



Wir verwenden die Methode `beginPath()`, um mit dem Zeichnen einer Linie zu beginnen, und erstellen dann unseren Pfad:

html5canvasgraph/logo.html

```
context.lineWidth = 2;
context.beginPath();
context.moveTo(0, 40);
context.lineTo(30, 0);
```

```
context.lineTo(60, 40);
context.lineTo(285, 40);
context.stroke();
context.closePath();
```

Wenn wir damit fertig sind, uns über das canvas-Element zu bewegen, rufen wir die Methode `stroke` auf, um die Linie zu zeichnen. Anschließend rufen wir die Methode `closePath` auf, um den Zeichenvorgang zu beenden.

Bleibt also nur noch die Kombination aus Rechteck und Dreieck innerhalb des größeren Dreiecks.

Den Ursprung verschieben

Innerhalb des größeren Dreiecks müssen wir ein kleines Quadrat und ein kleines Dreieck zeichnen. Wenn wir Formen und Pfade zeichnen, geben wir die X- und Y-Koordinaten relativ zum Ursprung in der linken oberen Ecke des canvas-Elements an. Wir können den Ursprung aber auch an eine neue Position verschieben.

Zum Zeichnen des inneren kleinen Quadrats verschieben wir den Ursprung:

```
html5canvasgraph/logo.html
```

```
context.save();
context.translate(20,20);
context.fillRect(0,0,20,20);
```

Beachten Sie, dass wir vor dem Verschieben des Ursprungs die Methode `save()` aufrufen. Hierdurch wird der bisherige Zustand des canvas-Elements gespeichert, sodass wir jederzeit Änderungen rückgängig machen können. Das funktioniert wie ein Wiederherstellungspunkt – Sie können sich das wie einen Stapel vorstellen. Jedes Mal, wenn Sie `save()` aufrufen, erhalten Sie einen neuen Eintrag. Wenn wir fertig sind, rufen wir `restore()` auf, wodurch die jeweils letzte Speicherung auf dem Stapel wiederhergestellt wird.

Nun zeichnen wir mit Pfaden das innere Dreieck. Aber statt Linien verwenden wir eine Füllung, um den Eindruck zu erwecken, dass das Dreieck aus dem Quadrat „herausgeschnitten“ wird.

```
html5canvasgraph/logo.html
```

```
context.fillStyle = '#fff';
context.strokeStyle = '#fff';
```

```

context.lineWidth = 2;
context.beginPath();
context.moveTo(0, 20);
context.lineTo(10, 0);
context.lineTo(20, 20);
context.lineTo(0, 20);
context.fill();
context.closePath();
context.restore();

```

In diesem Fall legen wir für die Striche und die Füllung die Farbe Weiß (#fff) fest, bevor wir mit dem Zeichnen beginnen. Anschließend zeichnen wir unsere Linien. Nachdem wir zuvor den Ursprung verschoben haben, bewegen wir uns nun relativ zur oberen linken Ecke des zuvor gezeichneten Quadrats.

Wir sind beinahe fertig, aber es fehlt noch ein bisschen Farbe.

Farben

Im Abschnitt *Den Ursprung verschieben* auf der vorherigen Seite haben Sie kurz gesehen, wie Sie die Striche und die Füllfarbe für die Zeichenwerkzeuge festlegen. Wir könnten beispielsweise für alles die Farbe Rot wählen, indem wir vor dem Zeichnen den folgenden Code einfügen:

```
html5canvasgraph/logo.html
```

```

context.fillStyle = "#f00";
context.strokeStyle = "#f00";

```

Aber das wäre ein bisschen langweilig. Stattdessen erzeugen wir Verläufe und weisen diese den Strichen und Füllungen zu:

```
html5canvasgraph/logo_gradient.html
```

```

var gradient = context.createLinearGradient(0, 0, 0, 40);
gradient.addColorStop(0, '#a00'); // Rot
gradient.addColorStop(1, '#f00'); // Rot
context.fillStyle = gradient;
context.strokeStyle = gradient;

```

Wir erstellen einfach ein Gradient-Objekt und legen die Farbwerte fest. In diesem Beispiel bewegen wir uns nur zwischen zwei Rotschattierungen. Wir könnten aber auch den gesamten Regenbogen zeichnen, wenn wir das wollten.²

² Bitte tun Sie das nicht.

Beachten Sie, dass wir die Farbe festlegen müssen, bevor wir etwas zeichnen.

Damit ist unser Logo fertig und wir haben ein besseres Verständnis davon, wie wir einfache Formen im canvas-Element zeichnen können. Allerdings bietet der Internet Explorer vor Version 9 keine Unterstützung für das canvas-Element. Darum müssen wir uns kümmern.

Ausweichlösung

Google hat eine Bibliothek mit dem Namen ExplorerCanvas³ veröffentlicht, die den größten Teil der Canvas-API auch Benutzern des Internet Explorer zur Verfügung stellt. Wir müssen lediglich diese Bibliothek in unsere Seite einbinden:

```
html5canvasgraph/logo_gradient.html
```

```
<!--[if lte IE 8]>
<script src="javascripts/excanvas.js"></script>
<![endif]-->
```

Und schon sollte alles auch im Internet Explorer funktionieren – tut es aber nicht. Als dieses Buch geschrieben wurde, hat die neueste stabile Version das Einfügen von Text noch gar nicht unterstützt. Und die Versionen aus dem Subversion-Repository⁴ verwenden falsche Schriftarten. Außerdem bietet diese Bibliothek noch keine Unterstützung für Farbverläufe von Strichen.

Also müssen wir auf andere Lösungen zurückgreifen, wie etwa ein PNG des Logos innerhalb des canvas-Elements. Oder wir verwenden das canvas-Element überhaupt nicht. Dieser Teil sollte nur eine Übung sein, um Ihnen zu zeigen, wie Sie zeichnen können. Es ist also kein Weltuntergang, wenn wir dieses bestimmte Beispiel noch nicht in einem plattformübergreifenden Produktionssystem einsetzen können.

³ <http://code.google.com/p/explorercanvas/>

⁴ <http://explorercanvas.googlecode.com/svn/trunk/excanvas.js>

14

Statistiken grafisch darstellen mit RGraph

AwesomeCo macht eine Menge auf der Website, und das obere Management wünscht sich eine grafische Darstellung der Webstatistiken. Die Backend-Programmierer können die Daten in Echtzeit liefern, möchten aber erst einmal sehen, ob Sie eine Möglichkeit finden, die Grafik im Browser darzustellen. Also haben sie Sie mit Testdaten versorgt. Unser Ziel ist es, die Testdaten in etwas umzuwandeln, das an Abbildung 6.2 auf der folgenden Seite erinnert.

Es gibt viele Möglichkeiten, Diagramme auf einer Webseite zu zeichnen. Manche Entwickler verwenden immer Flash, aber das hat den Nachteil, dass es auf einigen mobilen Geräten wie dem iPad oder iPhone nicht funktioniert. Es gibt serverseitige Lösungen, die gut funktionieren, aber für Echtzeitdaten zu rechenintensiv sein könnten. Eine standardbasierte clientseitige Lösung wie das canvas-Element ist eine ausgezeichnete Möglichkeit, solange wir darauf achten, dass es auch in älteren Browsern funktioniert. Sie haben bereits gesehen, wie Sie Quadrate zeichnen können. Aber um etwas Komplexes zu zeichnen, ist eine Menge mehr JavaScript erforderlich. Wir brauchen eine Bibliothek für die grafische Darstellung.

Von der Tatsache, dass HTML5 noch nicht überall verfügbar ist, haben sich die Entwickler der RGraph-Bibliothek⁵ nicht abhalten lassen. Mit RGraph ist es geradezu lächerlich einfach, mit dem canvas-Element von HTML5 Diagramme zu zeichnen. Es handelt sich allerdings um eine reine JavaScript-Lösung, die dementsprechend nicht auf User Agents ohne JavaScript funktioniert. Aber das Gleiche gilt ja auch für das canvas-Element. Hier sehen Sie den Code für ein sehr einfaches Balkendiagramm:

html5canvasgraph/rgraph_bar_example.html

```
<canvas width="500" height="250" id="test">[no canvas support]</canvas>

<script type="text/javascript" charset="utf-8">
  var bar = new RGraph.Bar('test', [50,25,15,10]);
  bar.Set('chart.gutter', 50);
  bar.Set('chart.colors', ['red']);

  bar.Set('chart.title', "A bar graph of my favorite pies");
```

5 <http://www.rgraph.net/>

```

    bar.Set('chart.labels',["Banana Creme", "Pumpkin", "Apple","Cherry" ]);
    bar.Draw();
</script>

```

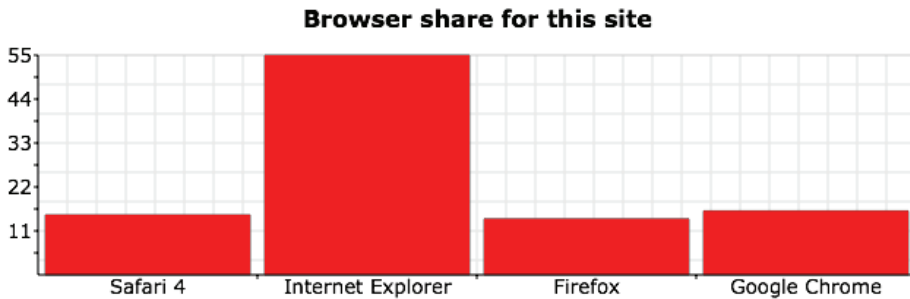


Abbildung 6.2: Clientseitiges Balkendiagramm mit canvas

Wir müssen lediglich einige JavaScript-Arrays anlegen, und schon zeichnet die Bibliothek das Diagramm im canvas-Element.

Daten mit HTML beschreiben

Wir könnten die Werte für die Browserstatistiken fest in den JavaScript-Code schreiben, aber dann können nur Benutzer mit JavaScript diese Werte sehen. Stattdessen schreiben wir die Daten als Text auf die Seite. Anschließend können wir die Daten mit JavaScript einlesen und die Diagramm-Bibliothek damit füttern.

`html5canvasgraph/canvas_graph.html`

```

<div id="graph_data">
  <h1>Browser share for this site</h1>
  <ul>
    <li>
      <p data-name="Safari 4" data-percent="15">
        Safari 4 -15%
      </p>
    </li>
    <li>
      <p data-name="Internet Explorer" data-percent="55">
        Internet Explorer -55%
      </p>
    </li>

    <li>
      <p data-name="Firefox" data-percent="14">
        Firefox -14%
      </p>
    </li>
  </ul>
</div>

```

```

        </li>
      </li>
      <p data-name="Google Chrome" data-percent="16">
        Google Chrome -16%
      </p>
    </li>
  </ul>
</div>

```

Wir verwenden HTML5-Datenattribute, um die Browsernamen und Prozentsätze zu speichern. Diese Informationen stehen zwar auch im Text, wir können aber so viel leichter programmgesteuert damit arbeiten, da wir keine Strings einlesen müssen.

Wenn Sie die Seite in Ihrem Browser öffnen oder einfach einen Blick auf Abbildung 6.3 werfen, sehen Sie, dass die Diagrammdaten hübsch lesbar angezeigt werden – auch ohne Diagramm. Das ist unser Alternativinhalt für mobile Geräte und andere Benutzer, für die entweder das canvas-Element oder JavaScript nicht verfügbar sind.

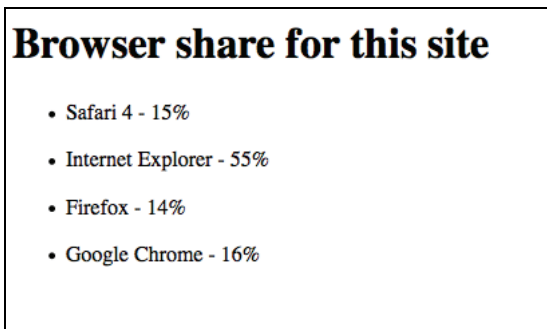


Abbildung 6.3: Unser Diagramm als HTML

Jetzt machen wir aus diesem Markup ein Diagramm.

Ein Balkendiagramm aus HTML erstellen

Wir verwenden ein Balkendiagramm, daher brauchen wir die Balkendiagrammbibliothek und die Hauptbibliothek von RGraph. Um die Daten aus dem Dokument auszulesen, verwenden wir jQuery. Im head der HTML-Seite müssen wir die erforderlichen Bibliotheken laden.

```
html5canvasgraph/canvas_graph.html
```

```
<script type="text/javascript"
  charset="utf-8"
  src="http://ajax.googleapis.com/ajax/libs/jquery/1.4.2/jquery.min.js">
</script>
<script src="javascripts/RGraph.common.js"></script>
<script src="javascripts/RGraph.bar.js"></script>
```

Zum Erstellen des Diagramms müssen wir den Titel, die Beschriftungen und die Daten aus dem HTML-Dokument auslesen und an die RGraph-Bibliothek übergeben. RGraph erwartet sowohl für die Beschriftungen als auch für die Daten Arrays. Diese Arrays können wir mit jQuery schnell erstellen.

```
html5canvasgraph/canvas_graph.html
```

```
1 function canvasGraph(){
2   var title = $('#graph_data h1').text();
3
4   var labels = $("#graph_data>ul>li>p[data-name]").map(function(){
5     return $(this).attr("data-name");
6   });
7
8   var percents = $("#graph_data>ul>li>p[data-percent]").map(function(){
9     return parseInt($(this).attr("data-percent"));
10  });
11
12  var bar = new RGraph.Bar('browsers', percents);
13  bar.Set('chart.gutter', 50);
14  bar.Set('chart.colors', ['red']);
15  bar.Set('chart.title', title);
16  bar.Set('chart.labels', labels);
17  bar.Draw();
18
19 }
```

Als Erstes rufen wir in Zeile 2 den Text für die Kopfzeile ab. In Zeile 4 wählen wir anschließend alle Elemente mit dem Attribut `data-name` aus. Wir verwenden die `map`-Funktion von jQuery, um die Werte dieser Elemente in ein Array zu schreiben.

In Zeile 8 wenden wir dieselbe Logik an, um ein Array mit den Prozentzahlen anzulegen.

Nachdem wir die Daten gesammelt haben, ist es für RGraph ein Leichtes, unser Diagramm zu zeichnen.

Alternativen Inhalt anzeigen

Im Abschnitt *Daten mit HTML beschreiben* auf Seite 123 hätte ich das Diagramm auch zwischen dem öffnenden und dem schließenden canvas-Tag platzieren können. Dadurch würden diese Elemente in Browsern versteckt, die das canvas-Element unterstützen und in Browsern angezeigt, die es nicht unterstützen. Jedoch würde der Inhalt auch verborgen bleiben, wenn der Browser zwar das canvas-Element unterstützt, der Benutzer aber JavaScript deaktiviert hat.

jQuery CSS contra CSS

In diesem Kapitel haben wir unsere Stilregeln mit jQuery auf die Elemente angewendet, während wir sie erstellt haben. Eine Menge dieser Stilinformationen, wie etwa die Farben der Beschriftungen und der Balken, sollten aber in einem separaten Stylesheet untergebracht werden. Umso wichtiger ist das, wenn Sie die Stilregeln unabhängig vom Skript ändern möchten. Für einen Prototyp ist unser bisheriger Ansatz in Ordnung, aber für eine Produktionsversion sollten Sie Darstellung, Verhalten und Inhalt strikt voneinander trennen.

Wir lassen die Daten einfach außerhalb des canvas-Elements stehen und verstecken sie anschließend mit jQuery, sobald wir überprüft haben, ob das canvas-Element existiert.

`html5canvasgraph/canvas_graph.html`

```
var canvas = document.getElementById('browsers');
if (canvas.getContext){
    $('#graph_data').hide();
    canvasGraph();
}
```

Damit ist unser Diagramm fertig, außer für diejenigen, deren Browser das canvas-Element nicht unterstützen.

Ausweichlösung

Beim Erstellen dieser Lösung haben wir bereits Ausweichlösungen für Barrierefreiheit und fehlendes JavaScript abgedeckt. Aber wir können ein alternatives Diagramm für Menschen erstellen, die zwar nicht über die Unterstützung für das canvas-Element verfügen, aber JavaScript verwenden können.

Es gibt tonnenweise Diagramm-Bibliotheken, von denen jede die Daten anders erfasst. Balkendiagramme sind aber letztlich nur Rechtecke mit einer bestimmten Höhe, und wir haben alle Daten auf der Seite, um dieses Diagramm von Hand aufzubauen.

html5canvasgraph/canvas_graph.html

```

1 function divGraph(barColor, textColor, width, spacer, lblHeight){
2   $('#graph_data ul').hide();
3   var container = $("#graph_data");
4
5   container.css( {
6     "display": "block",
7     "position": "relative",
8     "height": "300px" }
9 );
10
11   $("#graph_data>ul>li>p").each(function(i){
12
13     var bar = $("<div>" + $(this).attr("data-percent") + "%</div>");
14     var label = $("<div>" + $(this).attr("data-name") + "</div>");
15
16     var commonCSS = {
17       "width": width + "px",
18       "position": "absolute",
19       "left": i * (width + spacer) + "px" };
20
21     var barCSS = {
22       "background-color": barColor,
23       "color": textColor,
24       "bottom": lblHeight + "px",
25       "height": $(this).attr("data-percent") + "%"
26     };
27     var labelCSS = {"bottom": "0", "text-align": "center" };
28
29     bar.css( $.extend(barCSS, commonCSS) );
30     label.css( $.extend(labelCSS, commonCSS) );
31
32     container.append(bar);
33     container.append(label);
34   });
35 }
```

In Zeile 2 verstecken wir die ungeordnete Liste, damit die Textwerte nicht angezeigt werden. Anschließend nehmen wir das Element mit den Diagrammdaten und wenden einige grundlegende CSS-Stilregeln darauf an. In Zeile 7 legen wir für die Positionierung des Elements relative fest, wodurch wir unsere Balkendiagramme und Beschriftungen innerhalb dieses Containers absolut positionieren können.

Anschließend durchlaufen wir die Absätze der Aufzählungsliste (Zeile 11) und erstellen die Balken. In jedem Durchlauf für die Beschriftungen werden zwei div-Elemente erstellt: eins für den Balken und eins für

die Beschriftung, die wir darunter positionieren. Mit ein bisschen Mathematik und ein bisschen jQuery sind wir also in der Lage, das Diagramm neu zu erfinden. Auch wenn es nicht *genauso* aussieht, so ist es doch ähnlich genug, um unser Konzept zu bestätigen.

Anschließend müssen wir es nur noch in unsere canvas-Prüfung einbauen:

```
html5canvasgraph/canvas_graph.html
```

```
var canvas = document.getElementById('browsers');
if (canvas.getContext){
    $('#graph_data').hide();
    canvasGraph();
}
else{
    divGraph("#f00", "#fff", 140, 10, 20);
}
```

Sie können die Alternativversion in Abbildung 6.4 sehen. Mit einer Kombination aus JavaScript, HTML und CSS haben wir clientseitig ein Balkendiagramm und statistische Informationen zur Browsernutzung für jede beliebige Plattform bereitgestellt. Die Verwendung des canvas-Elements hat einen weiteren Vorteil – sie hat uns dazu gebracht, von Anfang an über eine Ausweidlösung nachzudenken, anstatt nachträglich etwas hineinzquetschen. Das ist ein echtes Plus in puncto Barrierefreiheit.

Diese Methode bietet mit die größtmögliche Barrierefreiheit und Wandlungsfähigkeit, Daten grafisch darzustellen. Sie können die visuelle Darstellung alternativ auch auf einfache Weise textbasiert anbieten. Auf diese Weise kann jeder die wichtigen Daten nutzen, die Sie bereitstellen.

Browser share for this site

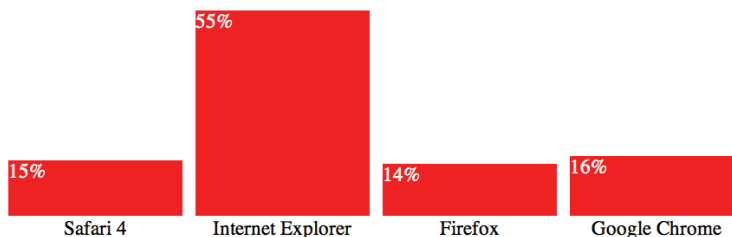


Abbildung 6.4: Unser Diagramm wird jetzt auch im Internet Explorer angezeigt.



Joe fragt ...

Warum haben wir nicht ExplorerCanvas ausprobiert?

Explorer Canvas, von dem wir im Abschnitt *Ausweichlösung* auf Seite 121 gesprochen haben, und RGraph funktionieren wirklich gut zusammen. Die Distribution von RGraph enthält sogar eine Version von ExplorerCanvas. Allerdings funktioniert diese Kombination nur mit dem Internet Explorer 8. Für einen IE 7 oder ältere Versionen müssen Sie als Alternative eine Lösung wie unsere verwenden. Ich möchte Sie ermuntern, ExplorerCanvas im Auge zu behalten, da es ständig weiterentwickelt wird. Sie sollten auch mal darüber nachdenken, selbst an der Entwicklung mitzuwirken, damit Sie ExplorerCanvas für Ihr Projekt nutzen können.

Die Zukunft

Nachdem Sie jetzt ein bisschen über die Funktionsweise des canvas-Elements wissen, können Sie über weitere Einsatzmöglichkeiten nachdenken. Sie könnten damit ein Spiel, eine Benutzeroberfläche für einen Medienplayer oder eine tolle Bildergalerie schreiben. Sie brauchen lediglich etwas JavaScript und ein bisschen Fantasie, um mit dem Zeichnen loszulegen.

Im Moment hat Flash einen gewissen Vorsprung gegenüber dem canvas-Element, weil es weiter verbreitet ist. Aber wenn HTML5 aufholt und canvas für ein breiteres Publikum zur Verfügung steht, werden mehr und mehr Entwickler für einfache 2D-Grafiken im Browser begeistert darauf zurückgreifen. Das canvas-Element erfordert keine zusätzlichen Plugins und benötigt weniger CPU-Leistung als Flash, insbesondere unter Linux und OS X. Zu guter Letzt bietet Ihnen das canvas-Element einen Mechanismus, um 2D-Grafiken auch in Umgebungen zu erstellen, in denen kein Flash verfügbar ist. Da das canvas-Element auf immer mehr Plattformen unterstützt wird, können Sie mit zunehmend mehr Geschwindigkeit und Funktionen rechnen und davon ausgehen, dass Sie von immer mehr Entwicklertools und Bibliotheken dabei unterstützt werden, außergewöhnliche Dinge zu tun.

Aber 2D-Grafiken sind noch nicht alles. Die canvas-Spezifikation wird irgendwann auch 3D-Grafiken unterstützen, und die Browserhersteller werden die Hardwarebeschleunigung implementieren. Mit dem canvas-Element wird es möglich sein, allein mit JavaScript verblüffende Benutzeroberflächen und packende Spiele zu schreiben.

Kapitel 7

Audio und Video einbetten

Audio und Video sind ein wichtiger Bestandteil des modernen Internets. Überall wimmelt es vor Podcasts, Hörproben und Videoanleitungen. Bis jetzt waren sie nur mithilfe von Browser-Plugins verwendbar, aber HTML5 bietet neue Möglichkeiten, um Audio- und Videodateien in Seiten einzubetten. In diesem Kapitel untersuchen wir Methoden, mit denen wir Audio- und Videoinhalte nicht nur einbetten können, sondern mit denen wir dafür sorgen können, dass sie auch für Menschen mit älteren Browsern verfügbar sind.

In diesem Kapitel diskutieren wir die folgenden beiden Elemente:¹

```
<audio> [<audio src="drums.mp3"></audio>]
```

Damit können Sie Audio nativ im Browser wiedergeben.

[C4, F3.6, IE9, S3.2, O10.1, IOS3, A2]

```
<video> [<video src="tutorial.m4v"></video>]
```

Damit können Sie Video nativ im Browser wiedergeben.

[C4, F3.6, IE9, S3.2, O10.5, IOS3, A2]

Bevor wir dazu kommen, müssen wir kurz über die Geschichte von Audio und Video im Web sprechen. Schließlich müssen wir zunächst einmal verstehen, woher wir kommen, um zu verstehen, in welche Richtung wir gehen.

¹ In den folgenden Beschreibungen wird die Unterstützung durch die verschiedenen Browser in eckigen Klammern mit einem Kurzcode und der mindestens erforderlichen Versionsnummer angegeben. Die verwendeten Codes lauten: C: Google Chrome, F: Firefox, IE: Internet Explorer, O: Opera, S: Safari, IOS: iOS-Geräte mit Mobile Safari und A: Android-Browser.

7.1 Ein bisschen Geschichte

Webdesigner versuchen schon seit langer Zeit, Audio und Video in Webseiten zu verwenden. Es fing damit an, dass manche von ihnen mithilfe des `embed`-Tags MIDI-Dateien in ihre Websites eingebettet haben. Damals wurde folgendermaßen mit dem `embed`-Tag auf die Datei verwiesen:

```
<embed src="awesome.mp3" autostart="true"
  loop="true" controller="true"></embed>
```

Das `embed`-Tag wurde nie zum Standard, daher verwendeten die Webdesigner stattdessen das `object`-Tag, einen vom W3C akzeptierten Standard. Um ältere Browser zu unterstützen, die das `object`-Tag nicht verstehen, finden Sie häufig ein `embed`-Tag, das in ein `object`-Tag verschachtelt ist:

```
<object>
<param name="src" value="simpsons.mp3">
<param name="autoplay" value="false">
<param name="controller" value="true">
<embed src="awesome.mp3" autostart="false"
  loop="false" controller="true"></embed>
</object>
```

Trotzdem konnte nicht jeder Browser den Inhalt auf diese Weise streamen, und nicht jeder Server war entsprechend konfiguriert, um den Inhalt korrekt zu liefern. Es wurde alles sogar noch komplizierter, als Video im Web populär wurde. Wir haben jede Menge Schleifen mit Audio- und Videoinhalten im Web durchlaufen, vom Real Player über Windows Media bis hin zu Quick Time. Jedes Unternehmen hatte eine eigene Videostrategie, und scheinbar verwendete jede Website eine andere Methode und ein anderes Format für die Kodierung ihrer Videos im Web.

Macromedia (jetzt Adobe) hat schon früh begriffen, dass sein Flash Player die perfekte Lösung sein könnte, um Audio- und Videoinhalte plattformübergreifend bereitzustellen. Flash ist auf fast 97 Prozent aller Webbrowser verfügbar. Sobald die Inhaltsproduzenten entdeckten, dass sie ihre Inhalte nur einmal kodieren mussten, um sie überall wiedergeben zu können, entschieden sich Tausende von Websites beim Streaming von Audio und Video für Flash.

Dann kam Apple 2007 mit dem iPhone und dem iPod Touch und entschied, dass auf diesen Geräten Flash nicht unterstützt wird. Viele Websitebetreiber reagierten darauf, indem sie Video-Streams zur Verfügung stellten, die direkt im Browser Mobile Safari wiedergegeben werden können. Diese Videos konnten durch Verwendung des Codecs

H.264 auch über einen normalen Flash Player wiedergegeben werden, wodurch Inhalte nach wie vor nur einmal kodiert werden mussten, um mit mehreren Plattformen kompatibel zu sein.

Die Schöpfer der HTML5-Spezifikation sind der Meinung, dass Browser Audio und Video nativ unterstützen sollten, anstatt auf Plugins zurückzugreifen, für die ein ganzer Wust an HTML erforderlich ist. Und genau das macht in HTML5 mehr Sinn: Audio und Video werden unter den Webinhalten endlich Bürger erster Klasse.



Joe fragt ...

**Flash funktioniert bereits browserübergreifend.
Warum sollen wir etwas anderes verwenden?**

Die einfache Antwort lautet, dass es keine herstellerspezifischen Einschränkungen gibt, was Sie als Entwickler mit den Inhalten tun können, nachdem Sie sie in die Seite eingebettet haben. Sie können das Element mit CSS und JavaScript manipulieren und müssen nicht mit Parametern herumfummeln, die Sie an den Flash-Film übergeben. Außerdem wird sich die Situation noch verbessern, wenn der Standard weiter reift.

7.2 Container und Codecs

Wenn wir über Video im Web sprechen, sprechen wir von Containern und Codecs. Sie denken vielleicht an ein Video von Ihrer Digitalkamera als AVI- oder MPEG-Datei, aber das ist eine zu starke Vereinfachung. Container sind wie ein Umschlag, der Audiostreams, Videostreams und manchmal zusätzliche Metadaten wie Untertitel enthält. Diese Audio- und Videostreams müssen kodiert werden – und da kommen Codecs ins Spiel. Video und Audio können auf Hunderte von Arten kodiert werden, aber für Video in HTML5 kommen nur einige wenige infrage.

Video-Codecs

Damit Sie ein Video ansehen können, muss Ihr Videoplayer es dekodieren. Leider kann es vorkommen, dass Ihr Player das Video nicht dekodieren kann, das Sie ansehen möchten. Manche Player verwenden Software, um Videos zu dekodieren, was langsamer und rechenintensiver sein kann. Andere Player verwenden Hardware-Decoder und sind daher in den Formaten eingeschränkt, die sie wiedergeben können.

Derzeit gibt es drei Videoformate, die Sie kennen sollten, wenn Sie das video-Tag von HTML5 schon heute verwenden möchten: H.264, Theora und VP8.

Codecs und unterstützte Browser

H.264

[IE9, S4, C3, IOS]

Theora

[F3.5, C4, O10]

VP8

[IE9 (falls der Codec installiert ist), F4, C5, O10.7]

H.264

H.264 ist ein qualitativ hochwertiger Codec, der von der MPEG-Gruppe entwickelt und 2003 standardisiert wurde. Damit Low-End-Geräte wie Mobiltelefone und hochauflösende Geräte gleichermaßen unterstützt werden können, ist die H.264-Spezifikation in verschiedene Profile aufgeteilt. Diese Profile haben eine Reihe von Funktionen gemeinsam, die High-End-Profilen bieten jedoch zusätzliche Optionen für eine bessere Qualität. So können zum Beispiel iPhone und Flash Player gleichermaßen mit H.264 kodierte Videos wiedergeben. Jedoch unterstützt das iPhone nur das Profil „Baseline“ mit geringerer Qualität, während der Flash Player auch Streams mit höherer Qualität unterstützt. So ist es möglich, ein Video nur einmal zu kodieren, aber verschiedene Profile einzubetten, damit es auf mehreren Plattformen gut aussieht.

H.264 ist de facto der Standard, weil es von Microsoft und Apple als Lizenznehmer unterstützt wird. Zudem hat Googles YouTube seine Videos auf den H.264-Codec konvertiert, sodass diese auf dem iPhone ebenso wie mit dem Flash Player von Adobe wiedergegeben werden können. Allerdings handelt es sich nicht um eine offene Technologie. Sie ist patentiert, und die Verwendung unterliegt Lizenzbestimmungen. Inhaltsproduzenten müssen eine Lizenzgebühr bezahlen, um Videos mit H.264 zu kodieren. Aber diese Lizenzgebühren fallen nicht für Inhalte an, die Endbenutzern kostenlos zur Verfügung gestellt werden.²

² <http://www.reelseo.com/mpeg-la-announces-avc-h264-free-license-lifetime/>

Befürworter von freier Software haben Bedenken, dass die Rechteinhaber irgendwann einmal hohe Lizenzgebühren von Inhaltsproduzenten verlangen könnten. Diese Bedenken haben zur Entwicklung und Förderung alternativer Codecs geführt.

Theora

Theora ist ein gebührenfreier Codec, der von der Xiph.Org Foundation entwickelt wurde. Obwohl Inhaltsproduzenten Videos mit Theora in guter Qualität erstellen können, nehmen die Gerätehersteller diesen Codec nur langsam an. Firefox, Chrome und Opera können mit Theora kodierte Videos ohne zusätzliche Software auf jeder Plattform wiedergeben, aber der Internet Explorer, Safari und die iOS-Geräte tun dies nicht. Apple und Microsoft sind auf der Hut vor sogenannten „U-Boot-Patenten“: Patenten, bei deren Anmeldung die Veröffentlichung und Ausstellung des Patents vorsätzlich verzögert wird und sich der Inhaber bedeckt hält, während andere die Technologie implementieren. Zum richtigen Zeitpunkt taucht der Patentinhaber dann plötzlich auf und verlangt von einem ahnungslosen Markt Lizenzgebühren.

VP8

VP8 von Google ist vollständig Open Source und ein lizenzgebührenfreier Codec, der H.264 ziemlich ähnlich ist. Er wird von Mozilla, Google Chrome und Opera unterstützt. Der Microsoft Internet Explorer 9 soll VP8 ebenfalls unterstützen, wenn die Benutzer einen Codec installieren. VP8 wird außerdem auch von Adobes Flash Player unterstützt und ist daher eine interessante Alternative. Er wird allerdings nicht von Safari oder den iOS-Geräten unterstützt. Daher ist dieser Codec zwar kostenlos, Inhaltsproduzenten, die Videos für iPhone oder iPad bereitstellen möchten, müssen aber dennoch den H.264-Codec verwenden.

Audio-Codecs

Als wenn konkurrierende Videostandards nicht schon genug wären, müssen wir uns auch noch mit konkurrierenden Audiostandards herum-schlagen.

Codecs und unterstützte Browser

AAC

[S4, C3, IOS]

MP3

[IE9, S4, C3, IOS]

Vorbis (OGG)

[F3, C4, O10]

Advanced Audio Coding (AAC)

Das ist das Audioformat, das Apple in seinem iTunes Store verwendet. Es wurde entwickelt, um bei gleicher Dateigröße eine bessere Audioqualität als MP3 zu bieten, und unterstützt ähnlich wie H.264 mehrere Audioprofile. Und ebenso wie H.264 ist es kein freier Codec, daher fallen entsprechende Lizenzgebühren an.

Alle Apple-Produkte können AAC-Dateien wiedergeben, ebenso wie der Adobe Flash Player und der Open Source-Player VLC.

Vorbis (OGG)

Dieses lizenzgebührenfreie Open Source-Format wird von Firefox, Opera und Chrome unterstützt. Es wird teilweise auch zusammen mit den Video-Codecs Theora und VP8 verwendet. Vorbis-Dateien haben eine sehr gute Audioqualität, werden aber nur von wenigen Hardware-Musikplayern unterstützt.

MP3

Obwohl das MP3-Format weit verbreitet und beliebt ist, wird es von Firefox und Opera nicht unterstützt, weil es ebenfalls mit Patenten belegt ist. Safari und Google Chrome unterstützen MP3 trotzdem.

Video-Codecs und Audio-Codecs müssen für die Distribution und Wiedergabe zusammengepackt werden. Sprechen wir also über Video-Container.

Container und Codecs als Team

Ein Container ist eine Metadatei, die Audio- und Videodateien kennzeichnet und miteinander verzahnt. Ein Container enthält keine Informationen darüber, wie die enthaltenen Informationen kodiert sind. Im Grunde „verpackt“ ein Container Audio- und Video-Streams. Container können eine beliebige Kombination kodierter Medien enthalten. Wenn wir uns mit Video im Web beschäftigen, sehen wir uns die folgenden Kombinationen genauer an:

- OGG-Container mit Theora-Video und Vorbis-Audio, unterstützt von Firefox, Chrome und Opera
- MP4-Container mit H.264-Video und AAC-Audio, der in Safari und Chrome, mit dem Adobe Flash Player und auf iPhones, iPods und iPads wiedergegeben werden kann
- WebM-Container mit VP8-Video und Vorbis-Audio, der mit Firefox, Chrome, Opera und Adobe Flash Player funktioniert

Wenn wir davon ausgehen, dass Google und Mozilla bei VP8 und WebM bleiben, können wir Theora aus der Liste streichen. Es sieht fast so aus, als müssten wir unsere Videos zweimal kodieren – einmal für Apple-Benutzer (die einen kleinen Marktanteil bei den Desktop-Rechnern, aber einen großen Marktanteil bei den mobilen Geräten haben) und einmal für Benutzer mit Firefox und Opera, weil sich diese Browser weigern, H.264 wiederzugeben.³

So, das waren eine Menge Informationen. Nachdem Sie jetzt die Hintergründe und die Grenzen der verschiedenen Video-Codecs kennengelernt haben, machen wir uns an eine echte Implementierung. Wir beginnen mit Audio.

3 <http://lists.whatwg.org/pipermail/whatwg-whatwg.org/2009-June/020620.html>

15

Mit Audio arbeiten

AwesomeCo entwickelt eine Website, auf der lizenzfreie Audio-Loops für Screencasts vorgestellt werden, und möchte eine Demo-Seite mit einer einzelnen Loop-Sammlung sehen. Wenn wir damit fertig sind, soll auf der Seite eine Liste der Audio-Loops angezeigt werden, in die die Besucher schnell und einfach hineinhören können. Wir müssen uns nicht um die Audio-Loops für dieses Projekt kümmern, da uns der Toningenieur des Kunden bereits mit Mustern in den Formaten MP3 und OGG versorgt hat. In Anhang C auf Seite 257 finden Sie Informationen dazu, wie Sie Ihre eigenen Audiodateien kodieren.

Aufbau der grundlegenden Liste

Der Toningenieur hat uns vier Beispieldateien gegeben: drums, organ, bass und guitar. Wir müssen jede einzelne dieser Beispieldateien mit HTML-Markup beschreiben. So sieht das Markup für den Loop drums aus:

```
html5_audio/audio.html
```

```
<article class="sample">
  <header><h2>Drums</h2></header>
  <audio id="drums" controls>
    <source src="sounds/ogg/drums.ogg" type="audio/ogg">
    <source src="sounds/mp3/drums.mp3" type="audio/mpeg">
    <a href="sounds/mp3/drums.mp3">Download drums.mp3</a>
  </audio>
</article>
```

Wir definieren zuerst das Element audio und teilen ihn mit, dass wir Schaltflächen für die Steuerung haben möchten. Als Nächstes geben wir mehrere Quellen für die Datei an. Wir definieren zuerst die MP3- und OGG-Versionen der Datei und zeigen anschließend einen Link an, über den die Besucher die MP3-Datei direkt herunterladen können, falls der Browser das audio-Element nicht unterstützt.

Dieser sehr einfache Code funktioniert in Chrome, Safari und Firefox. Schreiben wir ihn also zusammen mit den drei anderen Soundbeispielen in eine HTML5-Vorlage.

html5_audio/audio.html

```

<article class="sample">
  <header><h2>Drums</h2></header>
  <audio id="drums" controls>
    <source src="sounds/ogg/drums.ogg" type="audio/ogg">
    <source src="sounds/mp3/drums.mp3" type="audio/mpeg">
    <a href="sounds/mp3/drums.mp3">Download drums.mp3</a>
  </audio>
</article>

<article class="sample">
  <header><h2>Guitar</h2></header>
  <audio id="guitar" controls>
    <source src="sounds/ogg/guitar.ogg" type="audio/ogg">
    <source src="sounds/mp3/guitar.mp3" type="audio/mpeg">
    <a href="sounds/mp3/guitar.mp3">Download guitar.mp3</a>
  </audio>
</article>

<article class="sample">
  <header><h2>Organ</h2></header>
  <audio id="organ" controls>
    <source src="sounds/ogg/organ.ogg" type="audio/ogg">
    <source src="sounds/mp3/organ.mp3" type="audio/mpeg">
    <a href="sounds/mp3/organ.mp3">Download organ.mp3</a>
  </audio>
</article>

<article class="sample">
  <header><h2>Bass</h2></header>
  <audio id="bass" controls>
    <source src="sounds/ogg/bass.ogg" type="audio/ogg">
    <source src="sounds/mp3/bass.mp3" type="audio/mpeg">
    <a href="sounds/mp3/bass.mp3">Download bass.mp3</a>
  </audio>
</article>
</body>
</html>

```

Wenn wir die Seite in einem HTML5-kompatiblen Browser öffnen, erhält jeder Eintrag in der Liste einen eigenen Audioplayer, wie Sie in Abbildung 7.1 auf der folgenden Seite sehen. Der Browser kümmert sich selbst um die Wiedergabe der Audio-Dateien, wenn Sie auf die Wiedergabeschaltfläche klicken.

Laden wir die Seite dagegen in Internet Explorer, werden die Links zum Herunterladen angezeigt, da der IE das audio-Element nicht versteht. Das ist eine praktikable Ausweichlösung. Wir schauen aber trotzdem mal, ob wir das nicht besser hinbekommen.

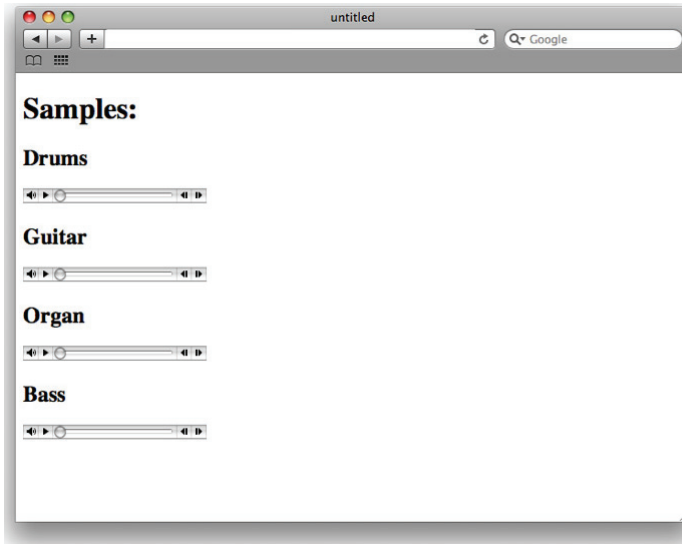


Abbildung 7.1: Unsere Seite in Safari

Ausweichlösung

Die alternative Audio-Unterstützung ist bereits im Element selbst enthalten. Wir haben mehrere Audio-Quellen über das `source`-Element angegeben und halten Links zum Herunterladen der Audiodateien bereit. Wenn der Browser das `audio`-Element nicht rendern kann, wird der Link angezeigt, den wir in das Feld platziert haben. Wir könnten sogar noch einen Schritt weitergehen und nach der Definition unserer Quellen Flash als Alternativlösung einsetzen.

Aber unter Umständen ist das nicht die beste Wahl. Sie könnten an einen Browser geraten, der zwar das `audio`-Element unterstützt, aber nicht die von Ihnen bereitgestellten Formate. Außerdem könnten Sie zu dem Schluss kommen, dass es sich für Sie zeitlich gar nicht lohnt, das Audiomaterial in mehreren Formaten anzubieten. Und die HTML5-Spezifikation sagt ausdrücklich, dass die Ausweichunterstützung nicht für Inhalte verwendet werden soll, die von Screenreadern vorgelesen werden.

Die einfachste Lösung besteht also darin, den Link zum Herunterladen aus dem `audio`-Element heraus zu schieben und mit JavaScript auszublenden:

```
html5_audio/advanced_audio.html
```

```
<article class="sample">
  <header><h2>Drums</h2></header>
  <audio id="drums" controls>
```

```

<source src="sounds/ogg/drums.ogg" type="audio/ogg">
<source src="sounds/mp3/drums.mp3" type="audio/mpeg">
</audio>
<a href="sounds/mp3/drums.mp3">Download drums.mp3</a>
</article>

```

Anschließend müssen wir lediglich ermitteln, ob das audio-Element unterstützt wird, und die Links verstecken. Hierzu erstellen wir mit JavaScript ein neues audio-Element und überprüfen, ob es auf die Methode `canPlayType()` reagiert:

```
html5_audio/advanced_audio.html
```

```
var canPlayAudioFiles = !! (document.createElement('audio').canPlayType);
```

Wir werten die Antwort aus und verstecken anschließend alle Anker, die in unsere Abschnitte mit den Beispieldateien verschachtelt sind.

```
html5_audio/advanced_audio.html
```

```

$(function(){
  var canPlayAudioFiles = !! (document.createElement('audio').canPlayType);

  if(canPlayAudioFiles){
    $(".sample a").hide();
  };
});

```

Die Ausweichlösungen für Audio sind relativ einfach. Und manche Benutzer wissen vielleicht auch die Möglichkeit zu schätzen, dass sie die Datei einfach herunterladen können.

Die native Wiedergabe von Audio im Browser ist allerdings nur der erste Schritt. Die Browser fangen gerade erst an, die HTML5-JavaScript-APIs für Audio und Video zu unterstützen. Das können Sie im Kasten auf Seite 146 nachlesen.

16

Video einbetten

AwesomeCo möchte eine neue Reihe mit Schulungsvideos auf seiner Website präsentieren. Die Videos sollen auf so vielen Geräten wie möglich darstellbar sein, insbesondere auf dem iPad. Zum Test wurden uns zwei Videos aus der Reihe „Photoshop-Tipps“ überlassen, mit denen wir den Prototyp erstellen werden. Glücklicherweise haben wir die Videodateien in den Formaten H.264, Theora und VP8 bekommen, sodass wir uns auf die Erstellung der Seite konzentrieren können.⁴

Das `video`-Tag funktioniert genauso wie das `audio`-Element. Wir müssen lediglich unsere Quellen angeben, und schon zeigen Chrome, Firefox, Safari, iPhone, iPad und der Internet Explorer 9 das Video ohne zusätzliche Plugins an. Das Markup für unsere erste Videodatei `01_blur` sieht folgendermaßen aus:

```
html5video/index.html
```

```
<article>
  <header>
    <h2>Saturate with Blur</h2>
  </header>
  <video controls>
    <source src="video/h264/01_blur.mp4">
    <source src="video/theora/01_blur.ogv">
    <source src="video/webm/01_blur.webm">
    <p>Your browser does not support the video tag.</p>
  </video>
</article>
```

Wir definieren das `video`-Tag mit `controls`. Außerdem legen wir implizit fest, dass das Video nicht automatisch wiedergegeben werden soll, indem wir das Attribut `autoplay` *nicht* angeben. Schon jetzt werden unsere Videos in einer Vielzahl von Browsern wiedergegeben. Unsere Benutzer sehen einen Videoplayer, der so ähnlich aussieht wie der in Abbildung 7.2 auf der nächsten Seite.

Wir erreichen damit aber keine Benutzer mit einem Internet Explorer der Version 8 oder älter. Das funktioniert nur mit Flash.

4 Wenn Sie mehr darüber erfahren möchten, wie Sie Ihre eigenen Video-Dateien kodieren, schauen Sie in Anhang C ab Seite 257 nach .

Saturate with Blur

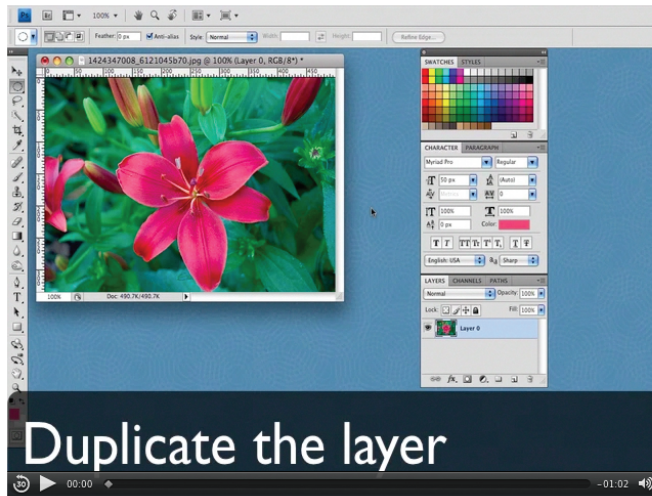


Abbildung 7.2: Unser Video mit dem HTML5-Videoplayer von Safari

Ausweichlösung

Um eine Flash-basierte Ausweichlösung anzubieten und dennoch HTML5-Video zu verwenden, platzieren wir den Code für das Flash-Objekt innerhalb des video-Tags. Die Website „Video For Everybody“⁵ erklärt das ganz genau. Wir nehmen hier aber nur eine einfache Implementierung vor.

Flowplayer⁶ ist ein Flash-basierter Player, der unser bereits codiertes H.264-Video abspielen kann. Wir laden die Open Source-Version des Players herunter und legen die Dateien `flowplayer-x.x.x.swf` und `flowplayer-controls-x.x.x.swf` in das Verzeichnis `swf` unseres Projekts, damit alles schön aufgeräumt bleibt.

Anschließend fügen wir den folgenden Code in unser video-Tag ein, direkt unterhalb unseres letzten source-Elements:

`html5video/index.html`

```
<object width="640" height="480" type="application/x-shockwave-flash"
  data="swf/flowplayer-3.2.2.swf">
  <param name="movie" value="swf/flowplayer-3.2.2.swf" />
  <param name="allowfullscreen" value="true" />
```

⁵ <http://videoforeverybody.com>

⁶ <http://flowplayer.org/download/index.html>

```

    <param name="flashvars"
      value='config={"clip":{"url":"../video/h264/01_blur.mp4",
                           "autoPlay":false,
                           "autoBuffering":true
                        }}' />
    
  </object>

```

Passen Sie in diesem Teil ganz genau auf:

html5video/index.html

```

<param name="flashvars"
value='config={"clip":{"url":"../video/h264/01_blur.mp4",
                       "autoPlay":false,
                       "autoBuffering":true
                    }}' />

```

Die Quelle der Videodatei muss *relativ* zum Speicherort von Flowplayer angegeben werden. Nachdem wir Flowplayer im Verzeichnis *swf* abgelegt haben, müssen wir den Pfad `../video/h264/01_blur.mp4` angeben, damit der Player unser Video findet.

Wenn wir jetzt unsere Seite im Internet Explorer aufrufen, wird unser Video abgespielt – dank Flowplayer auch ohne, dass wir es in einem *zusätzlichen* Format kodieren mussten. Unsere Internet Explorer-Freunde bekommen eine Ansicht wie in Abbildung 7.3 auf der folgenden Seite angezeigt.

Natürlich müssen wir immer noch eine Möglichkeit für Menschen finden, die nicht über native Videounterstützung verfügen *und* kein Flash installiert haben. In diesem Fall lassen wir die Besucher unsere Videoinhalte herunterladen, indem wir einen weiteren Abschnitt mit Links zum Herunterladen einfügen.

html5video/index.html

```

<section class="downloads">
  <header>
    <h3>Downloads</h3>
  </header>
  <ul>
    <li><a href="video/h264/01_blur.mp4">H264, playable on most platforms</a></li>
    <li><a href="video/theora/01_blur.ogv">OGG format</a></li>
    <li><a href="video/webm/01_blur.webm">WebM format</a></li>
  </ul>
</section>

```

Saturate with Blur

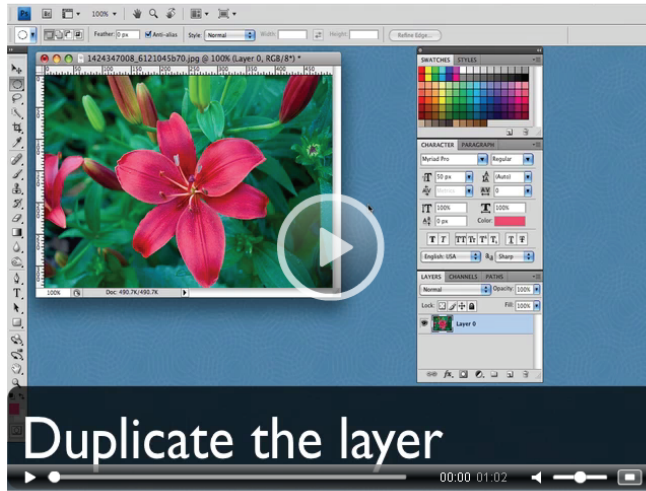


Abbildung 7.3: Unser Video im Internet Explorer mit Flowplayer

Falls HTML5-Video nicht unterstützt wird, können wir mit JavaScript die Videos folgendermaßen ausblenden:

```
function canPlayVideo() {
    return !!document.createElement('video').canPlayType;
}
if(canPlayVideo()){
    $('#videos .downloads').hide();
}
```

Hier verwenden wir einen ähnlichen Prüfmechanismus wie im Abschnitt *Mit Audio arbeiten* auf Seite 138. In unserem Fall macht es aber mehr Sinn, wenn die Benutzer die Videos herunterladen, damit sie sie später auf ihren iPods oder iPads ansehen können.

Die Grenzen von HTML5-Video

Derzeit ist die Nutzung von HTML5-Video durch drei äußerst wichtige Einschränkungen begrenzt.

Zum einen gibt es mit HTML5-Video keine Möglichkeit, die Videodateien zu streamen. Die Benutzer haben sich daran gewöhnt, dass sie zu einem bestimmten Teil eines Videos springen können. Das ist etwas, das Flash-basierte Videoplayer hervorragend beherrschen, weil Adobe eine Menge Aufwand in Flash als Video-Plattform gesteckt hat. In HTML5-Videos kann man dagegen nur navigieren, wenn die Datei vollständig in den Browser heruntergeladen wurde.

JavaScript-API für Medieninhalte

In diesem Kapitel haben wir uns kurz mit den JavaScript-APIs für die Elemente `audio` und `video` beschäftigt. Mit der API können Sie ermitteln, welche Arten von Audiodateien vom Browser unterstützt werden, und die Wiedergabe der `audio`-Elemente steuern.

Im Abschnitt *Mit Audio arbeiten* auf Seite 138 haben wir eine Seite mit mehreren Klangbeispielen erstellt. Hier sehen Sie eine simple Lösung, wie wir mit der JavaScript-API alle Klänge (ungefähr) zur gleichen Zeit abspielen können:

`html5_audio/advanced_audio.html`

```
var element = $("<p><input type='button' value='Play all' /></p>")
element.click(function(){
    $("audio").each(function(){
        this.play();
    })
});

$("body").append(element);
```

Wir erstellen eine Schaltfläche „Play all“. Nach einem Klick auf diese Schaltfläche werden alle `audio`-Elemente auf der Seite durchlaufen und wird jeweils die `play()`-Methode aufgerufen.

Mit Videos können wir ähnlich verfahren. Es gibt Methoden, um Elemente zu starten, anzuhalten und sogar die aktuelle Spielzeit abzufragen.

Leider wurde die JavaScript-API noch nicht auf allen Plattformen vollständig unterstützt, als dieses Buch geschrieben wurde. Das soll Sie aber nicht davon abhalten, die in der Spezifikation^a beschriebenen Möglichkeiten auszuloten.

^a <http://www.w3.org/TR/html5/video.html#media-elements>

Die Unterhaltungsindustrie für Erwachsene und der technische Fortschritt

Die Unterhaltungsindustrie für Erwachsene hat einen starken Einfluss auf die Technologien im Internet, von E-Commerce bis hin zum Erfolg von Flash.^a Denselben Einfluss wird sie auch auf HTML5-Video haben.^b Geräte wie das iPhone und iPad bieten mehr Privatsphäre als Desktops und Laptops, unterstützen aber kein Flash. Viele Websites für Erwachsene haben aus diesem Grund bei der Bereitstellung von Video bereits von Flash auf HTML5 mit H.264 umgestellt. Interessanterweise stört es die Anbieter anscheinend nicht, dass HTML5-Video derzeit keine Rechteverwaltung bietet.

Die Unterhaltungsindustrie für Erwachsene scheut sich niemals, Gelegenheiten beim Schopf zu packen. Und unter Umständen können wir durch dieses Interesse an neuen Technologien spannende Fortschritte bei HTML5-Video miterleben.

^a <http://chicagopressrelease.com/news/in-tech-world-porn-quietly-leads-the-way>

^b <http://news.avn.com/articles/Joone-Points-to-HTML-5-as-Future-of-Web-Content-Delivery-401434.html>

Außerdem gibt es keine Möglichkeiten zur Rechteverwaltung. Websites wie Hulu⁷, die die Piraterie ihrer Inhalte verhindern möchten, können HTML5-Video nicht einsetzen. Flash bleibt also in solchen Situationen auch weiterhin die bevorzugte Lösung.

Schließlich bleibt vor allem zu bedenken, dass der Vorgang der Video-kodierung zeit- und kostenintensiv ist. Die Notwendigkeit, Videos in mehreren Formaten zu kodieren, mindert die Attraktivität von HTML5-Video erheblich. Aus diesem Grund gibt es viele Websites, die Video im patentgeschützten H.264 anbieten, damit es mit einer Kombination aus dem HTML5-video-Tag und Flash auch auf iPods und iPads wiedergegeben werden kann.

HTML5 wird an diesen Schwierigkeiten zwar nicht scheitern. Wir müssen sie aber berücksichtigen, bevor wir Flash bei der Bereitstellung von Video durch HTML5-Video ersetzen können.

⁷ <http://www.hulu.com>

Audio, Video und Barrierefreiheit

Keine der Ausweichlösungen funktioniert wirklich gut für Menschen mit Behinderungen. Die HTML5-Spezifikation weist darauf explizit hin. Hörbehinderte Benutzer haben nichts davon, wenn sie die Audiodatei herunterladen können. Und Sehbehinderte erhalten keinen Vorteil durch Videodateien, die sie außerhalb des Browsers abspielen können. Wenn wir unseren Benutzern Inhalte anbieten, sollten wir immer brauchbare Alternativen bereithalten, soweit das möglich ist: zum Beispiel Transkriptionen von Video- und Audiodateien, die Ihre Besucher anzeigen können. Wenn Sie Ihre Inhalte selbst produzieren, können Sie das von Anfang an berücksichtigen und die Transkriptionen einfach aus dem ursprünglichen Skript anfertigen. Ist eine Transkription nicht möglich, sollten Sie wenigstens eine Zusammenfassung der wichtigsten Teile des Videos anbieten.

```
html5video/index.html
```

```
<section class="transcript">
  <h2>Transcript</h2>
  <p>We'll drag the existing layer to the new button on the bottom
    of the Layers palette to create a new copy.</p>
  <p>Next we'll go to the Filter menu and choose "Gaussian Blur".
    We'll change the blur amount just enough so that we lose a
    little bit of the detail of the image.</p>
  <p>Now we'll double-click on the layer to edit the layer and
    change the blending mode to "Overlay". We can then adjust the
    amount of the effect by changing the opacity slider.</p>
  <p>Now we have a slightly enhanced image.</p>
</section>
```

Sie können die Transkriptionen ausblenden oder von der Video-Seite aus darauf verlinken. Solange sie leicht zu finden sind, stellen sie eine echte Hilfe dar.

Die Zukunft

Erstklassige Audiounterstützung im Browser eröffnet den Entwicklern tonnenweise neue Möglichkeiten. JavaScript-Webanwendungen können auf einfache Weise Effekte und Alarmsignale auslösen, ohne dass die Audiodateien mit Flash eingebettet werden müssen. Durch die native Videounterstützung können Videos auch auf Geräten wie dem iPhone zur Verfügung gestellt werden. Außerdem erhalten wir dadurch eine offene standardisierte Methode, über JavaScript mit Videos zu interagieren. Am wichtigsten aber ist, dass wir Video- und Audioclips einfach wie Bilder behandeln können, indem wir sie semantisch mit Markup beschreiben und damit einfacher erkennen können.

Kapitel 8

Augenschmaus

Als Webentwickler sind wir immer auf der Suche nach Möglichkeiten, unsere Oberflächen mehr ins Auge stechen zu lassen. Und CSS3 bietet eine ganze Reihe Optionen dafür. Wir können eigene Schriften auf unseren Seiten verwenden. Wir können Elemente mit abgerundeten Ecken und Schlagschatten erstellen. Wir können Farbverläufe als Hintergrund verwenden. Wir können sogar Elemente rotieren, damit nicht immer alles so eckig und langweilig aussieht. Und das alles ohne Photoshop oder andere Grafikprogramme. In diesem Kapitel zeigen wir Ihnen, wie das geht. Wir beginnen damit, ein Formular etwas weicher zu gestalten, indem wir ein paar Ecken abrunden. Anschließend konstruieren wir den Prototyp eines Banners für eine Messe. Dabei lernen wir die Arbeit mit Schatten, Drehungen, Verläufen und Transparenz kennen. Zum Abschluss erklären wir, wie wir mit der CSS3-Funktion `@font-face` hübschere Schriften für den Unternehmensblog verwenden können.

Im Einzelnen erforschen wir in diesem Kapitel folgende CSS3-Funktionen:¹

`border-radius [border-radius:10px;]`

Rundet die Ecken von Elementen ab. [C4, F3, IE9, S3.2, O10.5]

`RGBA-Unterstützung [background-color: rgba(255,0,0,0.5);]`

Verwendet RGB-Farben und Transparenz statt Hexadezimalcodes. [C4, F3.5, IE9, S3.2, O10.1]

¹ In den folgenden Beschreibungen wird die Unterstützung durch die verschiedenen Browser in eckigen Klammern mit einem Kurzcode und der mindestens erforderlichen Versionsnummer angegeben. Die verwendeten Codes lauten: C: Google Chrome, F: Firefox, IE: Internet Explorer, O: Opera, S: Safari, IOS: iOS-Geräte mit Mobile Safari und A: Android-Browser.

`box-shadow [box-shadow: 10px 10px 5px #333;]`

Erstellt Schlagschatten für Elemente.

[C3, F3.5, IE9, S3.2, O10.5]

`Rotation [transform: rotate(7.5deg);]`

Rotiert beliebige Elemente.

[C3, F3.5, IE9, S3.2, O10.5]

`Verläufe [linear-gradient(top, #fff, #efefef);]`

Erstellt Verläufe, die als Bilder verwendet werden können.

[C4, F3.5, S4]

`@font-face [@font-face{font-family:AwesomeFont;]`

`src: url(http://example.com/awesomeco.ttf);font-weight:bold;}]`

Ermöglicht die Verwendung spezieller Schriftarten über CSS.

[C4, F3.5, IE5+, S3.2, O10.1]

17

Scharfe Ecken abrunden

Im Web ist standardmäßig alles eckig. Formularfelder, Tabellen und sogar die Abschnitte von Webseiten sehen klobig und kantig aus. Deshalb haben viele Designer im Laufe der Jahre auf die verschiedensten Techniken zurückgegriffen, um Elementen runde Ecken zu verpassen und so die Oberflächen ein bisschen weicher zu gestalten.

CSS3 bietet eine einfache Möglichkeit, Ecken abzurunden, die von Firefox und Safari bereits seit langer Zeit unterstützt wird. Leider ist der Internet Explorer noch nicht mit an Bord. Aber das können wir leicht ändern.

Ein Anmeldeformular auflockern

Die Modelle und Entwürfe für Ihr aktuelles Projekt zeigen Formularfelder mit abgerundeten Ecken. Zunächst runden wir die Ecken nur mit CSS3 ab. Das Ergebnis soll so aussehen wie in Abbildung 8.1 auf der nächsten Seite.

Für das Anmeldeformular schreiben wir ganz einfaches HTML.

```
css3roughedges/rounded_corners.html
```

```
<form action="/login" method="post">
  <fieldset id="login">
    <legend>Log in</legend>
    <ol>
      <li>
        <label for="email">Email</label>
        <input id="email" type="email" name="email">
      </li>
      <li>
        <label for="password">Password</label>
        <input id="password" type="password"
          name="password" value="" autocomplete="off"/>
      </li>
      <li><input type="submit" value="Log in"></li>
    </ol>
  </fieldset>
</form>
```

Log in

Email

user@example.com

Password

8-10 characters

Log in

Abbildung 8.1: Unser Formular mit runden Ecken

Wir stylen das Formular noch ein bisschen, damit es besser aussieht.

css3roughedges/style.css

```
fieldset{
  width: 216px;
  border: none;
  background-color: #ddd;
}

fieldset legend{
  background-color: #ddd;
  padding: 0 64px 0 2px;
}

fieldset>ol{list-style: none;
  padding:0;
  margin: 2px;
}

fieldset>ol>li{
  margin: 0 0 9px 0;
  padding: 0;
}

/* Jedes Eingabefeld erhält eine eigene Zeile */
fieldset input{
  display:block;
}

input{
  width: 200px;
  background-color: #fff;
  border: 1px solid #bbb;
}

input[type="submit"]{
  width: 202px;
  padding: 0;
  background-color: #bbb;
}
```

Diese einfachen Stilregeln entfernen die Aufzählungszeichen von der Liste und gewährleisten, dass alle Eingabefelder dieselbe Größe haben. Nachdem wir damit fertig sind, können wir unsere Elemente abrunden.

Browserspezifische Selektoren

Da die CSS3-Spezifikation noch nicht endgültig ist, haben die Browserhersteller selbst einige Funktionen hinzugefügt und den Namen jeweils das Präfix ihrer eigenen Implementierungen vorangestellt. Durch diese Präfixe können Browserhersteller frühzeitig Funktionen einführen, bevor diese Teil einer endgültigen Spezifikation werden. Und da sie nicht der eigentlichen Spezifikation folgen, können die Browserhersteller die tatsächliche Spezifikation und ihre eigene Version parallel implementieren. In den meisten Fällen entspricht die Version mit dem Hersteller-Präfix der CSS-Spezifikation, aber gelegentlich auch nicht. Für Sie bedeutet das leider, dass Sie den Rahmenradius für jeden Browsertyp gesondert deklarieren müssen.

Firefox verwendet diesen Selektor:

```
css3roughedges/style.css
```

```
-moz-border-radius: 5px;
```

WebKit-basierte Browser wie Safari und Chrome nutzen den folgenden Selektor:

```
css3roughedges/style.css
```

```
-webkit-border-radius: 5px;
```

Um alle Eingabefelder unseres Formulars abzurunden, benötigen wir eine CSS-Regel wie die folgende:

```
css3roughedges/style.css
```

```
input, fieldset, legend{
  border-radius: 5px;
  -moz-border-radius: 5px;
  -webkit-border-radius: 5px;
}
```

Fügen Sie das in Ihre Datei `style.css` ein, und die Ecken sind rund.

Ausweidlösung

Nun funktioniert alles in Firefox, Safari und Google Chrome. Aber wie Sie wissen, funktioniert es nicht im Internet Explorer. Und natürlich wissen Sie, dass es auch im Internet Explorer funktionieren muss und Sie daher etwas implementieren müssen, das dem so nahe wie möglich kommt.

Webentwickler runden Ecken nun schon seit geraumer Zeit mit Hintergrundbildern und anderen Techniken ab, aber wir möchten es möglichst einfach halten. Wir können mit JavaScript den Radius der Ecken ermitteln und eine ganze Reihe von Abrundungstechniken anwenden. In diesem Beispiel verwenden wir das jQuery-Plugin „Corner“ sowie eine Abwandlung des Corner-Plugins, die auch Textfelder abrundet.

Unterstützung für abgerundete Ecken ermitteln

Unsere Ausweidlösung sieht der im Abschnitt *Ausweidlösung* auf Seite 82 sehr ähnlich. Wir binden die jQuery-Bibliothek und das Plugin ein und überprüfen, ob der Browser das Attribut unterstützt. Falls nicht, aktivieren wir das Plugin. In diesem Fall müssen wir das Vorhandensein der CSS-Eigenschaft `border-radius`, aber auch browserspezifischer Präfixe wie `webkit` und `moz` ermitteln.

Erstellen Sie `corner.js`, und fügen Sie die folgende Funktion ein:

`css3roughedges/corner.js`

```
function hasBorderRadius(){
  var element = document.documentElement;
  var style = element.style;
  if (style){
    return typeof style.borderRadius == "string" ||
      typeof style.MozBorderRadius == "string" ||
      typeof style.WebkitBorderRadius == "string" ||
      typeof style.KhtmlBorderRadius == "string";
  }
  return null;
}
```

Nun können wir überprüfen, ob es bei unserem Browser an Unterstützung für abgerundete Ecken hapert. Schreiben wir also den Code für das eigentliche Abrunden der Ecken. Erfreulicherweise gibt es ein Plugin, das wir als Ausgangspunkt verwenden können.

jQuery Corners

jQuery Corners² ist ein kleines Plugin, das Ecken dadurch abrundet, dass die jeweiligen Elemente in zusätzliche div-Tags verpackt und so gestylt werden, dass das Zielelement abgerundet aussieht. Es funktioniert aber nicht mit Formularfeldern. Mit ein bisschen Fantasie und ein bisschen jQuery können wir dieses Plugin aber dazu bringen, dass es auch das tut.

Entscheiden, ob sich der Aufwand lohnt

In unserem Beispiel möchte der Kunde, dass die abgerundeten Ecken wirklich in allen Browsern funktionieren. Allerdings sollten Sie solche Funktionen nach Möglichkeit immer optional halten. Natürlich argumentieren manche Leute, dass es einen realen Nutzen bringt, wenn das Aussehen des Formulars aufgelockert wird. Trotzdem sollten Sie sich zunächst einen Eindruck davon verschaffen, wie viele Menschen überhaupt Browser verwenden, die die CSS-basierte Abrundung nicht unterstützen. Wenn Ihre Besucher hauptsächlich Safari und Firefox benutzen, lohnt sich unter Umständen der zeitliche Aufwand zum Schreiben und Pflegen eines Skripts zur Überprüfung und für die Auswechlösung nicht.

Als Erstes verweisen Sie von Ihrer HTML-Seite auf jQuery Corners. Bei dieser Gelegenheit binden Sie auch gleich noch die Datei `corner.js` mit ein.

```
css3roughedges/rounded_corners.html
```

```
<script src="jquery.corner.js" charset="utf-8" type='text/javascript'></script>
<script src="corner.js" charset="utf-8" type='text/javascript'></script>
```

Nun müssen wir lediglich den Code schreiben, der die Abrundung aufruft.

Unser formCorners-Plugin

Wir schreiben ein jQuery-Plugin, damit wir die Funktion zum Abrunden einfach auf alle Formularfelder anwenden können. Die Erstellung von jQuery-Plugins haben wir bereits im Abschnitt *Auswechlösung* auf Seite 60 besprochen und müssen sie daher nicht noch einmal erklären.

² <http://www.malsup.com/jquery/corner/>

Gehen Sie einfach den Code für das Plugin Schritt für Schritt durch (es baut zum Teil auf einer Lösung von Tony Amoyal³ auf).

Fügen Sie Folgendes in die Datei `corners.js` ein:

```
css3roughedges/corner.js
```

```
(function($){
    $.fn.formCorner = function(){
        return this.each(function() {
            var input = $(this);
            var input_background = input.css("background-color");
            var input_border = input.css("border-color");
            input.css("border", "none");
            var wrap_width = parseInt(input.css("width")) + 4;
            var wrapper = input.wrap("<div></div>").parent();
            var border = wrapper.wrap("<div></div>").parent();
            wrapper.css("background-color", input_background)
                .css("padding", "1px");
            border.css("background-color", input_border)
                .css("width", wrap_width + "px")
                .css('padding', '1px');
            wrapper.corner("round 5px");
            border.corner("round 5px");
        });
    };
})(jQuery);
```

Wir nehmen ein jQuery-Objekt, das entweder ein Element oder eine Sammlung von Elementen sein kann, und verpacken es in zwei `div`-Tags, die wir anschließend abrunden. Zuerst geben wir dem inneren `div` dieselbe Farbe wie dem Hintergrund des ursprünglichen Eingabefelds und deaktivieren den Rahmen des eigentlichen Formularfelds. Anschließend verschachteln wir das Feld in ein anderes Feld, das die Rahmenfarbe des ursprünglichen Eingabefelds als Hintergrundfarbe sowie ein bisschen Padding erhält. Und dieses Padding ergibt die Rahmenlinie. Stellen Sie sich einfach zwei Blätter Buntpapier vor: ein grünes, das 10 cm breit ist, und ein rotes, das 8 cm breit ist. Wenn Sie das kleinere auf das größere legen, sehen Sie um das rote Stück Papier einen grünen Rahmen herum. So funktioniert das.

3 <http://www.tonyamoyal.com/2009/06/23/text-inputs-with-rounded-corners-using-jquery-without-image/>

Die Abrundung aufrufen

Mit dem Plugin und unserer Prüfbibliothek können wir nun die Abrundung starten.

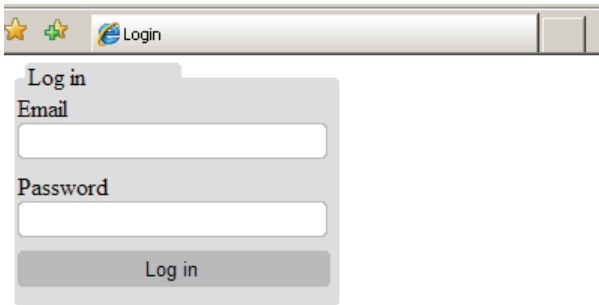


Abbildung 8.2: Unsere Formulare haben nun auch im Internet Explorer runde Ecken.

Fügen Sie Folgendes in die Datei `corners.js` ein:

`css3roughedges/corner.js`

```

1 $(function(){
2   if(!hasBorderRadius()){
3     $("input").formCorner();
4     $("fieldset").corner("round 5px");
5     $("legend").corner("round top 5px cc:#fff");
6   }
7 });

```

Wir runden die drei Formularfelder und die Feldgruppe ab. In Zeile 5 runden wir schließlich den oberen Teil der Legende ab und legen fest, dass für den Ausschnitt der Ecken Weiß verwendet werden soll. Das Plugin verwendet die Hintergrundfarbe des übergeordneten Elements als Farbe für den Ausschnitt, aber das ist hier nicht richtig.

Wenn der Browser die Eigenschaft `border-radius` unterstützt, führt er unser Plugin aus. Falls nicht, verwendet er das CSS, das wir vorhin eingefügt haben.

Ein kleiner IE-Trick

Der IE behandelt Legenden ein bisschen anders. Wir können einen Trick einsetzen, durch den der IE die Legende der Feldgruppe ein bisschen nach oben schiebt, sodass sie genauso aussieht wie in Firefox und Chrome.

`css3roughedges/rounded_corners.html`

```
<link rel="stylesheet" href="style.css" type="text/css" media="screen">
<!--[if IE]>
  <style>
    fieldset legend{margin-top: -10px }
  </style>
<![endif]-->
```

Jetzt sieht unser Formular auf allen wichtigen Browsern relativ ähnlich aus. Die Internet Explorer-Version sehen Sie in Abbildung 8.2 auf der vorherigen Seite.

Abgerundete Ecken lassen Ihre Oberflächen weicher erscheinen und sind extrem einfach zu verwenden. Sie sollten diese Technik aber wie jede andere Gestaltungsmöglichkeit mit Bedacht und nicht übermäßig verwenden.

18

Schatten, Verläufe und Transformationen

Abgerundete Ecken genießen natürlich eine Menge Ansehen, aber das ist nur der Anfang. Wir können noch viel mehr mit CSS3 machen: Elemente mit Schlagschatten vom restlichen Inhalt abheben, Hintergründe mit Verläufen definierter aussehen lassen und Elemente mit Transformationen rotieren. Kombinieren wir einige dieser Techniken, um ein Banner für die in Kürze stattfindende AwesomeConf zusammenzubasteln – eine Messe und Konferenz, die AwesomeCo jedes Jahr veranstaltet. Der Grafikdesigner hat ein PSD geschickt, das wie die Abbildung 8.3 aussieht. Wir können das Namensschild, den Schatten und sogar die Transparenz vollständig in CSS erstellen. Vom Grafikdesigner brauchen wir lediglich das Hintergrundbild mit den Menschen.

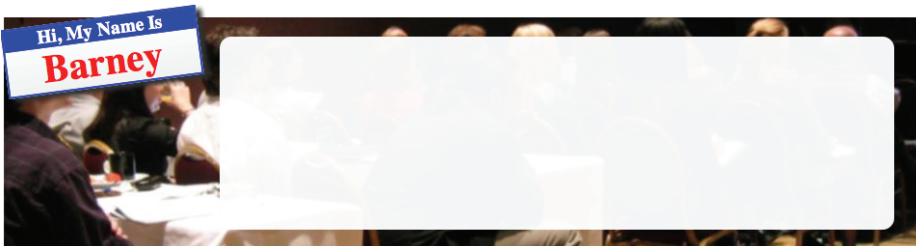


Abbildung 8.3: Das ursprüngliche Konzept, das wir mit CSS3 neu erfinden

Die grundlegende Struktur

Beginnen wir mit der grundlegenden HTML-Struktur der Seite:

```
css3banner/index.html
```

```
<div id="conference">
  <section id="badge">
    <h3>Hi, My Name Is</h3>
    <h2>Barney</h2>
  </section>

  <section id="info">
  </section>
</div>
```

Diesen Code stylen wir folgendermaßen:

`css3banner/style.css`

```
#conference{
  background-color: #000;
  width: 960px;
  float:left;
  background-image: url('images/awesomeconf.jpg');
  background-position: center;
  height: 240px;
}
#badge{
  text-align: center;
  width: 200px;
  border: 2px solid blue;
}
#info{
  margin: 20px;
  padding: 20px;
  width: 660px;
  height: 160px;
}
#badge, #info{
  float: left;
  background-color: #fff;
}

#badge h2{
  margin: 0;
  color: red;
  font-size: 40px;
}

#badge h3{
  margin: 0;
  background-color: blue;
  color: #fff;
}
```

Sobald wir dieses Stylesheet auf unsere Seite anwenden, werden unser Namensschild und der Inhaltsbereich wie in Abbildung 8.4 nebeneinander angezeigt. Jetzt kann es mit dem Styling des Namensschilds losgehen.



Abbildung 8.4: Die einfache Version unseres Banners

Einen Verlauf hinzufügen

Dem Ansteckschild können wir mehr Struktur verpassen, indem wir den weißen Hintergrund durch einen leichten Verlauf von Weiß nach Hellgrau ersetzen. Verläufe funktionieren in Firefox, Safari und Chrome. Aber die Implementierung in Firefox ist anders. Chrome und Safari verwenden die WebKit-Syntax, den ursprünglichen Vorschlag. Für Firefox benötigen wir eine Syntax, die dem W3C-Vorschlag ähnelt. Wieder einmal verwenden wir die Browser-Präfixe, die Sie im Abschnitt *Browserspezifische Selektoren* auf Seite 153 kennengelernt haben.⁴

css3banner/style.css

```
#badge{
  background-image: -moz-linear-gradient(
    top, #fff, #efefef
  );

  background-image: -webkit-gradient(
    linear, left top, left bottom,
    color-stop(0, #fff),
    color-stop(1, #efefef)
  );

  background-image: linear-gradient(
    top, #fff, #efefef
  );
}
```

Firefox verwendet die Methode `-moz-linear-gradient`, mit der wir den Anfangspunkt des Verlaufs, gefolgt von der Startfarbe und der Zielfarbe, angeben.

Für WebKit-basierte Browser können wir Farbstopps angeben. In unserem Beispiel soll der Verlauf von Weiß bis Grau gehen. Für zusätzliche Farben müssen wir lediglich einen weiteren Farbstopp definieren.

Schatten für das Ansteckschild

Wir können es leicht so aussehen lassen, als ob das Ansteckschild oberhalb des Banners liegt, indem wir einen Schlagschatten hinzufügen. Üblicherweise würden wir den Schatten in Photoshop zum Bild hinzu- oder als Hintergrundbild einfügen. In CSS3 können wir aber einfach mit der Eigenschaft `box-shadow` einen Schatten für Elemente definieren.⁵

4 <http://dev.w3.org/csswg/css3-images/#linear-gradients>

5 <http://www.w3.org/TR/css3-background/#the-box-shadow>

Für den Schatten des Namensschilds brauchen wir nur die folgende Regel in unser Stylesheet einzufügen:

```
css3banner/style.css
```

```
#badge{
  -moz-box-shadow: 5px 5px 5px #333;
  -webkit-box-shadow: 5px 5px 5px #333;
  -o-box-shadow: 5px 5px 5px #333;
  box-shadow: 5px 5px 5px #333;
}
```

Die Eigenschaft `box-shadow` erwartet vier Parameter. Der erste ist der horizontale Versatz: Eine positive Zahl bedeutet, dass der Schatten zur rechten Seite des Objekts fällt. Bei einer negativen Zahl fällt der Schatten zur linken Seite. Der zweite Parameter gibt den vertikalen Versatz an: Positive Zahlen lassen den Schatten unterhalb der Box erscheinen, negative Zahlen oberhalb.

Der dritte Parameter gibt den Weichzeichnungsradius an. Beim Wert 0 wird der Schatten sehr scharf dargestellt, bei höheren Werten wird der Schatten entsprechend weichgezeichnet. Der letzte Parameter bestimmt die Farbe des Schattens.

Mit diesen Werten müssen Sie experimentieren, bis Sie ein Gefühl dafür entwickeln, wie sie funktionieren, und die passenden Werte finden. Für die Arbeit mit Schatten sollten Sie erforschen, wie Schatten in der physischen Welt funktionieren.

Textschatten

Zusätzlich zu Elementen können Sie auch Text ganz einfach Schatten werfen lassen. Das funktioniert genauso wie `box-shadow`.

```
h1{text-shadow: 2px 2px 2px #bbbbbb;}
```

Sie geben den X- und Y-Versatz an sowie den Grad der Weichzeichnung und die Farbe des Schattens. IE 6, 7 und 8 bieten mit dem Filter „Shadow“ dieselbe Möglichkeit.

```
filter: Shadow(Color=bbbbbb,
  Direction=135,
  Strength=3);
```

Das ist derselbe Ansatz wie beim Schlagschatten für Elemente. Schatten ergänzen Texte um einen hübschen Effekt, aber zu starke Schatten schaden der Lesbarkeit.

Schnappen Sie sich eine Taschenlampe, und beleuchten Sie Objekte damit, oder gehen Sie vor die Tür, und beobachten Sie, wie die Sonne Gegenstände Schatten werfen lässt. Diese Perspektive ist sehr wichtig. Inkonsistente Schatten lassen Ihre Oberflächen schnell verwirrend erscheinen, insbesondere dann, wenn Sie für mehrere Elemente die Schatten falsch zeichnen. Die einfachste Variante besteht darin, für jeden Schatten dieselben Einstellungen zu verwenden.

Das Namensschild drehen

Mit CSS3-Transformationen können Sie Elemente genauso wie mit den Vektorgrafikprogrammen Flash, Illustrator oder Inkscape drehen, skalieren und verzerren.⁶ Elemente heben sich dadurch deutlicher ab, und die Webseiten sehen nicht ganz so „kastenförmig“ aus. Wir drehen das Namensschild ein bisschen, damit es aus der geraden Kante des Banners ausbricht.

`css3banner/style.css`

```
#badge{
  -moz-transform: rotate(-7.5deg);
  -o-transform: rotate(-7.5deg);
  -webkit-transform: rotate(-7.5deg);
  -ms-transform: rotate(-7.5deg);
  transform: rotate(-7.5deg);
}
```

Drehungen sind mit CSS3 ziemlich einfach. Wir müssen lediglich den Drehwinkel angeben, und los geht es. Alle Elemente innerhalb des rotierten Elements werden mitgedreht.

Die Drehung ist genauso simpel wie die abgerundeten Ecken, aber auch damit sollten Sie es nicht übertreiben. Das Ziel beim Interface-Design besteht darin, Oberflächen benutzerfreundlicher zu machen. Wenn Sie Elemente mit viel Inhalt drehen, sollten Sie sicherstellen, dass Ihre Benutzer den Inhalt auch lesen können, ohne den Kopf zu weit zur Seite legen zu müssen!

Transparente Hintergründe

Grafikdesigner verwenden schon seit einer ganzen Weile semitransparente Ebenen hinter Text. Dafür erstellen sie entweder ein komplettes Bild im Photoshop oder legen mit CSS ein transparentes PNG über ein

6 <http://www.w3.org/TR/css3-2d-transforms/#transform-property>

Element. In CSS3 können wir Hintergrundfarben mit einer neuen Syntax definieren, die auch Transparenz unterstützt.

Als Neuling in der Webentwicklung lernen Sie zunächst, Farben mit hexadezimalen Farbcodes zu definieren: Sie geben die Menge an Rot, Grün und Blau in Zahlenpaaren. 00 bedeutet „ganz aus“, und FF bedeutet „ganz an“. FF0000 entspricht der Farbe Rot und bedeutet „Rot ganz an, Blau ganz aus und Grün ganz aus“.

In CSS3 werden die Funktionen `rgb` und `rgba` eingeführt. Die `rgb`-Funktion funktioniert wie ihr hexadezimaler Gegenstück, jedoch verwenden Sie Werte zwischen 0 und 255 für die jeweiligen Farben. Rot wird beispielsweise als `rgb(255,0,0)` definiert.

Die `rgba`-Funktion arbeitet auf die gleiche Weise wie die `rgb`-Funktion, erwartet aber einen vierten Parameter, der die Deckkraft zwischen 0 und 1 angibt. Für den Wert 0 wird überhaupt keine Farbe angezeigt, die Farbe ist vollkommen transparent. Mit der folgenden Stilregel machen wir den weißen Kasten semitransparent:

css3banner/style.css

```
#info{
  background-color: rgba(255,255,255,0.95);
}
```

Wenn Sie mit solchen Transparenzwerten arbeiten, können sich die Kontrasteinstellungen der Benutzer auf das Endergebnis auswirken. Sie sollten unbedingt auf verschiedenen Bildschirmen mit den Werten experimentieren, um konsistente Ergebnisse zu gewährleisten.

Während wir am Infobereich unseres Banners arbeiten, können wir auch gleich die Ecken abrunden:

css3banner/style.css

```
#info{
  moz-border-radius: 12px;
  webkit-border-radius: 12px;
  o-border-radius: 12px;
  border-radius: 12px;
}
```

Damit sieht unser Banner in Safari, Firefox und Chrome ziemlich gut aus. Als Nächstes implementieren wir ein Stylesheet für den Internet Explorer.

Ausweichlösung

Die in diesem Abschnitt verwendeten Techniken funktionieren wunderbar im IE 9, sind aber auch mit Internet Explorer 6, 7 und 8 möglich! Dafür müssen wir die DirectX-Filter von Microsoft verwenden. Das bedeutet, dass wir auf einen bedingten Kommentar zurückgreifen und ein IE-spezifisches Stylesheet laden müssen. Außerdem müssen wir mit JavaScript das `section`-Element erstellen, damit wir es mit CSS stylen können, da diese Versionen des IE das Element nicht nativ erkennen.

css3banner/index.html

```
<!--[if lte IE 8]>

<script>
document.createElement("section");
</script>

<link rel="stylesheet" href="ie.css" type="text/css" media="screen">

<![endif]-->
</head>
<body>
  <div id="conference">
    <section id="badge">
      <h3>Hi, My Name Is</h3>
      <h2>Barney</h2>
    </section>

    <section id="info">
      </section>
    </div>

  </body>
</html>
```

Die DirectX-Filter funktionieren in IE 6, 7 und 8. Aber im IE 8 werden die Filter anders aufgerufen. Daher müssen Sie jeden Filter zweimal deklarieren. Sehen wir uns zunächst an, wie wir Elemente drehen können.

Drehen

Wir können Elemente mit diesen Filtern drehen. Aber es reicht nicht aus, einfach nur den Winkel vorzugeben. Für den gewünschten Effekt müssen wir den Matrixfilter verwenden und den Cosinus bzw. Sinus des gewünschten Winkels angeben. Genauer gesagt müssen wir den Cosinus, den negativen Sinus, den Sinus und nochmals den Cosinus angeben:⁷

7 Wir machen eine lineare Transformation mit einer 2x2-Matrix.

css3banner/filters.css

```

filter: progid:DXImageTransform.Microsoft.Matrix(
    sizingMethod='auto expand',
    M11=0.9914448613738104,
    M12=0.13052619222005157,
    M21=-0.13052619222005157,
    M22=0.9914448613738104
);

-ms-filter: "progid:DXImageTransform.Microsoft.Matrix(
    sizingMethod='auto expand',
    M11=0.9914448613738104,
    M12=0.13052619222005157,
    M21=-0.13052619222005157,
    M22=0.9914448613738104
)";

```

Kompliziert? Ja, vor allem, wenn Sie sich das vorherige Beispiel genauer ansehen: Erinnern Sie sich, dass unser ursprünglicher Winkel *minus* 7,5 Grad beträgt? Entsprechend brauchen wir für den *negativen* Sinus einen positiven Wert, und unser Sinus erhält einen *negativen* Wert.

Mathe ist kompliziert. Machen wir lieber einen Verlauf.

Farbverläufe

Der Verlaufsfiler des IE funktioniert wie im Standard vorgesehen, allerdings haben Sie eine ganze Menge mehr Tipparbeit. Eigentlich geben Sie nur die Start- und die Endfarbe an – und schon wird der Verlauf angezeigt.

css3banner/filters.css

```

filter: progid:DXImageTransform.Microsoft.gradient(
    startColorStr=#FFFFFF, endColorStr=#EFEFEF
);
-ms-filter: "progid:DXImageTransform.Microsoft.gradient(
    startColorStr=#FFFFFF, endColorStr=#EFEFEF
)";

```

Anders als bei den anderen Browsern wenden Sie im IE den Verlauf direkt auf das Element an statt auf die Eigenschaft background-image.

Wir wenden den Filter gleich noch einmal für die transparente Hintergrundfarbe unseres Infobereichs an.

Transparenz

Sie können dem Verlaufsfilter erweiterte Hexadezimalwerte für die Start- und Endfarbe übergeben, wobei die ersten beiden Stellen den Transparenzgrad angeben:

css3banner/filters.css

```
background: none;
filter:
  progid:DXImageTransform.Microsoft.gradient(
    startColorStr=#BBFFFFFF, endColorStr=#BBFFFFFF
  );

-ms-filter: "progid:DXImageTransform.Microsoft.gradient(
  startColorStr='#BBFFFFFF', EndColorStr='#BBFFFFFF'
)";
```

Die achtstelligen Hexcodes funktionieren ähnlich wie die rgba-Funktion, jedoch steht der Transparenzwert an *erster* Stelle, nicht am Ende. In Wahrheit haben wir es also mit Alpha, Rot, Grün und Blau zu tun.

Wir müssen die Hintergrundeinstellungen des Elements entfernen, damit das im IE 7 funktioniert. Falls Sie das Stylesheet mittippen, haben Sie sicherlich festgestellt, dass es noch nicht funktioniert. Aber das können wir ändern.

Alles zusammengekommen

Eines der schwierigeren Probleme mit den IE-Filtern besteht darin, dass wir sie nicht stückchenweise definieren können. Um mehrere Filter auf ein einzelnes Element anzuwenden, müssen wir die Filter als mit Kommata getrennte Liste definieren. So sieht das tatsächliche IE-Stylesheet aus:

css3banner/ie.css

```
#info{
  background: none;
  filter:
    progid:DXImageTransform.Microsoft.gradient(
      startColorStr=#BBFFFFFF, endColorStr=#BBFFFFFF
    );
  -ms-filter: "progid:DXImageTransform.Microsoft.gradient(
    startColorStr='#BBFFFFFF', EndColorStr='#BBFFFFFF'
  )";
}

#badge{
  filter:
    progid:DXImageTransform.Microsoft.Matrix(
      sizingMethod='auto expand',
      M11=0.9914448613738104,
      M12=0.13052619222005157,
```

```

        M21=-0.13052619222005157,
        M22=0.9914448613738104
    ),
    progid:DXImageTransform.Microsoft.gradient(
        startColorStr=#FFFFFF, endColorStr=#EFEFEF
    ),
    progid:DXImageTransform.Microsoft.Shadow(
        color=#333333, Direction=135, Strength=3
    );

-ms-filter: "progid:DXImageTransform.Microsoft.Matrix(
    sizingMethod='auto expand',
    M11=0.9914448613738104,
    M12=0.13052619222005157,
    M21=-0.13052619222005157,
    M22=0.9914448613738104
),
progid:DXImageTransform.Microsoft.gradient(
    startColorStr=#FFFFFF, endColorStr=#EFEFEF
),
progid:DXImageTransform.Microsoft.Shadow(
    color=#333333, Direction=135, Strength=3
)";
}

```

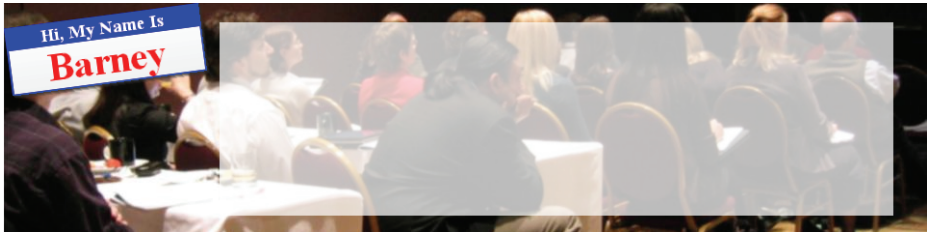


Abbildung 8.5: Unser Banner im Internet Explorer 8

Das ist eine Menge Code für das gewünschte Ergebnis, beweist aber, dass es möglich ist, diese Funktionen zu verwenden. Wenn Sie einen Blick auf Abbildung 8.5 werfen, sehen Sie, dass wir unserem Ziel ziemlich nahe kommen. Jetzt müssen wir nur noch die Ecken des info-Abschnitts abrunden. Wie das geht, können Sie im Abschnitt *Scharfe Ecken abrunden* auf Seite 151 nachlesen.

Auch wenn diese Filter unhandlich und ein bisschen seltsam sind, sollten Sie sie in eigenen Projekten weiter erforschen, weil Sie dadurch IE-Benutzern ein ähnliches Ergebnis bieten können.

Denken Sie daran: Die Effekte, die wir in diesem Abschnitt untersucht haben, betreffen nur die Präsentation. Beim ursprünglichen Stylesheet haben wir sorgfältig Hintergrundfarben ausgesucht, mit denen der Text gut lesbar ist. Auch Browser, die kein CSS3 verstehen, können die Seite also auf lesbare Art und Weise darstellen.

19

Echte Schriften nutzen

Typographie ist für den Eindruck, den die Benutzer von Ihrer Website bekommen, entscheidend. Für das Buch, das Sie gerade lesen, wurden die Schriften sorgfältig von Menschen ausgesucht, die genau wissen, dass die richtige Auswahl von Schriften und der richtige Leerraum ein Buch besser lesbar machen. Diese Konzepte sind für das Web genauso wichtig.

Die Schriftarten, mit denen wir Lesern unsere Inhalte vermitteln, haben einen Einfluss darauf, wie unsere Inhalte aufgenommen werden. Die folgende Schrift ist die perfekte Wahl für eine laute Heavy Metal-Band:

Aber für den Einband dieses Buchs ist die Schrift eher weniger geeignet:

Wie Sie sehen, ist es wirklich wichtig, dass Sie eine Schrift wählen, die zum Inhalt passt. Das Problem mit Schriften im Web ist allerdings, dass Webentwickler bisher auf eine Handvoll Schriften beschränkt waren, die sogenannten „websicheren“ Schriften. Das sind Schriften, die auf den Betriebssystemen der meisten Benutzer vorhanden sind.

Um diese Einschränkung zu umgehen, haben wir bisher Grafiken mit den gewünschten Schriften direkt in das Markup eingefügt oder zu anderen Methoden gegriffen, wie etwa zu CSS-Hintergrundbildern oder zu sIFR,⁸ der Schriften über Flash rendert. Das Schriftmodul von CSS3 bietet da einen wesentlich eleganteren Ansatz.

8 <http://www.mikeindustries.com/blog/sifr>

@font-face

Die Anweisung `@font-face` wurde eigentlich als Teil der CSS2-Spezifikation eingeführt und bereits in den Internet Explorer 5 implementiert. Allerdings verwendete die Implementierung von das Schriftformat „Embedded Open Type“ (EOT), während die meisten Schriften im TrueType- oder OpenType-Format vorliegen. Andere Browser unterstützen aktuell Schriften im OpenType- und TrueType-Format.

Schriften und Rechte

Manche Schriften sind nicht kostenlos. Genau wie bei Bildarchiven und anderem urheberrechtlich geschützten Material wird von Ihnen erwartet, dass Sie die Rechte und Lizenzen des Materials einhalten, das Sie auf Ihrer Website verwenden. Wenn Sie eine Schrift kaufen, sind Sie normalerweise berechtigt, die Schrift in Ihrem Logo und den Bildern auf Ihren Seiten zu verwenden. Das nennt man *Nutzungsrechte*. Allerdings bringt `@font-face` eine andere Art von Lizenz ins Spiel – das Recht zur Weiterverbreitung.

Wenn Sie eine Schrift in Ihre Seite einbetten, müssen die Benutzer diese Schrift herunterladen. Das bedeutet, dass Sie diese Schrift über Ihre Website an andere weiterverbreiten. Sie müssen sich daher absolut sicher sein, dass diese Art der Nutzung für die verwendeten Schriften auch zulässig ist.

Typekit^a bietet eine große Bibliothek an lizenzierten Schriften sowie Tools und Code, mit dem Sie die Schriften auf einfache Art in Ihre Website integrieren können. Der Service ist nicht kostenlos, aber bezahlbar, wenn Sie eine bestimmte Schrift verwenden möchten.

Google bietet die Google Font API^b, die ähnlich wie Typekit funktioniert, aber nur Open Source-Schriften bereitstellt.

Beide Dienste laden die Schriften über JavaScript. Daher müssen Sie sicherstellen, dass Ihr Inhalt auch für Benutzer ohne JavaScript leicht lesbar ist.

Solange Sie mit Schriften nicht anders umgehen als mit anderen Wirtschaftsgütern, sollten Sie keine Schwierigkeiten bekommen..

^a <http://www.typekit.com/>

^b <http://code.google.com/apis/webfonts/>

Der Marketingdirektor von AwesomeCo hat entschieden, dass sich das Unternehmen für eine bestimmte Schrift als Standard sowohl für den Druck als auch für das Web entscheiden soll. Sie wurden beauftragt, eine Schrift mit dem Namen Garogier zu recherchieren – eine einfache, schlanke Schrift, die auch für die kommerzielle Nutzung absolut kostenlos ist. Testweise verwenden wir diese Schrift für das Blog-Beispiel aus dem Abschnitt *Einen Blog mit semantischem Markup neu definieren* von Seite 26. Auf diese Weise kann jeder die Schrift in Aktion erleben.

\\ Joe fragt ...

Wie konvertiere ich meine eigenen Schriften?

Falls Sie eine eigene Schrift entwickelt oder die Rechte für eine Schrift erworben haben und diese in mehreren Formaten zur Verfügung stellen müssen, bietet die Website „Font Squirrel“ einen Konverter^a, der sowohl die konvertierten Schriften als auch ein Stylesheet mit dem erforderlichen Code für @font-face liefert. Vergewissern Sie sich aber, ob die Lizenz für Ihre Schrift diese Art der Nutzung zulässt.

a <http://www.fontsquirrel.com/fontface/generator>

Schriftformate

Schriften sind in einer Vielzahl von Formaten erhältlich. Letztlich entscheiden die Browser, die Sie erreichen möchten, über das benötigte Format.

Formate und unterstützte Browser

Embedded OpenType (EOT) [IE5–8]

TrueType (TTF) [IE9, F3.5, C4, S4]

OpenType (OTF) [IE9, F3.5, C4, S4, O10.5]

Scalable Vector Graphics (SVG) [IOS]

Web Open Font Format (WOFF) [IE9, F3.6]

Der Internet Explorer unterstützt bis einschließlich Version 8 nur das Format *Embedded OpenType* (EOT). Andere Browser unterstützen außerdem die gebräuchlicheren TrueType- und OpenType-Schriften.

Microsoft, Opera und Mozilla haben gemeinsam das Format *Web Open Font* entwickelt, das die verlustfreie Kompression und bessere Lizenzoptionen für Schriftdesigner ermöglicht.

Um alle Browser zu erreichen, müssen Sie Ihre Schriften in mehreren Formaten bereitstellen.

Schrift ersetzen

Die Schrift, nach der wir suchen, ist bei Font Squirrel⁹ in den Formaten TrueType, WOFF, SVG und EOT erhältlich. Das ist perfekt.

Die Nutzung der Schrift erfolgt in zwei Schritten: Zunächst definieren Sie die Schrift und verknüpfen sie dann mit den jeweiligen Elementen. Fügen Sie folgenden Code in das Stylesheet für den Blog ein:

css3fonts/style.css

```
@font-face {
  font-family: 'GarogierRegular';
  src: url('fonts/Garogier_unhinted-webfont.eot');
  src: url('fonts/Garogier_unhinted-webfont.woff') format('woff'),
        url('fonts/Garogier_unhinted-webfont.ttf') format('truetype'),
        url('fonts/Garogier_unhinted-webfont.svg#webfontew0qE009')
        format('svg');
  font-weight: normal;
}
```

Wir definieren zunächst die Schriftfamilie, geben ihr einen Namen und geben dann die Quellen für die Schrift an. Die Version für *Embedded Open Type* schreiben wir an erster Stelle, damit der IE sie auch gleich findet. Anschließend geben wir die anderen Quellen an. Der Browser des Benutzers probiert einfach eine Quelle nach der anderen aus, bis er eine findet, die funktioniert.

Nachdem wir die Schriftfamilie definiert haben, können wir sie in unserem Stylesheet verwenden. So ändern wir unseren ursprünglichen Schriftstil:

css3fonts/style.css

```
body{
  font-family: "GarogierRegular";
}
```

Durch diese einfache Änderung wird der Text unserer Seite wie in Abbildung 8.6 auf der nächsten Seite mit der neuen Schrift angezeigt.

9 Sie erhalten sie unter <http://www.fontsquirrel.com/fonts/Garogier> auch im Code zu diesem Buch, den Sie ebenfalls herunterladen können.

Die Verwendung von Schriften ist in modernen Browsern relativ einfach. Wir müssen aber auch Browser berücksichtigen, die das noch nicht unterstützen.



Abbildung 8.6: Der Blog mit neuer Schrift

Ausweidlösung

Wir haben bereits Ausweidlösungen für verschiedene Versionen des IE und anderer Browser bereitgestellt. Aber wir müssen auch gewährleisten, dass unsere Seiten in Browsern lesbar sind, die die Funktion @font-face nicht unterstützen.

Wir stellen zwar Alternativversionen der Schrift Garogier bereit. Wir haben aber keine anderen Schriftalternativen angegeben. Das bedeutet, dass der Browser die Standardschrift verwendet, wenn er unsere Garogier-Schrift nicht unterstützt. Das könnte sich als ungünstig erweisen.

Font Stacks sind Listen von Schriften, die in der Reihenfolge ihrer Priorität geordnet sind. Zuerst geben Sie die Schrift an, die Sie *wirklich* verwenden möchten, und danach geeignete Alternativen.

Nehmen Sie sich bei der Erstellung eines Font Stacks ausreichend Zeit, um wirklich passende Alternativschriften anzugeben. Zeichenabstand, Strichstärke und das allgemeine Aussehen sollten ähnlich sein. Auf der Website *Unit Interactive* steht ein ausgezeichnete englischer Artikel dazu.¹⁰

¹⁰ <http://unitinteractive.com/blog/2008/06/26/better-css-font-stacks/>

Ändern wir unsere Schrift folgendermaßen:

```
css3fonts/style.css
```

```
font-family: "Garogier Regular", Georgia,  
            "Palatino", "Palatino Linotype",  
            "Times", "Times New Roman", serif;
```

Wir geben eine ganze Reihe an Alternativen an, wodurch wir eine ähnliche Darstellung erreichen sollten. Das ist nicht in jedem Fall perfekt, aber sicherlich besser als die Standardschrift, die manchmal schwer zu lesen ist.

Schriften können viel dazu beitragen, Ihre Seite attraktiver und leichter lesbar zu machen. Experimentieren Sie ein bisschen damit. Es warten eine Menge Schriften auf Sie – kostenlose und kommerzielle.

Die Zukunft

In diesem Kapitel haben wir einige Möglichkeiten untersucht, mit CSS3 traditionelle Techniken zur Webentwicklung abzulösen. Aber wir haben nur an der Oberfläche gekratzt. Die CSS3-Spezifikation spricht von 3D-Transformationen und sogar von einfachen Animationen. Das heißt, dass wir unseren Benutzern statt mit JavaScript auch mit Stylesheets interaktive Hinweise geben können, ähnlich wie mit `:hover`.

Außerdem unterstützen einige Browser bereits mehrfache Hintergrundbilder und Verlaufsrahmen. Halten Sie außerdem Ausschau nach Verbesserungen für seitenweise organisierte Inhalte, wie etwa fortlaufende Kopf- und Fußzeilen sowie Unterstützung für Seitenzahlen.

Wenn die CSS3-Module einmal fertig sind, werden wir es viel einfacher haben, vielseitigere, bessere und einladendere Elemente für unsere Benutzeroberflächen zu erstellen. Achten Sie also darauf, ob neue Funktionen veröffentlicht werden.

Teil III

Jenseits von HTML5

Kapitel 9

Mit clientseitigen Daten arbeiten

Wir haben bisher nur über HTML5- und CSS3-Markup gesprochen, aber nun richten wir unsere Aufmerksamkeit auf einige der mit HTML5 verwandten Technologien und Funktionen. Mit dokumentübergreifenden Benachrichtigungen und Offline-Unterstützung können wir beispielsweise über Domains hinweg kommunizieren und Lösungen erstellen, mit denen unsere Benutzer auch offline arbeiten können.

Manche Funktionen wie Web Storage, Web SQL Databases und Web Sockets wurden aus der HTML5-Spezifikation ausgegliedert. Andere, wie etwa Geolocation, waren nie Teil der Spezifikation, wurden aber von Browserherstellern und Entwicklern mit HTML5 in Verbindung gebracht, weil die Spezifikation parallel zu anderen Funktionalitäten implementiert wird.

Diese Funktionen werden in diesem Teil des Buchs behandelt, wobei der Schwerpunkt auf jenen liegt, die bereits jetzt verwendbar sind. In Kapitel 11 sprechen wir darüber, was als Nächstes kommt. Wir beginnen vorerst damit, uns Web Storage und Web SQL Storage anzusehen, zwei Spezifikationen mit denen wir Daten auf dem Client ablegen können.

Erinnern Sie sich noch an die Zeit, als Cookies toll waren? Ich auch nicht. Cookies waren von Anfang an eher unangenehm, seit sie plötzlich auftauchten. Aber wir haben uns den ganzen Stress angetan, weil sie die einzige Möglichkeit waren, Informationen auf dem Clientrechner zu speichern. Um sie zu verwenden, mussten wir den Cookies einen Namen geben und ein Verfallsdatum festlegen.

Dazu ist eine Portion JavaScript-Code nötig, den wir in eine Funktion verpackt haben, um nie darüber nachdenken zu müssen, wie er eigentlich funktioniert. Ungefähr so:

`html5_localstorage/setcookie.js`

```
// via http://www.javascripter.net/faq/settinga.htm
function SetCookie(cookieName, cookieValue, nDays) {
    var today = new Date();
    var expire = new Date();
    if (nDays==null || nDays==0) nDays=1;
    expire.setTime(today.getTime() + 3600000 * 24 * nDays);
    document.cookie = cookieName+"=" +escape(cookieValue)
        + ";expires=" +expire.toGMTString();
}
```

Abgesehen von der schwer zu merkenden Syntax gibt es auch Sicherheitsbedenken. Manche Websites tracken mit Cookies das Surfverhalten ihrer Besucher, die dann ihrerseits die Cookies deaktivieren, um das zu verhindern.

In HTML5 wurden einige neue Optionen zum Speichern von Daten auf dem Client eingeführt: Web Storage (`localStorage` oder `sessionStorage`)¹ und Web SQL Databases.² Sie sind einfach zu verwenden, unglaublich leistungsfähig und einigermaßen sicher. Das Beste daran: Sie sind heutzutage in mehreren Browsern implementiert, z.B. in Mobile Safari unter iOS und im Webbrowser von Android 2.0. Allerdings sind sie nicht mehr Bestandteil der HTML5-Spezifikation – sie wurden in eigene Spezifikationen ausgegliedert.

`localStorage`, `sessionStorage` und Web SQL Databases können zwar keine Cookies ersetzen, die von Client und Server gemeinsam genutzt werden – wie im Fall von Web-Frameworks, die Cookies einsetzen, um Zustände zwischen den einzelnen Anfragen zu speichern. Sie können aber für die Speicherung von Daten verwendet werden, die nur für den Benutzer wichtig sind, wie etwa Ansichtseinstellungen oder andere Einstellungen. Sie sind auch für die Entwicklung von mobilen Anwendungen praktisch, die auch ohne Internetverbindung im Browser ausgeführt werden können. Viele aktuelle Webanwendungen rufen den Server auf, um Benutzerdaten zu speichern, aber mit diesen neuen Speichermechanismen besteht keine Notwendigkeit mehr für eine Internetverbindung. Benutzerdaten können lokal gespeichert und bei Bedarf gesichert werden.

1 <http://www.whatwg.org/specs/web-apps/2007-10-26/#storage>

2 <http://www.whatwg.org/specs/web-apps/2007-10-26/#sql>

Wenn Sie diese Methoden mit den neuen Offline-Funktionen von HTML5 kombinieren, können Sie direkt im Browser vollwertige Datenbankanwendungen erstellen, die auf einer Vielzahl von Plattformen laufen – von Desktop-Rechnern bis hin zu iPads und Android-Telefonen. In diesem Kapitel lernen Sie, wie Sie mit diesen Techniken Benutzereinstellungen ablegen und eine einfache Notizdatenbank erstellen können.

In diesem Kapitel lernen wir die folgenden Funktionen kennen:³

localStorage

Speichert Daten in Schlüssel-/Wertpaaren, die mit einer Domain verknüpft sind und über Browsersitzungen hinweg erhalten bleiben.

[C5, F3.5, S4, IE8, O10.5, IOS, A]

sessionStorage

Speichert Daten in Schlüssel-/Wertpaaren, die mit einer Domain verknüpft sind und am Ende einer Browsersitzung gelöscht werden.

[C5, F3.5, S4, IE8, O10.5, IOS, A]

Web SQL Databases

Vollständig relationale Datenbanken mit Unterstützung für das Erstellen von Tabellen und das Einfügen, Aktualisieren, Löschen und Auswählen von Daten mit Transaktionen. Sie sind mit einer Domain verknüpft und bleiben über Sitzungen hinweg erhalten.

[C5, S3.2, O10.5, IOS3.2, A2]

Offline-Webanwendungen

Definieren Dateien, die für die Offline-Verwendung im Cache zwischengespeichert werden sollen, damit Anwendungen auch ohne eine Internetverbindung ausgeführt werden können.

[C4, S4, F3.5, O10.6, IOS3.2, A2]

³ In den folgenden Beschreibungen wird die Unterstützung durch die verschiedenen Browser in eckigen Klammern mit einem Kurzcode und der mindestens erforderlichen Versionsnummer angegeben. Die verwendeten Codes lauten: C: Google Chrome, F: Firefox, IE: Internet Explorer, O: Opera, S: Safari, IOS: iOS-Geräte mit Mobile Safari und A: Android-Browser.

20

Einstellungen mit localStorage speichern

Der Mechanismus `localStorage` bietet Entwicklern eine sehr einfache Möglichkeit, Daten auf dem Clientrechner abzulegen. `localStorage` ist ein in den Webbrowser integrierter Speicher für Name-/Wertpaare.

Mit `localStorage` gespeicherte Informationen bleiben über die Browser-sitzungen hinweg erhalten und können nicht von anderen Websites gelesen werden, weil sie auf die aktuell besuchte Domain beschränkt sind.⁴

AwesomeCo ist gerade dabei, ein neues Kundenservice-Portal zu entwickeln und möchte, dass die Benutzer Textgröße, Hintergrund und Textfarbe der Website ändern können. Wir implementieren das mit `localStorage`, sodass die Änderungen von einer Browsersitzung zur nächsten erhalten bleiben, wenn wir sie abspeichern. Wenn wir damit fertig sind, sieht unser Prototyp so aus wie in Abbildung 9.1.

Formular für die Einstellungen

Wir erstellen ein Formular mit ein wenig semantischem HTML5-Markup und einigen neuen Formularsteuerelementen, die Sie in Kapitel 3, *Benutzerfreundliche Webformulare*, auf Seite 45 kennengelernt haben. Wir möchten, dass die Benutzer die Vordergrundfarbe, die Hintergrundfarbe sowie die Schriftgröße anpassen können.

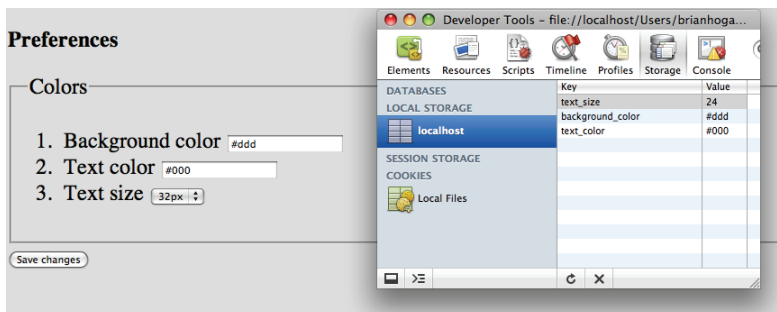


Abbildung 9.1: Die Werte für die Benutzereinstellungen werden lokal über `localStorage` gespeichert.

⁴ Passen Sie auf, wenn Sie lokal entwickeln. Wenn Sie beispielsweise auf `localhost` arbeiten, können Sie schnell Ihre Variablen durcheinanderbringen!

html5_localstorage/index.html

```

<p><strong>Preferences</strong></p>
<form id="preferences" action="save_prefs"
      method="post" accept-charset="utf-8">
  <fieldset id="colors" class="">
    <legend>Colors</legend>
    <ol>
      <li>
        <label for="background_color">Background color</label>
        <input type="color" name="background_color"
              value="" id="background_color">
      </li>
      <li>
        <label for="text_color">Text color</label>
        <input type="color" name="text_color"
              value="" id="text_color">
      </li>
      <li>
        <label for="text_size">Text size</label>
        <select name="text_size" id="text_size">
          <option value="16">16px</option>
          <option value="20">20px</option>
          <option value="24">24px</option>
          <option value="32">32px</option>
        </select>
      </li>
    </ol>
  </fieldset>
  <input type="submit" value="Save changes">
</form>

```

Für die Farben verwenden wir HTML-Farbcodes.

Einstellungen speichern und laden

Für die Arbeit mit dem localStorage-System greifen Sie mit JavaScript auf das Objekt `window.localStorage()` zu. So einfach können Sie ein Name-/Wertpaar festlegen:

html5_localstorage/index.html

```
localStorage.setItem("background_color", $("#background_color").val());
```

Genauso einfach ist es, einen Wert wieder zurückzulesen:

html5_localstorage/index.html

```
var bgcolor = localStorage.getItem("background_color");
```

Erstellen wir eine Methode zum Speichern aller Einstellungen des Formulars:

html5_localstorage/index.html

```
function save_settings(){
    localStorage.setItem("background_color", $("#background_color").val());
    localStorage.setItem("text_color", $("#text_color").val());
    localStorage.setItem("text_size", $("#text_size").val());
    apply_preferences_to_page();
}
```

Als Nächstes erstellen wir eine ähnliche Methode, die die Daten aus dem local-Storage-System lädt und in die Formularfelder platziert:

html5_localstorage/index.html

```
function load_settings(){
    var bgcolor = localStorage.getItem("background_color");
    var text_color = localStorage.getItem("text_color");
    var text_size = localStorage.getItem("text_size");

    $("#background_color").val(bgcolor);
    $("#text_color").val(text_color);
    $("#text_size").val(text_size);

    apply_preferences_to_page();
}
```

Diese Methode ruft außerdem eine Methode auf, die diese Einstellungen auf die Seite selbst anwendet. Diese schreiben wir als Nächstes.

Einstellungen anwenden

Nachdem wir jetzt die Einstellungen aus dem localStorage abrufen können, müssen wir sie auf die Seite anwenden. Die Einstellungen, mit denen wir arbeiten, haben alle irgendwie mit CSS zu tun. Daher können wir jQuery verwenden, um die Stilregeln der Elemente anzupassen

html5_localstorage/index.html

```
function apply_preferences_to_page(){
    $("body").css("background-color", $("#background_color").val());
    $("body").css("color", $("#text_color").val());
    $("body").css("font-size", $("#text_size").val() + "px");
}
```

Zum Schluss müssen wir noch dafür sorgen, dass unsere Funktion aufgerufen wird, sobald das Dokument dazu bereit ist.

```
html5_localstorage/index.html
```

```
$(function(){
    load_settings();

    $('form#preferences').submit(function(event){
        event.preventDefault();
        save_settings();
    });
});
```

Ausweichlösung

Die Methode `localStorage` funktioniert nur mit den neuesten Versionen von Internet Explorer, Firefox, Chrome und Safari. Also brauchen wir eine alternative Lösung für ältere Browser. Es gibt eine ganze Reihe von Möglichkeiten. Wir können die Informationen auf dem Server speichern oder clientseitig mit Cookies ablegen.

Serverseitige Speicherung

Wenn Ihr System über Benutzerkonten verfügt, besteht eine Möglichkeit darin, die Daten der Einstellungsseite im Benutzerdatensatz Ihrer Anwendung abzulegen. Wenn Sie die Benutzer anmelden, können Sie prüfen, ob clientseitige Einstellungen vorhanden sind. Falls nein, laden Sie sie vom Server. Auf diese Weise bleiben die Benutzereinstellungen unabhängig vom Browser und vom Computer erhalten.

Um die Daten auf dem Server zu halten, müssen Sie nur dafür sorgen, dass Ihr Formular einen Post an den Server durchführt – Sie dürfen also nicht das standardmäßige Übermittlungsverhalten mit JavaScript verhindern, wenn es keine Unterstützung für Cookies gibt.

Die serverseitige Speicherung ist die einzige Methode, die auch funktioniert, wenn die Benutzer JavaScript deaktivieren. Denn so können Sie die Einstellungen aus der Datenbank abrufen statt aus dem `localStorage`-Hash. Außerdem ist das die einzige Möglichkeit, mit der Sie mehr als 4 KB Daten speichern können – also mehr als die maximale Datenmenge, die Sie in einem Cookie speichern können.

Cookies und JavaScript

Die altbewährte Kombination von Cookies und JavaScript ist hier die passende Ausweichlösung. Mit dem allseits bekannten Cookie-Skript von Quirksmode⁵ können wir eine eigene localStorage-Ausweichlösung erstellen.

Wir können ziemlich einfach überprüfen, ob localStorage im Browser unterstützt wird. Wir prüfen einfach, ob das window-Objekt über die localStorage-Methode verfügt:

html5_localstorage/index.html

```
if (!window.localStorage){
}
```

Als Nächstes brauchen wir Methoden zum Schreiben der Cookies, die wir uns aus dem Quirksmode-Artikel leihen. Fügen Sie die folgenden JavaScript-Funktionen innerhalb der geschweiften Klammern in Ihren Skriptblock ein:

html5_localstorage/index.html

```
function createCookie(name,value,days) {
    if (days) {
        var date = new Date();
        date.setTime(date.getTime()+(days*24*60*60*1000));
        var expires = "; expires="+date.toGMTString();
    }
    else var expires = "";
    document.cookie = name+"="+value+expires+"; path=/";
}

function readCookie(name) {
    var result = ""
    var nameEQ = name + "=";
    var ca = document.cookie.split(';');
    for(var i=0;i < ca.length;i++) {
        var c = ca[i];
        while (c.charAt(0)==' ') c = c.substring(1,c.length);
        if (c.indexOf(nameEQ) == 0){
            result = c.substring(nameEQ.length,c.length);
        }else{
            result = "";
        }
    }
    return(result);
}
```

⁵ <http://www.quirksmode.org/js/cookies.html>

Zu guter Letzt müssen wir noch ein localStorage-Objekt erstellen, das die Cookies im Backend verwendet. Ein ziemlich rudimentäres Beispiel, mit dem das gerade so funktioniert, könnte folgendermaßen aussehen:

html5_localstorage/index.html

```

1  localStorage = (function () {
2      return {
3          setItem: function (key, value) {
4              createCookie(key, value, 3000)
5          },
6          getItem: function (key) {
7              return(readCookie(key));
8          }
9      };
10 })();

```

Achten Sie auf Zeile 4. Wir erstellen ein Cookie mit einem Verfallsdatum von 3.000 Tagen von jetzt an. Wir können keine Cookies erstellen, die niemals verfallen. Also lege ich den Wert auf einen Zeitpunkt fest, der lächerlich weit in der Zukunft liegt.

Von außen betrachtet haben wir die grundlegende Implementierung von localStorage beibehalten. Wenn Sie Elemente entfernen oder alles löschen müssen, müssen Sie ein bisschen kreativer sein. Idealerweise können wir diese spartanische Lösung in naher Zukunft entfernen und uns einzig auf die localStorage()-Methoden des Browsers verlassen.

sessionStorage

Wir können localStorage für Daten verwenden, die auch erhalten bleiben sollen, nachdem unsere Benutzer den Webbrowser geschlossen haben. Aber manchmal brauchen wir eine Möglichkeit, Informationen zu speichern, während der Browser geöffnet ist, die aber verworfen werden können, sobald die Sitzung beendet ist. In diesem Fall kommt sessionStorage ins Spiel. Das funktioniert genauso wie localStorage, die Inhalte von sessionStorage werden aber gelöscht, sobald die Browsersitzung beendet ist. Anstatt das localStorage-Objekt abzurufen, rufen Sie einfach das sessionStorage-Objekt ab.

```

sessionStorage.setItem('name', 'Brian Hogan');
var name = sessionStorage.getItem('name');

```

Bei der Ausweidlösung müssen Sie lediglich dafür sorgen, dass die Cookies verfallen, sobald der Browser geschlossen wird.

21

Daten in einer clientseitigen relationalen Datenbank speichern

Die Methoden `localStorage` und `sessionStorage` bieten uns eine einfache Möglichkeit, simple Name-/Wertpaare auf dem Clientrechner zu speichern. Aber manchmal brauchen wir etwas mehr als das. Mit der HTML5-Spezifikation wurde ursprünglich auch die Möglichkeit eingeführt, Daten in relationalen Datenbanken zu speichern. Mittlerweile wurde dieser Teil in eine eigene Spezifikation mit dem Namen `Web SQL Storage`⁶ ausgegliedert. Mit ein bisschen Erfahrung mit dem Schreiben von SQL-Statements werden Sie sich gleich wie zu Hause fühlen. Damit Sie sich dabei auch wohlfühlen, werden wir mit `Web SQL Storage` Notizen in einer clientseitigen Datenbank erstellen, abrufen, aktualisieren und löschen.

CRUD in Ihrem Browser

Der Begriff `CRUD`, ein Akronym für „Create, Retrieve, Update and Delete“⁷ (Erstellen, Abrufen, Aktualisieren und Löschen), beschreibt ziemlich genau, was wir mit unserer clientseitigen Datenbank tun können. Die Spezifikation und die Implementierungen ermöglichen es uns, Datensätze einzufügen, auszuwählen, zu aktualisieren und zu löschen.

AwesomeCo möchte sein Vertriebsteam mit einer einfachen Anwendung ausrüsten, mit der die Mitarbeiter von unterwegs aus Notizen sammeln können. Mit der Anwendung sollen Benutzer neue Notizen anlegen, aktualisieren und löschen können. Damit Benutzer vorhandene Notizen verändern können, müssen sie sie außerdem aus der Datenbank abrufen können.

Hier kommen die SQL-Statements, die wir dafür schreiben müssen:

Typ	Statement
Notiz erstellen	<code>INSERT INTO notes (title,note) VALUES("Test", "This is a note");</code>
Alle Notizen abrufen	<code>SELECT id, title, note FROM notes;</code>
Bestimmte Notiz abrufen	<code>SELECT id, title, note FROM notes where id =1;</code>

6 <http://dev.w3.org/html5/webdatabase/>

7 Oder „Create, Read, Update and Destroy“, wenn Ihnen das besser gefällt.

Typ

Notiz aktualisieren

Notiz löschen

Statement

```
UPDATE notes set title = "bar",
note = "Changed" where id = 1;
```

```
DELETE FROM notes where id = 1;
```

**Joe fragt ...****Ist die Web SQL Database-Spezifikation nicht längst tot?**

Im November 2010 hat die Arbeitsgruppe für diese bestimmte Spezifikation erklärt, dass sie die Spezifikation nicht weiterverfolgt und sich stattdessen auf die IndexedDB-Spezifikation konzentriert. Wir sprechen in diesem Buch dennoch darüber, weil sie bereits in Webkit-basierten Browsern implementiert wurde, und dazu gehören alle Browser für iOS- und Android-Geräte, Safari und Google Chrome. Als dieses Buch geschrieben wurde, war IndexedDB noch überhaupt nicht implementiert. Im Gegensatz dazu können Sie Web SQL Databases bereits für Ihre Projekte verwenden. Vielleicht ist das ja genau das, was Sie brauchen.

Die Oberfläche für die Notizen

Die Oberfläche für die Notizanwendung besteht aus einer Seitenleiste auf der linken Seite mit einer Liste der vorhandenen Notizen und einem Formular auf der rechten Seite mit einem kleinen Feld und einem größeren Textbereich für die Notizen selbst. In Abbildung 9.2 sehen Sie, was wir basteln.

New note

- First note
- Second note

Save Delete

Title

First note

Note

This program lets you take notes and store them in the offline database.

Abbildung 9.2: Die Oberfläche unserer Notizanwendung

Als Erstes müssen wir die Oberfläche erstellen:

```
html5sql/index.html
```

```
<!doctype html>

<html>
  <head>
    <title>AwesomeNotes</title>
    <link rel="stylesheet" href="style.css">

    <script type="text/javascript"
      charset="utf-8"
      src="http://ajax.googleapis.com/ajax/libs/jquery/1.4.2/jquery.min.js">
    </script>

    <script type="text/javascript"
      charset="utf-8" src="javascripts/notes.js">
    </script>

  </head>

  <body>
    <section id="sidebar">
      <input type="button" id="new_button" value="New note">
      <ul id="notes">
      </ul>
    </section>

    <section id="main">
      <form>
        <ol>
          <li>
            <input type="submit" id="save_button" value="Save">
            <input type="submit" id="delete_button" value="Delete">
          </li>
          <li>
            <label for="title">Title</label>
            <input type="text" id="title">
          </li>
          <li>
            <label for="note">Note</label>
            <textarea id="note"></textarea>
          </li>
        </ol>
      </form>
    </section>

  </body>
</html>
```

Wir definieren die Seitenleiste und die Hauptbereiche mit `section`-Tags. Und wir geben jedem wichtigen Steuerelement der Benutzeroberfläche eine ID, wie zum Beispiel der Schaltfläche „Save“. Dadurch können wir die Elemente leichter ausfindig machen, um Event-Listener einzubinden.

Außerdem brauchen wir ein Stylesheet, damit die Oberfläche so wie in der Abbildung aussieht. `style.css` sieht folgendermaßen aus:

`html5sql/style.css`

```
#sidebar, #main{
  display: block;
  float: left;
}

#sidebar{
  width: 25%;
}

#main{
  width: 75%;
}

form ol{
  list-style: none;
  margin: 0;
  padding: 0;
}

form li{
  padding: 0;
  margin: 0;
}

form li label{
  display: block;
}

#title, #note{
  width: 100%;
  font-size: 20px;
  border: 1px solid #000;
}

#title{
  height: 20px;
}

#note{
  height: 40px;
}
```

Im Stylesheet werden die Aufzählungszeichen deaktiviert sowie die Größen für die Textbereiche und ein zweispaltiges Layout festgelegt. Nachdem wir mit der Oberfläche fertig sind, können wir jetzt das JavaScript schreiben.

Mit der Datenbank verbinden

Wir müssen eine Verbindung herstellen und eine Datenbank erstellen:

```
html5sql/javascripts/notes.js
```

```
// Referenz auf Datenbank
var db = null;

// Stellt eine Verbindung zur lokalen Datenbank her
connectToDB = function()
{
    db = window.openDatabase('awesome_notes', '1.0',
                             'AwesomeNotes Database', 1024*1024*3);
};
```

Im ersten Teil unseres Skripts deklarieren wir die Variable `db`. Dadurch ist sie für die übrigen Methoden verfügbar, die wir erstellen.⁸ Anschließend deklarieren wir die Methode, die über die Methode `window.openDatabase` eine Verbindung zur Datenbank herstellt. Dafür sind der Name der Datenbank, eine Versionsnummer, eine Beschreibung und ein Größenparameter erforderlich.

Eine Notiztabelle erstellen

Für unsere Notiztabelle brauchen wir drei Spalten:

Feld	Beschreibung
<code>id</code>	Eindeutige Kennzeichnung der Notiz. Primärschlüssel, Integer, Autoinkrement.
<code>title</code>	Der Titel der Notiz, um leichter darauf Bezug zu nehmen.
<code>note</code>	Die Notiz selbst.

Erstellen wir eine Methode, um diese Tabelle anzulegen:

```
html5sql/javascripts/notes.js
```

```
createNotestable = function()
{
    db.transaction(function(tx){
        tx.executeSql(
            "CREATE TABLE notes (id INTEGER \
             PRIMARY KEY, title TEXT, note TEXT)", [],
            function(){ alert('Notes database created successfully!'); },
            function(tx, error){ alert(error.message); });
    });
};
```

⁸ Dadurch erhält die Variable einen globalen Geltungsbereich. Das ist zwar nicht immer eine gute Idee, aber für dieses Beispiel möchten wir den JavaScript-Code so einfach wie möglich halten.

Wir führen die SQL-Anweisung innerhalb einer Transaktion aus. Die Transaktion verfügt über zwei Callback-Methoden: eine für die erfolgreiche Ausführung und eine für Fehler. Dieses Muster verwenden wir für alle Aktionen.

Beachten Sie, dass die Methode `executeSql()` als zweiten Parameter ein Array erwartet. Dieses Array dient dazu, SQL-Platzhalter mit Variablen zu verknüpfen. So können wir die Konkatenation von Zeichenfolgen vermeiden, ähnlich wie bei Prepared Statements in anderen Sprachen. In diesem Fall ist das Array leer, weil es in unserer Abfrage keine Platzhalter auszufüllen gibt.

Nachdem wir nun eine erste Tabelle haben, können wir die Anwendung auch tatsächlich etwas tun lassen.

Notizen laden

Wenn die Anwendung geladen wird, möchten wir uns mit der Datenbank verbinden und die Tabelle erstellen, falls sie noch nicht vorhanden ist. Anschließend möchten wir alle vorhandenen Notizen aus der Datenbank abrufen.

`html5sql/javascripts/notes.js`

```
// Lädt alle Datensätze aus der Tabelle "notes"
fetchNotes = function(){
  db.transaction(function(tx) {
    tx.executeSql('SELECT id, title, note FROM notes',[],
      function(SQLTransaction, data){
        for (var i = 0; i < data.rows.length; ++i) {
          var row = data.rows.item(i);
          var id = row['id'];
          var title = row['title'];

          addToNotesList(id, title);
        }
      });
  });
};
```

Diese Methode ruft die Ergebnisse aus der Datenbank ab. Falls dies erfolgreich geschieht, durchläuft sie die Ergebnisse und ruft die Methode `addNoteToList` auf, die wir folgendermaßen definieren:

`html5sql/javascripts/notes.js`

```
// Fügt das Listenelement zur Notizliste hinzu,
// erwartet eine ID und einen Titel
addToNotesList = function(id, title){
  var notes = $("#notes");
```

```

    var item = $("

```

Wir betten die ID des Datensatzes in ein benutzerdefiniertes Datenattribut ein. Die ID verwenden wir, um den zu ladenden Datensatz ausfindig zu machen, wenn ein Benutzer auf das Element klickt. Anschließend fügen wir das neu erstellte Listenelement in der Benutzeroberfläche zur ungeordneten Liste mit der ID `notes` hinzu. Jetzt müssen wir noch den Code schreiben, um dieses Element in das Formular zu laden, wenn eine Notiz aus der Liste ausgewählt wird.

Bestimmte Datensätze abrufen

Wir können zu jedem Listenelement ein `click`-Event hinzufügen. Aber der praktischere Ansatz besteht darin, jegliche Klicks auf die ungeordnete Liste zu überwachen und festzustellen, auf welches Element geklickt wurde. Auf diese Weise müssen wir nicht für jeden neuen Eintrag in die Liste (wenn wir zum Beispiel einen neuen Eintrag hinzufügen) ein `click`-Event hinzufügen.

Wir fügen den folgenden Code in unsere jQuery-Funktion ein:

html5sql/javascripts/notes.js

```

$("#notes").click(function(event){
    if ($(event.target).is('li')) {
        var element = $(event.target);
        loadNote(element.attr("data-id"));
    }
});

```

Dadurch wird die Methode `loadNote()` aufgerufen, die folgendermaßen aussieht:

html5sql/javascripts/notes.js

```

loadNote = function(id){
    db.transaction(function(tx) {
        tx.executeSql('SELECT id, title, note FROM notes where id = ?', [id],

        function(SQLTransaction, data){
            var row = data.rows.item(0);
            var title = $("#title");
            var note = $("#note");

            title.val(row["title"]);
            title.attr("data-id", row["id"]);
            note.val(row["note"]);
        }
    );
}

```

```

        $("#delete_button").show();
    });
}

```

Diese Methode sieht der Methode `fetchNotes()` sehr ähnlich. Sie schickt eine SQL-Anweisung ab und arbeitet bei Erfolg weiter. Diesmal enthält die Anweisung ein Fragezeichen als Platzhalter. Der tatsächliche Wert dafür steht im Array, dem zweiten Parameter.

Wenn wir einen Datensatz gefunden haben, zeigen wir ihn im Formular an. Außerdem aktiviert die Methode die Schaltfläche „Delete“ und bettet die ID des Datensatzes in ein benutzerdefiniertes Datenattribut ein, um Aktualisierungen einfacher handhaben zu können. Die Schaltfläche „Save“ prüft auf das Vorhandensein dieser ID. Ist sie vorhanden, aktualisieren wir den Datensatz. Falls sie nicht vorhanden ist, gehen wir davon aus, dass es sich um einen neuen Datensatz handelt. Die Logik dafür schreiben wir als Nächstes.

Datensätze einfügen, aktualisieren und löschen

Wenn ein Benutzer auf die Schaltfläche „Save“ klickt, möchten wir Code ausführen, der entweder einen neuen Datensatz einfügt oder einen vorhandenen aktualisiert. Wir fügen einen Handler für das `click`-Event zur Schaltfläche „Save“ hinzu, indem wir den folgenden Code in die `jQuery`-Funktion einfügen:

`html5sql/javascripts/notes.js`

```

$("#save_button").click(function(event){
    event.preventDefault();
    var title = $("#title");
    var note = $("#note");

    if(title.attr("data-id")){
        updateNote(title, note);
    }else{
        insertNote(title, note);
    }
});

```

Die Methode prüft das Attribut `data-id` des Formularfelds `title`. Wenn es keine ID gibt, geht das Formular davon aus, dass wir einen neuen Datensatz einfügen und ruft die Methode `insertNote` auf:

html5sql/javascrpts/notes.js

```

insertNote = function(title, note)
{
    db.transaction(function(tx){
        tx.executeSql("INSERT INTO notes (title, note) VALUES (?, ?)",
            [title.val(), note.val()],
            function(tx, result){
                var id = result.insertId ;
                alert('Record ' + id+ ' saved!');
                title.attr("data-id", result.insertId);
                addToNotesList(id, title.val());
                $("#delete_button").show();
            },

        ),

        function(){
            alert('The note could not be saved. ');
        }
    );
});
};

```

Die Methode insertNote() fügt den Datensatz in die Datenbank ein und verwendet die Eigenschaft insertId des Ergebnissatzes, um die eben eingefügte ID abzufragen. Anschließend wenden wir die ID als benutzerdefiniertes Datenattribut auf das Formularfeld title an und rufen die Methode addToNotesList() auf, um die Notiz zu unserer Liste am Rand der Seite hinzuzufügen.

Als Nächstes müssen wir uns um die Aktualisierungen kümmern. Die Methode updateNote() sieht genauso wie die anderen bisherigen Methoden aus:

html5sql/javascrpts/notes.js

```

updateNote = function(title, note)
{
    var id = title.attr("data-id");
    db.transaction(function(tx){
        tx.executeSql("UPDATE notes set title = ?, note = ? where id = ?",
            [title.val(), note.val(), id],
            function(tx, result){
                alert('Record ' + id + ' updated!');
                $("#notes>li[data-id=" + id + "]").html(title.val());
            },
            function(){
                alert('The note was not updated!');
            }
        );
    });
};

```

Ist die Update-Anweisung erfolgreich, aktualisieren wir den Titel der Notiz in unserer Liste, indem wir das Element suchen, dessen Feld `data-id` den Wert der ID enthält, die wir gerade aktualisiert haben.

Beim Löschen von Datensätzen gehen ähnlich vor. Wir brauchen einen Handler für das Delete-Event:

```
html5sql/javascripts/notes.js
```

```
$("#delete_button").click(function(event){
    event.preventDefault();
    var title = $("#title");
    deleteNote(title);
});
```

Außerdem brauchen wir die Methode `delete`, die den Datensatz nicht nur aus der Datenbank, sondern auch aus der Notizliste in der Seitenleiste entfernt:

```
html5sql/javascripts/notes.js
```

```
deleteNote = function(title)
{
    var id = title.attr("data-id");
    db.transaction(function(tx){
        tx.executeSql("DELETE from notes where id = ?",[id],
            function(tx, result){
                alert('Record ' + id + ' deleted!');
                $("#notes>li[data-id=" + id + "]").remove();
            },
            function(){
                alert('The note was not deleted!');
            }
        );
    });
};
```

Jetzt müssen wir nur noch das Formular leeren, damit wir einen neuen Datensatz erstellen können, ohne einen vorhandenen zu duplizieren.

Verpacken

Unsere Notizanwendung ist fast fertig. Wir müssen nur noch die Schaltfläche „New“ aktivieren, die bei einem Klick das Formular leert, damit Benutzer auch eine neue Notiz anlegen können, nachdem sie eine vorhandene bearbeitet haben. Wir verwenden dasselbe Muster wie zuvor – wir beginnen mit einem Event-Handler in der jQuery-Funktion für die Schaltfläche „New“:

css3transitions/style.css

```

    $("#new_button").click(function(event){
        event.preventDefault();
        newNote();
    });
    // Ende newbutton

    newNote();

});

```

Als Nächstes löschen wir das Attribut `data-id` des Felds `“title”` und entfernen die Werte aus dem Formular. Außerdem blenden wir die Schaltfläche `„Delete“` aus der Oberfläche aus.

html5sql/javascripts/notes.js

```

newNote = function(){
    $("#delete_button").hide();
    var title = $("#title");
    title.removeAttr("data-id");
    title.val("");
    var note = $("#note");
    note.val("");
}

```

Die Methode `newForm` soll von unserer jQuery-Funktion aus aufgerufen werden, wenn die Seite geladen wird, damit das Formular einsatzbereit ist. Auf diese Weise wird auch die Schaltfläche `„Delete“` versteckt.

Und das war's auch schon. Unsere Anwendung funktioniert jetzt auf iPhones, Android-Geräten und Desktop-Rechnern mit Chrome, Safari und Opera. Allerdings gibt es wenig Aussichten, dass sie jemals in Firefox funktionieren wird, und der Internet Explorer unterstützt sie auch nicht.

Ausweidlösung

Im Gegensatz zu unseren anderen Lösungen gibt es keine guten Bibliotheken, mit denen wir SQL Storage selbst implementieren können. Wir haben also keine Möglichkeit, Internet Explorer-Benutzer nativ zu unterstützen. Wenn Sie aber glauben, dass eine solche Anwendung nützlich ist, können Sie eventuell Ihre Benutzer überzeugen, für diese bestimmte Anwendung Google Chrome zu verwenden, der auf allen Plattformen funktioniert. Das wäre kein allzu ungewöhnliches Vorgehen, vor allem wenn Sie durch die Verwendung eines anderen Browsers eine interne Anwendung schreiben können, die auch auf mobilen Geräten funktioniert.

Eine weitere Alternative ist das Google Chrome Frame-Plugin⁹. Fügen Sie oben in Ihrer HTML-Seite direkt unterhalb des head-Tags Folgendes ein:

```
html5sql/index.html
```

```
<meta http-equiv="X-UA-Compatible" content="chrome=1">
```

Wenn das Google Chrome Frame-Plugin dieses Snippet liest, aktiviert es das Plugin für diese Seite.

Falls Sie überprüfen möchten, ob das Plugin verfügbar ist und gegebenenfalls Ihre Benutzer auffordern möchten, es zu installieren, können Sie das folgende Snippet direkt oberhalb des schließenden body-Tags einfügen:

```
html5sql/index.html
```

```
<script type="text/javascript"
  src= "http://ajax.googleapis.com/ajax/libs/chrome-frame/1/CFInstall.min.js">
</script>
```

```
<script>
```

```
  window.attachEvent("onload", function(){
    CFInstall.check({
      mode: "inline", // Standardwert
      node: "prompt"
    });
  });
</script>
```

Dadurch erhalten Benutzer die Option, das Plugin zu installieren, damit sie mit Ihrer Website arbeiten können.

Google Chrome Frame ist vielleicht keine geeignete Lösung für Webanwendungen, die für eine breite Öffentlichkeit gedacht sind, aber für interne Anwendungen wie diese funktioniert es gut. Eventuell gibt es in Unternehmen IT-Richtlinien, die so etwas verbieten. Aber dann liegt es an Ihnen, ob Sie in einer derartigen Situation eine solche Lösung genehmigt bekommen. Die Installation eines Plugins ist sicherlich günstiger als die Programmierung eines eigenen SQL-Datenbanksystems.

9 <http://code.google.com/chrome/chromeFrame>

22

Offline arbeiten

Mit der Offline-Unterstützung von HTML5,¹⁰ können wir mit HTML und verwandten Technologien Anwendungen erstellen, die auch ohne Internetverbindung funktionieren. Das ist besonders nützlich bei Anwendungen für mobile Geräte, bei denen unter Umständen die Verbindung abbricht.

Diese Technik funktioniert in Firefox, Chrome und Safari ebenso wie mit iOS- und Android 2.0-Geräten. Aber es gibt keine Ausweichlösung für eine vernünftige Offline-Unterstützung für den Internet Explorer.

AwesomeCo hat gerade einige iPads für sein Vertriebsteam gekauft. Die Notizanwendung, die wir im Abschnitt *Daten in einer clientseitigen relationalen Datenbank speichern* auf Seite 186 entwickelt haben, soll nun auch offline funktionieren. Dank der Manifestdatei von HTML5 ist das einfach.

Mit dem Manifest einen Cache definieren

Die Manifestdatei enthält eine Liste aller clientseitigen Dateien einer Webanwendung, die im Cache des Clientbrowsers vorhanden sein müssen, damit die Anwendung auch offline funktioniert. Jede Datei, auf die die Anwendung zugreift, muss angegeben werden. Lediglich die Datei, die das Manifest einbindet, muss nicht angegeben werden, da sie implizit im Cache zwischengespeichert wird.

Erstellen Sie die Datei `notes.manifest`. Sie sollte folgenden Inhalt haben:

```
html5offline/notes.manifest
```

```
CACHE MANIFEST
# v = 1.0.0
/style.css
/javascripts/notes.js
/javascripts/jquery.min.js
```

Mit dem Versionskommentar in der Datei haben wir ein Instrument, über das wir dem Browser mitteilen können, wann er neue Versionen unserer Dateien holen soll. Immer wenn wir unseren Code ändern, müssen wir auch das Manifest ändern.

¹⁰ <http://www.w3.org/TR/html5/offline.html>

Bisher haben wir jQuery von Google für uns hosten lassen. Aber wenn unsere Anwendung auch offline funktionieren soll, geht das nicht. Entsprechend müssen wir jQuery herunterladen und unser script-Tag so ändern, dass jQuery aus unserem lokalen Verzeichnis javascripts geladen wird.

```
html5offline/index.html
```

```
<script type="text/javascript"
  charset="utf-8"
  src="javascripts/jquery.min.js">
</script>
```

Als Nächstes müssen wir die Manifestdatei in unser HTML-Dokument einbinden. Dazu ändern wir das html-Element folgendermaßen:

```
html5offline/index.html
```

```
<html manifest="notes.manifest">
```

Mehr brauchen wir nicht zu tun. Die Sache hat nur einen Haken – die Manifestdatei muss von einem Webserver bereitgestellt werden, weil das Manifest mit dem MIME-Type `text/cache-manifest` bereitgestellt werden muss. Wenn Sie Apache verwenden, können Sie den MIME-Type in einer `.htaccess`-Datei folgendermaßen festlegen:

```
html5offline/.htaccess
```

```
AddType text/cache-manifest .manifest
```

Wenn wir unsere Notizanwendung zum ersten Mal anfordern, werden die im Manifest aufgeführten Dateien heruntergeladen und im Cache zwischengespeichert. Danach können wir die Netzwerkverbindung trennen und die Anwendung so oft wir möchten offline verwenden.

Recherchieren Sie unbedingt in der Spezifikation. Für die Manifestdatei existieren auch komplexere Optionen, die Sie verwenden können. So können Sie beispielsweise festlegen, dass bestimmte Dinge nicht zwischengespeichert werden sollen und dass darauf niemals offline zugegriffen werden darf. Das kann nützlich sein, um bestimmte dynamische Dateien auszuschließen.

Manifest und Caching

Während der Entwicklung Ihrer Anwendungen sollten Sie auf Ihrem Webserver jegliches Caching deaktivieren. Standardmäßig speichern viele Webserver Dateien im Cache, indem sie im Header den Browser anweisen, für eine bestimmte Zeit keine neue Version einer Datei abzu-

holen. Das kann Sie ins Stolpern bringen, wenn Sie Dateien zu Ihrer Manifestdatei hinzufügen.

Wenn Sie Apache verwenden, können Sie das Caching deaktivieren, indem Sie Folgendes zu Ihrer `.htaccess`-Datei hinzufügen:

```
html5offline/.htaccess
```

```
ExpiresActive On  
ExpiresDefault "access"
```

Dadurch wird das Caching für das gesamte Verzeichnis deaktiviert. Das sollten Sie in einer Produktionsumgebung nicht tun, aber so ist gewährleistet, dass Ihr Browser immer eine neue Version der Manifestdatei anfordert.

Wenn Sie eine in Ihrer Manifestdatei aufgeführte Datei ändern, müssen Sie auch den Kommentar mit der Versionsnummer in der Manifestdatei ändern.

Die Zukunft

Funktionen wie `localStorage` und Web SQL Databases geben Entwicklern die Möglichkeit, Anwendungen im Browser zu entwickeln, die nicht mit einem Webserver verbunden sein müssen. Anwendungen wie die, an denen wir gearbeitet haben, laufen ebenso auf einem iPad oder Android-Gerät. Und in Kombination mit der HTML5-Manifestdatei können wir mit vertrauten Werkzeugen statt mit proprietären Plattformen umfangreiche Offline-Anwendungen erstellen. Da immer mehr Browser die Unterstützung hierfür bieten, können Sie von Entwicklern wirksam eingesetzt werden: für Anwendungen, die auf mehreren Plattformen und Geräten funktionieren, die Daten lokal speichern und synchronisieren, wenn eine Verbindung besteht.

Die Zukunft von Web SQL Storage ist ungewiss. Mozilla hat keine Pläne für eine Implementierung in Firefox, und das W3C hat sich entschieden, stattdessen die IndexedDB-Spezifikation zu implementieren. Auf diese Spezifikation gehen wir im Abschnitt *Indexed Database API* auf Seite 226 noch näher ein. Allerdings wird Web SQL Storage seit geraumer Zeit auf iOS- und Android-Geräten verwendet, und so wird es wahrscheinlich auch bleiben. Diese Spezifikation könnte für Sie daher extrem nützlich sein, wenn Sie Anwendungen für diesen Bereich entwickeln.

Kapitel 10

Mit anderen APIs spielen

Viele interessante APIs waren anfangs Teil der HTML5-Spezifikation und wurden irgendwann in eigene Projekte ausgegliedert. Andere werden so oft mit HTML5 in Verbindung gebracht, sodass es für Entwickler (und sogar für Autoren) manchmal schwierig ist, den Unterschied zu erkennen. In diesem Kapitel befassen wir uns mit genau diesen APIs. Wir arbeiten ein bisschen mit der HTML5-History-API und lassen Seiten auf verschiedenen Servern sich über Cross-Document Messaging unterhalten. Anschließend werfen wir einen Blick auf Web Sockets und Geolocation, zwei sehr leistungsfähige APIs, mit denen Sie sogar noch interaktivere Anwendungen erstellen können.

Diese Anwendungen erstellen wir mit den folgenden APIs:¹

History

Verwaltet den Browserverlauf. *[C5, S4, IE8, F3, O10.1, IOS3.2, A2]*

Cross-Document Messaging

Sendet Nachrichten zwischen Fenstern mit Inhalten von verschiedenen Domains. *[C5, S5, F4, IOS4.1, A2]*

¹ In den folgenden Beschreibungen wird die Unterstützung durch die verschiedenen Browser in eckigen Klammern mit einem Kurzcode und der mindestens erforderlichen Versionsnummer angegeben. Die verwendeten Codes lauten: *C*: Google Chrome, *F*: Firefox, *IE*: Internet Explorer, *O*: Opera, *S*: Safari, *IOS*: iOS-Geräte mit Mobile Safari und *A*: Android-Browser.

Web Sockets

Stellt eine zustandsbezogene Verbindung zwischen einem Browser und einem Server her. *[C5, S5, F4, IOS4.2]*

Geolocation

Ruft den Breiten- und Längengrad vom Browser des Clients ab. *[C5, S5, F3.5, O10.6, IOS3.2, A2]*

23

Den Verlauf erhalten

Die HTML5-Spezifikation führt eine API zur Verwaltung des Browserverlaufs ein.¹ Im Abschnitt *Barrierefreie aktualisierbare Bereiche erstellen* auf Seite 106 haben wir den Prototyp für die neue Homepage von AwesomeCo erstellt, deren Hauptinhalt ausgetauscht wird, wenn wir auf einen der Navigationspunkte klicken. Ein Nachteil dieses Ansatzes besteht darin, dass die Rückwärtstaste des Browsers damit nicht funktioniert. Mit einigen Hacks lässt sich das zwar ändern, aber mit der History API können wir eine wirklich gute Lösung dafür finden.

Die Unterstützung für die API überprüfen wir folgendermaßen:

```
html5history/javascripts/application.js
```

```
function supportsHistory(){
  return !!window.history && window.history.pushState;
}
```

Diese Methode verwenden wir jedes Mal, wenn wir mit den History-Objekten zu tun haben.

Den aktuellen Zustand speichern

Wenn ein Besucher eine neue Webseite öffnet, fügt der Browser diese Seite zum Verlauf hinzu. Wenn ein Besucher dagegen einen Abschnitt auswählt, müssen wir diesen Abschnitt selbst zum Verlauf hinzufügen.

```
html5history/javascripts/application.js
```

```
1  $("nav ul").click(function(event){
2    target = $(event.target);
3    if(target.is("a")){
4      event.preventDefault();
5      if ( $(target.attr("href")).hasClass("hidden") ){
6
7        if(supportsHistory()){
8          var tab = $(target).attr("href");
9          var stateObject = {tab: tab};
10         window.history.pushState(stateObject, tab);
11       };
12
13       $(".visible").removeClass("visible").addClass("hidden").hide();
```

¹ <http://www.w3.org/TR/html5/history.html>

```

14         $(target.attr("href")).removeClass("hidden").addClass("visible").show();
15     };
16 };
17 });
18
19 });

```

Wir schnappen uns die ID des sichtbaren Elements und fügen einen Verlaufszustand zum Browser hinzu. Der erste Parameter der Methode `pushstate()` ist ein Objekt, mit dem wir später interagieren können. Wir verwenden es, um die ID des Abschnitts zu speichern, den wir anzeigen möchten, wenn unser Benutzer an diese Stelle zurücknavigiert. Wenn der Benutzer beispielsweise auf den Link „Services“ klickt, speichern wir `#services` im Zustandsobjekt.

Der zweite Parameter ist ein Titel, anhand dessen wir den Zustand in unserem Verlauf erkennen können. Das hat nichts mit dem `title`-Element der Seite zu tun. Das ist einfach nur eine Möglichkeit, den Eintrag im Browserverlauf zu kennzeichnen. Wir verwenden auch hierfür wieder die ID des Abschnitts.

Den vorherigen Status abrufen

Hierdurch wird zwar ein Verlaufszustand hinzugefügt, wir müssen aber auch den Code schreiben, um auf einen Zustandswechsel des Verlaufs zu reagieren. Wenn der Benutzer auf die Zurück-Taste klickt, wird das Event `window.onpopstate()` ausgelöst. An dieser Stelle haken wir uns ein, um den im Zustandsobjekt gespeicherten Navigationspunkt anzuzeigen.

html5history/javascrpts/application.js

```

if(supportsHistory()){
    window.onpopstate = function(event){
        if(event.state){
            var tab = (event.state["tab"]);
            $(".visible")
                .removeClass("visible")
                .addClass("hidden")
                .hide();
            $(tab)
                .removeClass("hidden")
                .addClass("visible")
                .show();
        }
    };
};

```

Wir lesen den Namen des Abschnitts aus und suchen dann mit jQuery das auszublendende Element anhand seiner ID. Wir wiederholen hier wieder den ursprünglichen Code zum Ausblenden und Anzeigen der einzelnen Abschnitte. Dies ist eine gute Gelegenheit für Refactoring, um die Wiederholung zu vermeiden.

Standardwert

Wenn wir zum ersten Mal unsere Seite aufrufen, ist unserer Verlaufszustand null, wir müssen ihn also selbst festlegen. Das können wir direkt oberhalb der Definition der Methode `window.onpopstate()` tun.

html5history/javascripts/application.js

```
if(supportsHistory()){
  window.history.pushState( {tab: "#welcome"}, '#welcome');
  window.onpopstate = function(event) {
    if(event.state){
      var tab = (event.state["tab"]);
      $(".visible")
        .removeClass("visible")
        .addClass("hidden")
        .hide();
      $(tab)
        .removeClass("hidden")
        .addClass("visible")
        .show();
    }
  };
};
```

Wenn wir jetzt die Seite öffnen, können wir über den Browserverlauf durch unsere Tabs blättern.¹

Ausweichlösung

Das funktioniert in Firefox 4, Safari 4 und auch in Chrome 5, aber nicht im Internet Explorer. Lösungen wie das jQuery Address-Plugin² bieten dieselbe Funktionalität. Wir werden es aber nicht implementieren, weil es weniger eine Ausweichlösung als vielmehr ein Ersatz mit einer Menge zusätzlicher Funktionen ist. Behalten Sie aber die Browserunterstützung für die Manipulation des Verlaufs im Auge, weil Sie damit ganz einfach benutzerfreundlichere Anwendungen schreiben können, wenn diese API in jedem Browser unterstützt wird.

¹ Beim Testen sollten Sie Ihren Browser immer wieder schließen und den Verlauf löschen. Das kann manchmal ziemlich nerven.

² <http://www.asual.com/jquery/address/>

24

Über Domains hinweg kommunizieren

Clientseitigen Webanwendungen war es immer untersagt, direkt mit Skripten von anderen Domains zu kommunizieren. Diese Einschränkung dient dem Schutz der Benutzer.¹ Es gibt zahlreiche schlaue Möglichkeiten, diese Einschränkung zu umgehen, darunter die Verwendung serverseitiger Proxies und intelligenter URL-Hacks. Aber jetzt gibt es eine bessere Möglichkeit.

Mit der HTML5-Spezifikation wurde *Cross-Document Messaging* eingeführt, eine API, über die auf verschiedenen Domains gehostete Skripten Nachrichten hin- und herschicken können. So können wir beispielsweise mit einem Formular auf `http://support.awesomecompany.com` Inhalte in ein anderes Fenster oder `iframe` posten, dessen Inhalt auf `http://www.awesomecompany.com` gehostet wird. Wie sich herausstellt, müssen wir in unserem aktuellen Projekt genau das tun.

Auf der neuen Support-Website von AwesomeCo gibt es ein Kontaktformular. Und der Supportmanager möchte neben dem Kontaktformular alle Supportkontakte und die entsprechenden E-Mail-Adressen auflisten. Die Supportkontakte werden irgendwann von einem Content-Management-System oder einem anderen Server geliefert. Wir können also die Kontaktliste neben dem Formular in ein `iframe` einbetten. Der Haken ist allerdings: Der Supportmanager möchte unbedingt, dass wir die E-Mail-Adresse automatisch in unser Formular einfügen, wenn Benutzer in der Kontaktliste auf einen Namen klicken.

Das ist ziemlich einfach. Aber Sie brauchen einen Webserver, um alles richtig zu testen. Die Beispiele, mit denen wir hier arbeiten, funktionieren ohne Server nicht in jedem Browser. Mehr dazu können Sie im Kasten auf der folgenden Seite lesen.

Die Kontaktliste

Als Erstes erstellen wir die Kontaktliste. Das einfache Markup hierfür sieht folgendermaßen aus:

¹ Das ist die sogenannte „Same Origin Policy“. Eine genaue Erklärung finden Sie unter https://developer.mozilla.org/en/Same_origin_policy_for_JavaScript.

```
html5xdomain/contactlist/public/index.html
```

```
<ul id="contacts">
  <li>
    <h2>Sales</h2>
    <p class="name">James Norris</p>
    <p class="email">j.norris@awesomeco.com</p>
  </li>
  <li>
    <h2>Operations</h2>
    <p class="name">Tony Raymond</p>
    <p class="email">t.raymond@awesomeco.com</p>
  </li>
  <li>
    <h2>Accounts Payable</h2>
    <p class="name">Clark Greenwood</p>
    <p class="email">c.greenwood@awesomeco.com</p>
  </li>
  <li>
    <h2>Accounts Receivable</h2>
    <p class="name">Herbert Whitmore</p>
    <p class="email">h.whitmore@awesomeco.com</p>
  </li>
</ul>
```

Einfacher Webserver

Wenn Sie nicht die Mühe auf sich nehmen möchten, Apache-Instanzen zu konfigurieren oder Ihre eigenen Server einzurichten, können Sie die einfachen Ruby-basierten Server aus dem Beispielcode zu diesem Buch verwenden. Anweisungen, wie Sie Ruby auf Ihrem System zum Laufen bekommen, finden Sie in der Datei `RUBY_README.txt` im Quellcode zu diesem Buch.

Um die Server zu starten, wechseln Sie in das Verzeichnis `html5xdomain/contactlist` und führen die Datei `server.rb` folgendermaßen aus:

```
ruby server.rb
```

Der Server wird auf Port 4567 gestartet. Dasselbe können Sie anschließend mit der `server.rb` in `html5xdomain/supportpage` tun, die auf Port 3000 gestartet wird. Sie können die Ports in der jeweiligen Datei `server.rb` ändern.

Auf dieser Seite laden wir auch die jQuery-Bibliothek und unsere eigene Datei `application.js` sowie ein einfaches Stylesheet. Folgendes schreiben wir in unseren head-Abschnitt:

```
html5xdomain/contactlist/public/index.html
```

```
<script type="text/javascript"
  charset="utf-8"
  src="http://ajax.googleapis.com/ajax/libs/jquery/1.4.2/jquery.min.js">
</script>
<script type="text/javascript"
  src="javascripts/application.js">
</script>
<link rel="stylesheet" href="style.css" type="text/css" media="screen">
```

Und so sieht das Stylesheet für die Kontaktliste aus:

```
html5xdomain/contactlist/public/style.css
```

```
ul{
  list-style: none;
}

ul h2, ul p{margin: 0;}
ul li{margin-bottom: 20px;}
```

Nur ein paar Kniffe, damit die Liste halbwegs aufgeräumt aussieht.

Nachrichten senden

Wenn ein Benutzer auf einen Eintrag in unserer Kontaktliste klickt, lesen wir die E-Mail-Adresse aus dem Listenelement und senden eine Nachricht zurück an das übergeordnete Fenster. Die Methode `postMessage()` erwartet zwei Parameter: die Nachricht selbst und den Ursprung des Zielfensters. So sieht der komplette Event-Handler aus:

```
html5xdomain/contactlist/public/javascripts/application.js
```

```
$(function(){
  $("#contacts li").click(function(event){
    var email = ($(this).find(".email").html());
    var origin = "http://192.168.1.244:3000/index.html";
    window.parent.postMessage(email, origin);
  });
});
```

Falls Sie den Code mitschreiben, müssen Sie den Ursprung anpassen, da er der URL des übergeordneten Fensters entsprechen muss.¹

Nun müssen wir die Seite implementieren, die den Frame aufnimmt und die Nachrichten empfängt.

¹ Das stimmt nicht ganz. Sie können auch einfach die Domain oder sogar einen Platzhalter verwenden. Aber unsere Ausweichlösung benötigt die gesamte URL, und das empfiehlt sich auch aus Sicherheitsgründen.

Die Support-Website

Die Struktur der Support-Website wird ziemlich ähnlich aussehen. Aber um die beiden Websites voneinander zu trennen, sollten wir in verschiedenen Verzeichnissen arbeiten, insbesondere da die Website auf einem anderen Webserver abgelegt wird. Wir müssen Links auf ein Stylesheet, jQuery und die Datei `application.js` einfügen.

Unsere Supportseite braucht ein Kontaktformular und ein `iframe`, das auf unsere Kontaktliste verweist. Wir lösen das ungefähr so:

```
html5xdomain/supportpage/public/index.html
```

```
<div id="form">
  <form id="supportform">
    <fieldset>
      <ol>
        <li>
          <label for="to">To</label>
          <input type="email" name="to" id="to">
        </li>
        <li>
          <label for="from">From</label>
          <input type="text" name="from" id="from">
        </li>
        <li>
          <label for="message">Message</label>
          <textarea name="message" id="message"></textarea>
        </li>
      </ol>
      <input type="submit" value="Send!">
    </fieldset>
  </form>
</div>

<div id="contacts">
  <iframe src="http://192.168.1.244:4567/index.html"></iframe>
</div>
```

Gestylt wird das Ganze mit dem folgenden CSS, das wir in `style.css` schreiben:

```
html5xdomain/supportpage/public/style.css
```

```
#form{
  width: 400px;
  float: left;
}
#contacts{
  width: 200px;
  float: left;
}
```

```

#contacts iframe{
  border: none;
  height: 400px;
}

fieldset{
  width: 400px;
  border: none;
}

fieldset legend{
  background-color: #ddd;
  padding: 0 64px 0 2px;
}

fieldset>ol{
  list-style: none;
  padding: 0;
  margin: 2px;
}

fieldset>ol>li{
  margin: 0 0 9px 0;
  padding: 0;
}

/* Jedes Eingabefeld erhält eine eigene Zeile */
fieldset input, fieldset textarea{
  display: block;
  width: 380px;
}

fieldset input[type=submit]{
  width: 390px;
}

fieldset textarea{
  height: 100px;
}

```

To

From

Message

Send!

Sales

James Norris
j.norris@awesomeco.com

Operations

Tony Raymond
t.raymond@awesomeco.com

Accounts Payable

Clark Greenwood
c.greenwood@awesomeco.com

Accounts Receivable

Herbert Whitmore
h.whitmore@awesomeco.com

Abbildung 10.1: Unsere fertige Support-Website

Dadurch werden Formular und `iframe` nebeneinander angeordnet und das Formular so geändert, dass es wie in Abbildung 10.1 auf der vorherigen Seite aussieht.

Nachrichten empfangen

Das `onmessage`-Event wird ausgelöst, wenn das aktuelle Fenster eine Nachricht erhält. Die Nachricht wird als Eigenschaft von `event` zurückgeliefert. Wir registrieren dieses Event mit der `bind()`-Methode von jQuery, damit es in allen Browsern gleich funktioniert.

```
html5xdomain/supportpage/public/javascripts/application.js
```

```
$(function(){
    $(window).bind("message",function(event){
        $("#to").val(event.originalEvent.data);
    });
});
```

Die `bind()`-Methode von jQuery verpackt das eigentliche Event und reicht nicht alle Eigenschaften durch. Wir bekommen aber alle erforderlichen Daten, indem wir auf die Eigenschaft `originalEvent` des Events zugreifen.

Wenn Sie das in Firefox, Chrome, Safari oder im Internet Explorer 8 öffnen, werden Sie feststellen, dass es extrem gut funktioniert. Jetzt müssen wir es noch im IE 6 und 7 zum Laufen bekommen.

Ausweichlösung

Um den IE 6 und 7 zu unterstützen, verwenden wir das jQuery-Plugin `Postback`, das `Cross-Domain Messaging` emuliert. Wir verwenden die `getScript()`-Methode von jQuery, um diese Bibliothek nur dann einzubinden, wenn wir sie brauchen. Dazu ermitteln wir einfach, ob die Methode `postMessage()` vorhanden ist.

Zuerst passen wir unsere Kontaktliste an:

```
html5xdomain/contactlist/public/javascripts/application.js
```

```
► if(window.postMessage){
    window.parent.postMessage(email, origin);
► }else{
►     $.getScript("javascripts/jquery.postmessage.js", function(){
►         $.postMessage(email, origin, window.parent);
►     });
► }
```

Das jQuery-Plugin Postmessage stellt die Methode `postMessage()` bereit, die beinahe genauso wie ihr standardmäßiges Pendant funktioniert.

Jetzt gehen wir die Support-Website an. Wir folgen demselben Ansatz, binden die Bibliothek ein und rufen die neu hinzugefügte Methode `receiveMessage()` auf:

```
html5xdomain/supportpage/public/javascripts/application.js
```

```
► if(window.postMessage){
    $(window).bind("message",function(event){
        $("#to").val(event.originalEvent.data);
    });
► }else{
►     $.getScript("javascripts/jquery.postmessage.js", function(){
►         $.receiveMessage(
►             function(event){
►                 $("#to").val(event.data);
►             });
►         });
► }
```

Das war's! Wir können jetzt in einer ganzen Reihe von Browsern von Fenster zu Fenster kommunizieren. Und das ist erst der Anfang. Sie können diese Technik auf eine bidirektionale Kommunikation erweitern. Jedes Fenster kann Sender oder Empfänger sein. Werfen Sie also einen Blick auf die Spezifikation, und schauen Sie, was Sie damit machen können!

Webentwickler versuchen schon seit Jahren, Interaktion in Echtzeit zu realisieren. Aber bei den meisten Implementierungen wurde JavaScript verwendet, um den entfernten Server regelmäßig auf Änderungen zu überprüfen. HTTP ist ein zustandsloses Protokoll. Der Webbrowser stellt eine Verbindung zu einem Server her, erhält eine Antwort und trennt die Verbindung. Jegliche Echtzeitfunktionalität über ein zustandsloses Protokoll stellt eine echte Herausforderung dar. Mit der HTML5-Spezifikation wurden Web Sockets eingeführt, über die der Browser eine zustandsbezogene Verbindung zu einem entfernten Server herstellen kann.¹ Mit Web Sockets können wir alle möglichen Arten von tollen Anwendungen erstellen. Eine der besten Möglichkeiten, ein Gefühl dafür zu entwickeln, wie Web Sockets funktionieren, ist die Entwicklung eines Chat-Clients. Rein zufällig möchte AwesomeCo so etwas für seine Support-Website haben.

AwesomeCo möchte eine einfache, webbasierte Chat-Oberfläche für seine Support-Website entwickeln, über die die Supportmitarbeiter intern kommunizieren können. Die Mitarbeiter sind nämlich auf verschiedene Städte verteilt. Wir werden Web Sockets verwenden, um die Webschnittstelle für den Chat-Server zu implementieren. Benutzer können eine Verbindung herstellen und eine Nachricht an den Server schicken. Alle verbundenen Benutzer bekommen diese Nachricht angezeigt. Unsere Besucher können sich selbst einen Spitznamen geben, indem sie eine Nachricht wie zum Beispiel „/nick brian“ schicken, in Anlehnung an das IRC-Chat-Protokoll. Den eigentlichen Server hierfür werden wir nicht schreiben, weil das glücklicherweise schon ein anderer Entwickler getan hat.²

Die Chat-Oberfläche

Wir möchten eine sehr einfache Chat-Oberfläche erstellen, die wie in Abbildung 10.2 auf der nächsten Seite aussieht: Sie enthält ein Formular, mit dem der Benutzer seinen Spitznamen ändern kann, einen großen Bereich, in dem die Nachrichten angezeigt werden, und ein Formular zum Senden einer Chat-Nachricht.

1 Web Sockets wurde in eine eigene Spezifikation ausgegliedert. Sie finden sie unter <http://www.w3.org/TR/websockets/>.

2 Im Abschnitt *Server* auf Seite 219 erfahren Sie mehr zu Servern.

Wir fügen das Markup für die Chat-Oberfläche, die aus zwei Formularen und einem div für die Chat-Nachrichten besteht, in eine neue HTML5-Seite ein.

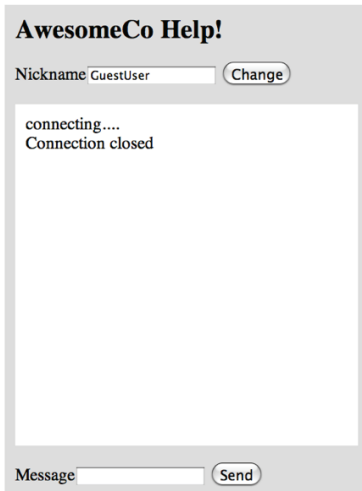


Abbildung 10.2: Unsere Chat-Oberfläche

html5_websockets/public/index.html

```
<div id="chat_wrapper">
  <h2>AwesomeCo Help!</h2>
  <form id="nick_form" action="#" method="post" accept-charset="utf-8">
    <p>
      <label>Nickname
        <input id="nickname" type="text" value="GuestUser"/>
      </label>
      <input type="submit" value="Change">
    </p>
  </form>

  <div id="chat">connecting....</div>

  <form id="chat_form" action="#" method="post" accept-charset="utf-8">
    <p>
      <label>Message
        <input id="message" type="text" />
      </label>
      <input type="submit" value="Send">
    </p>
  </form>
</div>
```

Außerdem müssen wir ein Stylesheet und eine JavaScript-Datei verknüpfen, die unseren Code für die Kommunikation mit dem Web Sockets-Server enthält.

```
html5_websockets/public/index.html
```

```
<script src='chat.js' type='text/javascript'></script>
<link rel="stylesheet" href="style.css" media="screen">
```

Unser Stylesheet enthält die folgenden Stildefinitionen:

```
html5_websockets/public/style.css
```

```
1 #chat_wrapper{
2   width: 320px;
3   height: 440px;
4   background-color: #ddd;
5   padding: 10px;
6 }
7 #chat_wrapper h2{
8   margin: 0;
9 }
10
11 #chat{
12   width: 300px;
13   height: 300px;
14   overflow: auto;
15   background-color: #fff;
16   padding: 10px;
17 }
```

In Zeile 14 legen wir die Eigenschaft `overflow` für den Chat-Nachrichtenbereich fest, damit er eine feste Höhe hat und nicht sichtbarer Text über Scrollleisten angezeigt werden kann.

Nachdem unsere Oberfläche jetzt fertig ist, können wir uns an die Arbeit mit dem JavaScript machen, das mit unserem Chat-Server spricht.

Mit dem Server sprechen

Egal mit welchem Web Sockets-Server wir arbeiten, wir wenden immer dasselbe Muster an. Wir stellen eine Serververbindung her, warten auf Events vom Server und reagieren entsprechend.

Event	Beschreibung
<code>onopen()</code>	Wird ausgelöst, wenn die Verbindung zum Server hergestellt wurde.
<code>onmessage()</code>	Wird ausgelöst, wenn über die Serververbindung eine Nachricht gesendet wird.
<code>onclose()</code>	Wird ausgelöst, wenn die Verbindung zum Server abbricht oder geschlossen wird.

In der Datei `chat.js` müssen wir zunächst eine Verbindung zum Web Sockets-Server herstellen:

```
html5_websockets/public/chat.js
```

```
var websocket = new WebSocket('ws://localhost:9394/');
```

Wir sollten es den Benutzer wissen lassen, wenn wir uns mit dem Server verbinden. So definieren wir die Methode `onopen()`:

```
html5_websockets/public/chat.js
```

```
websocket.onopen = function(event){
    $('#chat').append('<br>Connected to the server');
};
```

Wenn der Browser die Verbindung zum Server öffnet, geben wir im Chat-Fenster eine Nachricht aus. Als Nächstes müssen wir die an den Chat-Server gesendeten Nachrichten anzeigen. Dafür definieren wir die Methode `onmessage()`:

```
html5_websockets/public/chat.js
```

```
websocket.onmessage = function(event){
    $('#chat').append("<br>" + event.data);
    $('#chat').animate({scrollTop: $('#chat').height()});
};
```

Die Nachricht kommt über die Eigenschaft `data` des `event`-Objekts zu uns zurück. Wir fügen sie einfach in unser Chat-Fenster ein. Wir stellen einen Zeilenumbruch voran, sodass jede Antwort in einer eigenen Zeile steht. Aber das können Sie natürlich machen, wie Sie möchten.

Als Nächstes kümmern wir uns um den Fall, dass die Verbindung beendet wird. Die Methode `onclose()` wird ausgelöst, wenn die Verbindung geschlossen wird.

```
html5_websockets/public/chat.js
```

```
websocket.onclose = function(event){
    $("#chat").append('<br>Connection closed');
};
```

Jetzt müssen wir nur noch den Textbereich für das Chat-Formular scharf schalten, damit wir auch Nachrichten an den Server schicken können.

```
html5_websockets/public/chat.js
```

```
$(function(){
    $("#form#chat_form").submit(function(e){
        e.preventDefault();
        var textfield = $("#message");
```

```

        websocket.send(textfield.val());
        textfield.val("");
    });
}

```

Wir haken uns in das submit-Event des Formulars ein, schnappen den Wert des Formularfelds und senden es mit der send()-Methode an den Chat-Server.

Die Funktion zum Ändern des Spitznamens implementieren wir auf dieselbe Weise. Wir stellen der gesendeten Nachricht lediglich das Präfix /nick voran. Der Chat-Server erkennt das und ändert den Benutzernamen.

html5_websockets/public/chat.js

```

$("form#nick_form").submit(function(e){
    e.preventDefault();
    var textfield = $("#nickname");
    websocket.send("/nick " + textfield.val());
});

```

Das war's auch schon. Safari 5- und Chrome 5-Benutzer können sofort mit diesem Client den Echtzeit-Chat nutzen. Natürlich müssen wir auch Browser ohne native Web Sockets-Unterstützung berücksichtigen. Das machen wir mit Flash.

Ausweidlösung

Zwar unterstützen nicht alle Browser Socket-Verbindungen, aber dafür schon seit einiger Zeit Adobe Flash. Wir können Flash als Kommunikationsschicht für unseren Sockel verwenden. Dank der Bibliothek websocket-js¹ ist die Implementierung einer Flash-Ausweidlösung ein Klacks.

Wir können eine Version des Plugins² herunterladen und in unser Projekt einbinden. Dafür müssen wir drei JavaScript-Dateien in unsere Seite einbinden:

html5_websockets/public/index.html

```

<script type="text/javascript" src="websocket_js/swfobject.js"></script>
<script type="text/javascript" src="websocket_js/FABridge.js"></script>
<script type="text/javascript" src="websocket_js/web_socket.js"></script>

<script src='chat.js' type='text/javascript'></script>
<link rel="stylesheet" href="style.css" media="screen">

```

1 <http://github.com/gimite/web-socket-js/>

2 <http://github.com/gimite/web-socket-js/archives/master>

```

</head>
<body>
<div id="chat_wrapper">
  <h2>AwesomeCo Help!</h2>
  <form id="nick_form" action="#" method="post" accept-charset="utf-8">
    <p>
      <label>Nickname
      <input id="nickname" type="text" value="GuestUser"/>
    </label>
    <input type="submit" value="Change">
  </p>
</form>

<div id="chat">connecting...</div>

<form id="chat_form" action="#" method="post" accept-charset="utf-8">
  <p>
    <label>Message
    <input id="message" type="text" />
  </label>
  <input type="submit" value="Send">
  </p>
</form>
</div>

</body>
</html>

```

Als einzige Änderung an unserer Datei `chat.js` müssen wir eine Variable festlegen, die den Speicherort der `WebSocketMain`-Datei angibt:

```
html5_websockets/public/chat.js
```

```
WEB_SOCKET_SWF_LOCATION = "websocket_js/WebSocketMain.swf";
```

Damit funktioniert unsere Chat-Anwendung in allen wichtigen Browsern, vorausgesetzt, der Server für Ihren Chat-Server stellt auch eine Flash Socket Policy-Datei bereit.

Flash Socket Policy ähm, was ?

Aus Sicherheitsgründen kommuniziert Flash Player nur über Sockets mit Servern, die Verbindungen zu Flash Player zulassen. Flash Player versucht zunächst, auf Port 843 eine Flash Socket Policy-Datei abzurufen und anschließend auf demselben Port, den Ihr Server verwendet. Der Player erwartet eine Serverantwort wie die folgende:

```

<cross-domain-policy>
  <allow-access-from domain="*" to-ports="*" />
</cross-domain-policy>

```

Das ist eine sehr allgemein gehaltene Richtliniendatei, die es jedem erlaubt, sich mit diesem Service zu verbinden. Wenn Sie mit sensiblen Daten arbeiten, sollten Sie die Richtlinie natürlich etwas restriktiver gestalten. Denken Sie daran, diese Datei über denselben Server bereitzustellen, der auch Ihren Web Sockets-Server bereitstellt – entweder auf demselben Port oder auf Port 843.

Der Beispielcode für diesen Abschnitt enthält einen einfachen, in Ruby geschriebenen Flash Socket Policy-Server, den Sie zum Testen verwenden können. Im Abschnitt *Server* erfahren Sie mehr darüber, wie Sie Ihre eigene Testumgebung einrichten.

Chat-Server sind nur der Anfang. Mit Web Sockets haben wir nun eine stabile und einfache Möglichkeit, Daten an die Browser unserer Besucher zu schicken.

Server

Der Quellcode zu diesem Buch enthält eine Version des Web Sockets-Servers, mit dem wir kommunizieren. Er wurde in Ruby geschrieben, daher brauchen Sie einen Ruby-Interpreter. Anweisungen, wie Sie Ruby auf Ihrem System zum Laufen bekommen, finden Sie in der Datei `RUBY_README.txt` im Quellcode zu diesem Buch.

Sie können den Server starten, indem Sie zu dem entsprechenden Verzeichnis navigieren und Folgendes eingeben:

```
ruby server.rb
```

Neben dem Chat-Server gibt es noch zwei weitere Server, die Sie zum Testen der Beispiele aus diesem Kapitel verwenden können. Der erste Server, `client.rb`, stellt die Chat-Oberfläche und JavaScript-Dateien bereit. Der andere Server, `flashpolicyserver`, stellt eine Flash Policy-Datei bereit, die unser Flash-basierter Web Sockets-Ausweichcode benötigt, um eine Verbindung zum eigentlichen Chat-Server herzustellen. Der Flash Player ermittelt anhand dieser Richtliniendatei, ob er mit einer entfernten Domain kommunizieren darf.

Falls Sie auf einem Mac oder einem Linux-basierten Betriebssystem arbeiten, können Sie folgendermaßen alle Server vom Verzeichnis `html5_websockets` auf einmal starten:

```
rake start
```

26

Finden Sie sich selbst: Geolocation

Geolocation ist eine Technik, die dazu dient, über den Standort eines Computers zu ermitteln, wo sich bestimmte Personen befinden. „Computer“ kann dabei natürlich auch für ein Smartphone, Tablet-PC, andere tragbare Geräte oder einen Desktop-Rechner stehen. Bei der Geolocation wird der Aufenthaltsort einer Person anhand der IP- oder MAC-Adresse ihres Computers, des Standorts des Wi-Fi-Hotspots oder sogar anhand der GPS-Koordinaten bestimmt, falls diese verfügbar sind. Obwohl Geolocation streng genommen nicht Teil der HTML5-Spezifikation ist, wird sie trotzdem oft mit HTML5 gleichgesetzt, weil sie zur gleichen Zeit aufgetaucht ist. Im Gegensatz zu Web Storage war Geolocation nie Teil der HTML5-Spezifikation. Ebenso wie Web Storage ist Geolocation eine äußerst nützliche Technologie, die bereits in Firefox, Safari und Chrome implementiert ist. Sehen wir uns an, wie wir sie verwenden können.

AwesomeCo lokalisieren

Wir wurden gebeten, eine Kontaktseite für die Website von AwesomeCo zu erstellen. Der CIO hat uns gefragt, ob wir den Startort der Besucher zusammen mit den Support-Centern von AwesomeCo auf einer Karte anzeigen können. Er möchte unbedingt einen Prototyp sehen, also sollten wir das schnell zum Laufen bekommen.

Wir verwenden die Static Map API von Google, weil sie keinen API-Schlüssel erfordert und wir nur eine sehr einfache Karte erzeugen möchten.

Die Service-Center von AwesomeCo befinden sich in Portland (Oregon), Chicago (Illinois) und Providence (Rhode Island). Mit der Static Map API von Google ist es wirklich einfach, diese Punkte auf einer Karte darzustellen. Wir müssen lediglich ein `img`-Tag anlegen und die Adressen folgendermaßen in der URL angeben:

```
html5geo/index.html
```

```

```

Wir definieren die Größe des Bildes und teilen der Maps API anschließend mit den übergebenen Informationen mit, dass wir kein Sensorgerät wie etwa ein GPS oder eine clientseitige Geolocation verwendet haben. Dann definieren wir die einzelnen Markierungen auf der Karte, indem wir jeweils eine Beschriftung und die Adresse übergeben. Wir könnten auch die Koordinatenpaare für die Markierungen durch Kommata voneinander getrennt übergeben. Aber so ist es für unsere Demonstration einfacher.

Wie man gefunden wird

Für die aktuelle Position des Besuchers müssen wir eine weitere Markierung auf der Karte für seinen Breiten- und Längengrad anzeigen. Wir können einfach den Browser bitten, den Breiten- und Längengrad unseres Besuchers zu ermitteln:

html5geo/index.html

```
navigator.geolocation.getCurrentPosition(function(position) {
    showLocation(position.coords.latitude, position.coords.longitude);
});
```

Durch diese Methode wird der Benutzer aufgefordert, uns seine Koordinaten zur Verfügung zu stellen. Wenn der Benutzer die Erlaubnis erteilt, seine Standortinformationen zu verwenden, rufen wir die Methode `showLocation()` auf.

Die Methode `showLocation()` erstellt das Bild anhand des Breiten- und Längengrads neu und ersetzt die vorhandene Bildquelle durch die neue. Diese Methode implementieren wir folgendermaßen:

html5geo/index.html

```
1 var showLocation = function(lat, lng){
2     var fragment = "&markers=color:red|color:red|label:Y|" + lat + "," + lng;
3     var image = $("#map");
4     var source = image.attr("src") + fragment;
5     source = source.replace("sensor=false", "sensor=true");
6     image.attr("src", source);
7 };
```

Anstatt den gesamten Quellcode des Bilds zu duplizieren, fügen wir einfach den Breiten- und Längengrad unserer Position zur vorhandenen Quelle hinzu.

Bevor wir die veränderte Bildquelle wieder dem Dokument zuweisen, müssen wir den Parameter `sensor` von „false“ in „true“ ändern. Das machen wir in Zeile 5 mit der `replace()`-Methode.

Wenn wir die Seite in unserem Browser aufrufen, wird unser Standort zusätzlich zu den anderen Standorten mit einem „Y“ markiert. Ein Beispiel sehen Sie in Abbildung 10.3.

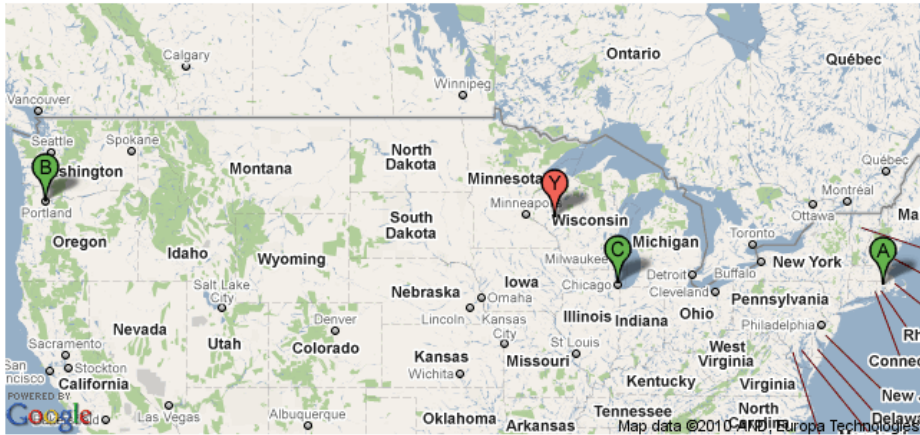


Abbildung 10.3: Unser aktueller Standort wird auf der Karte mit einem „Y“ markiert.

Ausweichlösung

Zum gegenwärtigen Zeitpunkt sehen Besucher auf der Seite auf jeden Fall die Karte mit den Standorten der Support-Center von AwesomeCo. Aber wir erhalten einen JavaScript-Fehler, wenn wir versuchen, unsere Seite zu laden. Wir müssen ermitteln, ob Geolocation unterstützt wird, bevor wir versuchen, den Standort des Besuchers zu bestimmen:

`html5geo/index.html`

```
if (navigator.geolocation) {
    navigator.geolocation.getCurrentPosition(function(position) {
        showLocation(position.coords.latitude, position.coords.longitude);
    });
}else{
};
```

Die Ajax API¹ von Google kann die Standortbestimmung übernehmen und ist daher eine ausgezeichnete Ausweichlösung. Um sie produktiv

1 <http://code.google.com/apis/ajax/documentation/#ClientLocation>

auf Ihrer Website zu verwenden, müssen Sie sich einen API-Schlüssel besorgen. Zum lokalen Testen brauchen Sie aber keinen.¹

So sieht unsere Ausweichlösung aus:

html5geo/index.html

```

1  var key = "your_key";
2  var script = "http://www.google.com/jsapi?key=" + key;
3  $.getScript(script, function(){
4      if ((typeof google == 'object') &&
5          google.loader && google.loader.ClientLocation) {
6          showLocation(google.loader.ClientLocation.latitude,
7                      google.loader.ClientLocation.longitude);
8      }else{
9          var message = $("<p>Couldn't find your address.</p>");
10         message.insertAfter("#map");
11     };
12 });

```

Wir verwenden die `getScript()`-Methode von jQuery, um die Google Ajax API zu laden. Anschließend verwenden wir in Zeile 5 Googles `ClientLocation()`-Methode, um den Standort unserer Besucher zu bestimmen, und rufen die Methode `showLocation()` auf, um den Standort auf unserer Karte anzuzeigen.

Leider kann Google nicht jede IP-Adresse geolokalisieren. Unter Umständen sind wir also trotzdem nicht in der Lage, den Benutzer auf der Karte anzuzeigen. In diesem Fall zeigen wir mit dem Code aus Zeile 9 unterhalb des Bilds eine Meldung an. Unsere Ausweichlösung ist nicht wasserdicht, gibt uns aber eine größere Wahrscheinlichkeit, unsere Besucher zu lokalisieren.

Ohne eine zuverlässige Methode, die Koordinaten des Clients zu ermitteln, müssen wir den Benutzern die Möglichkeit geben, ihre Adresse anzugeben. Aber diese Übung überlasse ich Ihnen.

Die Zukunft

Die Techniken, die wir in diesem Kapitel besprochen haben, sind zwar nicht alle fester Bestandteil von HTML5, repräsentieren aber die Zukunft der Webentwicklung. Wir werden noch viel mehr auf die Seite des Clients verlagern. Mit einer besseren Verlaufsverwaltung werden Ajax- und clientseitige Anwendungen wesentlich intuitiver. Web Sockets können bei der Anzeige von Echtzeitdaten das regelmäßige Pol-

¹ Auch für <http://localhost/> brauchen Sie einen Schlüssel. Sie bekommen ihn unter <http://code.google.com/apis/ajaxsearch/signup.html>.

ling von entfernten Diensten ersetzen. Mit Cross-Document Messaging können wir Webanwendungen zusammenführen, die sonst nie miteinander interagieren könnten. Und mit Geolocation können wir irgendwann standortbezogene Webanwendungen schreiben, die mit dem wachsenden Markt für mobile Computeranwendungen jeden Tag wichtiger werden.

Durchforsten Sie diese APIs, und verfolgen Sie mit, wie sie sich verbreiten. Vielleicht gehören diese wertvollen Tools schon bald zu Ihrer Werkzeugkiste als Webentwickler.

Kapitel 11

Wie es weitergeht

Der größte Teil dieses Buchs konzentriert sich auf Funktionen, die Sie bereits heute nutzen können. Aber es gibt noch einige andere Dinge, die Sie sehr bald verwenden können und die die standardbasierte Webentwicklung sogar noch interessanter machen – von der Unterstützung von 3D-Canvas mit WebGL bis hin zu neuen Storage APIs, CSS3-Übergängen und zur nativen Unterstützung von Drag-and-Drop. In diesem Kapitel besprechen wir einige Neuerungen, die sich am Horizont abzeichnen. Dadurch sollen Sie einen Eindruck bekommen, worauf Sie sich sonst noch freuen können. Wir zeigen Ihnen Funktionen, die Sie in mindestens einem Browser verwenden können, für die es aber noch keine ausreichenden Ausweichlösungen gibt oder die bei Weitem noch nicht klar genug umrissen sind, dass Sie damit bereits jetzt arbeiten könnten:¹

CSS3-Übergänge

Animationen bei Interaktion. *[C3, S3.2, F4, O10.5, IOS3.2, A2]*

Web Workers

Hintergrundverarbeitung von JavaScript. *[C3, S4, F3.5, O10.6]*

3D-Canvas mit WebGL

3D-Objekte mit dem canvas-Element. *[C5, F4]*

¹ In den folgenden Beschreibungen wird die Unterstützung durch die verschiedenen Browser in eckigen Klammern mit einem Kurzcode und der mindestens erforderlichen Versionsnummer angegeben. Die verwendeten Codes lauten: *C*: Google Chrome, *F*: Firefox, *IE*: Internet Explorer, *O*: Opera, *S*: Safari, *IOS*: iOS-Geräte mit Mobile Safari und *A*: Android-Browser.

IndexedDB

Fortschrittliche clientseitige Schlüssel/Wert-Datenbankspeicherung, ähnlich wie NoSQL-Lösungen. [F4]

Drag and Drop

API für interaktives Drag-and-Drop. [C3, S4, F3.5, IE6, A2]

Form validation

Clientseitige Validierung von Benutzereingaben. [C5, S5, 10.6]

Wir fangen mit den CSS3-Übergängen an und schauen, wie wir sie in WebKit-Browsern verwenden können.

11.1 CSS3-Übergänge

Aufforderungen zur Interaktion sind wichtig für eine positive Erfahrung Ihrer Besucher. CSS unterstützt schon seit einiger Zeit die Pseudoklasse `:hover`, damit wir einige grundlegende Hinweise für die Interaktion mit unseren Elementen geben können. Hier sehen Sie das CSS-Markup, mit dem Sie einen Link so stylen, dass er wie eine Schaltfläche aussieht:

`css3transitions/style.css`

```
a.button{
  padding: 10px;
  border: 1px solid #000;
  text-decoration: none;
}
a.button:hover{
  background-color: #bbb;
  color: #fff
}
```

Wenn wir den Mauszeiger über die Schaltfläche bewegen, ändert sich die Hintergrundfarbe von Weiß in Grau und die Textfarbe von Schwarz in Weiß. Der Übergang ist stufenlos. Mit CSS3-Übergängen² können wir eine ganze Menge mehr machen, bis hin zu einfachen Animationen, die bisher nur mit JavaScript möglich waren. Indem wir den folgenden hervorgehobenen Code zu unserer Stildefinition hinzufügen, können wir beispielsweise diesen Übergang als Überblendung gestalten:

² <http://dev.w3.org/csswg/css3-transitions/>

css3transitions/style.css

```

1  a.button{
2      padding: 10px;
3      border: 1px solid #000;
4      text-decoration: none;
5      ▶ -webkit-transition-property: background-color, color;
6      ▶ -webkit-transition-duration: 1s;
7      ▶ -webkit-transition-timing-function: ease-out;
8  }
9
10 a.button:hover{
11     background-color: #bbb;
12     color: #fff
13 }

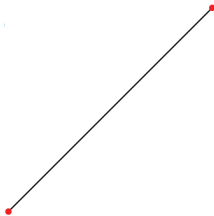
```

In Zeile 5 geben wir an, welche Eigenschaften auf den Übergang angewendet werden. In diesem Fall ändern wir die Hintergrund- und Vordergrundfarbe. In Zeile 6 geben wir die Dauer der Animation an, in Zeile 7 die Timingfunktion für den Übergang.

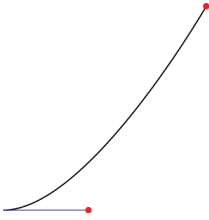
Timingfunktionen

Die Eigenschaft `transition-timing-function` beschreibt, wie die Übergänge zeitlich in Relation zur festgelegten Dauer erfolgen. Wir geben die Timingfunktion mit einer kubischen Bezier-Kurve an, die durch vier Stellpunkte auf einem Graph definiert wird. Jeder Punkt hat einen X- und einen Y-Wert zwischen 0 und 1. Der erste und der letzte Stellpunkt habt immer den Wert (0.0,0.0) und (1.0,1.0), während die beiden mittleren Punkte die Form der Kurve bestimmen.

Die Stellpunkte einer linearen Kurve entsprechen den beiden Endpunkten, wodurch sich eine gerade Linie in einem 45-Grad-Winkel ergibt. Die vier Punkte einer linearen Kurve lauten ((0.0, 0.0), (0.0,0.0), (1.0, 1.0), (1.0, 1.0)). Und so sieht sie aus:

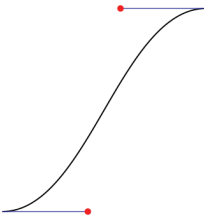


Eine etwas komplexere Kurve mit den Punkten $(0.0, 0.0)$, $(0.42, 0.0)$, $(1.0, 1.0)$, $(1.0, 1.0)$, eine sogenannte *Ease-in*-Kurve, sieht so aus:



Nur der zweite Punkt ist anders, wodurch der untere linke Teil der Linie gebogen ist.

Die *Ease-in-out*-Kurve ist noch komplexer, hier gibt es oben und unten eine Kurve:



Die Punkte dieser Kurve lauten $(0.0, 0.0)$, $(0.42, 0.0)$, $(0.58, 1.0)$, $(1.0, 1.0)$.

Wir können die Punkte direkt in der CSS-Eigenschaft angeben oder wie in unserem Beispiel vordefinierte Punkte verwenden.

Zur Wahl stehen die Typen `default`, `ease-in`, `ease-out`, `ease-in-out`, `ease-out-in` und `cubic-bezier`. Bei Letzterem legen Sie die Kurvenpunkte selbst fest.

Wenn der Übergang konstant verlaufen soll, verwenden Sie den Typ `linear`. Wenn die Animation langsam beginnen und dann schneller verlaufen soll, wählen Sie `ease-in`. Falls Sie ein bisschen mehr über die Gestaltungen solcher Kurven lernen möchten: Es gibt ein tolles Public Domain-Skript³ mit Beispielen, das Ihnen hilft, die Koordinaten zu erkennen.

Spielen Sie mit den Übergängen. Aber denken Sie daran, dass Ihre Oberfläche in erste Linie benutzerfreundlich und erst in zweiter Linie hübsch anzusehen sein soll. Erstellen Sie keine Übergänge, die Ihre

3 <http://www.netzgesta.de/dev/cubic-bezier-timing-function.html>

Benutzer frustrieren, wie etwa blinkende Elemente oder solche, deren Animationen zu lange dauern. Sie können auch einen Blick auf die CSS3-Animationen⁴ werfen – eine weitere Möglichkeit, CSS-Eigenschaften in Relation zu einem zeitlichen Rahmen zu ändern.

11.2 Web Workers

Web Workers⁵ sind nicht Teil der HTML5-Spezifikation. Sie können sich aber als nützlich erweisen, um Prozesse clientseitig im Hintergrund verarbeiten zu lassen. Deshalb sind sie erwähnenswert.

Wir verwenden JavaScript für unseren gesamten clientseitigen Code, aber JavaScript unterstützt immer nur einen Thread. Wir können immer nur eine Sache gleichzeitig machen. Wenn eine Aufgabe viel Zeit in Anspruch nimmt, sind die Benutzer gezwungen zu warten, bis die Aufgabe erledigt ist. Web Workers lösen dieses Problem, indem sie eine einfache Möglichkeit bieten, gleichzeitig ablaufende Programme zu schreiben.

Wenn wir ein Skript mit dem Name `worker.js` verwenden, das Bilder verarbeitet, können wir es folgendermaßen aufrufen:

webworkers/application.js

```
var worker = new Worker("worker.js");
```

Jede beliebige JavaScript-Datei kann als Worker gestartet werden. Damit der Worker aber unabhängig ablaufen kann, darf er nicht auf das DOM zugreifen. Das bedeutet, dass Sie Elemente nicht direkt manipulieren können.

Unser Hauptskript kann aber über `postMessage()` Nachrichten an das Worker-Skript senden:

webworkers/application.js

```
$("#button").click(function(event){
    $("#output").html("starting...");
    worker.postMessage("start");
});
```

Unser Worker-Skript kann seinerseits Nachrichten an die Hauptseite zurücksenden, und zwar ebenfalls über die `postmessage()`-Methode.

4 <http://www.w3.org/TR/css3-animations/>

5 <http://www.whatwg.org/specs/web-workers/current-work/>

webworkers/worker.js

```
onmessage = function(event) {
  if(event.data === "start"){
    // Diese Schleife zählt nur. Machen Sie lieber was Spannendes.
    for (var x = 1; x <= 100000; x++){
      postMessage(x);
    }
  }
};
```

Auf diese Events können wir reagieren, indem wir in unserem Hauptskript auf das `onmessage`-Event lauschen. Jedes Mal, wenn der Worker eine Nachricht zurücksendet, wird der folgende Code ausgeführt:

webworkers/application.js

```
worker.onmessage = function(event){
  $("#output").html(event.data);
}
```

Diese API funktioniert genau wie die API für das Cross-Domain Messaging, über die wir im Abschnitt *Über Domains hinweg kommunizieren* auf Seite 206 gesprochen haben. Es gibt keine Unterstützung für Web Workers im Internet Explorer, also müssen Sie auf Google Chrome Frame zurückgreifen. Aber falls Sie einige ernst zu nehmende clientseitige Aufgaben ohne eine Blockierung der Benutzer abarbeiten müssen, lohnt es sich, tiefer einzusteigen.

11.3 Native Unterstützung für Drag-and-Drop

Das Drag-and-Drop von Oberflächenelementen können wir unseren Benutzern schon eine ganze Weile mit JavaScript-Bibliotheken ermöglichen. Aber mittlerweile hat das W3C die Drag-and-Drop-Implementierung von Microsoft als Teil der HTML5-Spezifikation übernommen.⁶ Sie wird von Firefox, Safari, Internet Explorer und Chrome unterstützt, ist aber faktisch das reinste Chaos.

Auf den ersten Blick scheint die Implementierung unkompliziert zu sein. Wir bezeichnen ein Element als „draggable“, bestimmen ein Element, das auf „fallen gelassene“ Elemente wartet, und führen einen bestimmten Code aus, wenn das geschieht.

In der Realität ist es nicht mal annähernd so einfach. Wir demonstrieren das an einer einfachen Drag-and-Drop-Oberfläche, auf der wir kleine Bilder in einen Ablagebereich ziehen können, in dem daraufhin die größere Version geladen wird.

6 <http://dev.w3.org/html5/spec/dnd.html#dnd>

html5drag/index.html

```

<div id="images">
  
  
  

</div>

<div id="preview">
  <p>Drop images here</p>
</div>

```

Wir verwenden benutzerdefinierte Datenattribute, um die Quelle der größeren Fotos abzulegen.

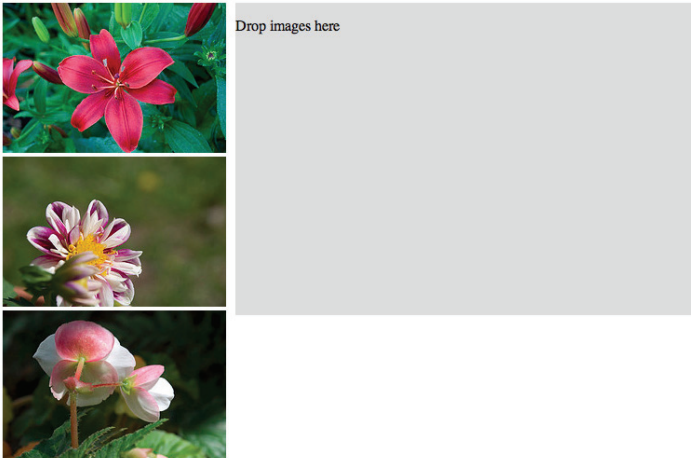


Abbildung 11.1: Unsere Fotoanzeige

Als Nächstes setzen wir einige einfache Stilregeln ein, um die beiden Spalten zu floaten:

html5drag/style.css

```

#images img{
  -webkit-user-drag
}

#images{
  float: left;
  width: 240px;
  margin-right: 10px;
}

```

```
#preview{
  float: left;
  width: 500px;
  background-color: #ddd;
  height: 335px;
}

.hover{
  border: 10px solid #000;
  background-color: #bbb !important;
}
```

Zu diesem Zeitpunkt sieht unsere Oberfläche wie in Abbildung 11.1 auf Seite 231 aus. Nun fügen wir einige Events hinzu, damit wir die Fotos ziehen und ablegen können.

Drag-and-Drop-Events

Beim Ziehen und Ablegen von Elementen müssen wir mit mehreren Events arbeiten.

Event	Beschreibung
ondragstart	Wird ausgelöst, wenn der Benutzer damit beginnt, das Objekt zu ziehen
ondragend	Wird ausgelöst, wenn der Benutzer aus <i>einem beliebigen Grund</i> das Ziehen des Objekts beendet
ondragenter	Wird ausgelöst, wenn ein „draggable“ Element in einen Ablege-Listener bewegt wird
ondragover	Wird ausgelöst, wenn der Benutzer ein Element über einen Ablege-Listener zieht
ondragleave	Wird ausgelöst, wenn der Benutzer ein Element aus einem Ablege-Listener zieht
ondrop	Wird ausgelöst, wenn der Benutzer ein Element erfolgreich in einem Ablege-Listener ablegt
ondrag	Wird ausgelöst, wenn der Benutzer ein Element an eine beliebige Stelle zieht; wird konstant ausgelöst und liefert die X- und Y-Koordinaten des Mauszeigers.

Das sind insgesamt sieben Events, nur für das Ziehen und Ablegen von Elementen. Und für einige der Events gibt es ein Standardverhalten. Wenn wir dieses nicht überschreiben, geht alles schief.

Zunächst müssen wir alle unsere Listenelemente als „draggable“ definieren:

```
html5drag/application.js
```

```
var contacts = $('#images img');
contacts.attr('draggable', 'true');
```

Wir fügen einfach das HTML5-Attribut `draggable` zu den Elementen hinzu. Wir könnten das auch im Markup machen, aber nachdem wir für die Interaktion ohnehin JavaScript benötigen, wenden wir das Attribut mit unserem Skript an.

Wenn wir das Bild ziehen, möchten wir die Adresse des großen Bilds abrufen und speichern. Wir stellen dafür eine Verknüpfung mit dem `ondragstart`-Event her. Damit das einfach und plattformübergreifend funktioniert, verwenden wir die `bind()`-Methode von jQuery.⁷

```
html5drag/application.js
```

```
1 contacts.bind('dragstart', function(event) {
2     var data = event.originalEvent.dataTransfer;
3     var src = $(this).attr("data-large");
4     data.setData("Text", src);
5     return true;
6 });
```

Die Spezifikation bietet einen `dataStorage`-Mechanismus, mit dem wir den Datentyp und die Daten selbst angeben können, die als Teil des Events übergeben werden. Die `bind()`-Methode von jQuery verpackt das Event in ein eigenes Objekt, deshalb verwenden wir in Zeile 2 wieder die Eigenschaft `originalEvent`, um auf das eigentliche Event zuzugreifen. In Zeile 4 speichern wir mit der Methode `setData()` die URL des Bildes im Event ab. Dafür verwenden wir den Datentyp `Text`.

Nachdem wir jetzt Elemente ziehen können, geht es darum, wie wir Events auslösen, wenn der Benutzer Elemente abgelegt.

Elemente ablegen

Wir möchten, dass das Feld „preview“ als Ziel zum Ablegen dient. Also suchen wir das Feld und verknüpfen es mit dem `drop`-Event.

```
html5drag/application.js
```

```
1 var target = $('#preview');
2
3 target.bind('drop', function(event) {
4     var data = event.originalEvent.dataTransfer;
5     var src = ( data.getData('Text') );
```

⁷ Beachten Sie, dass wir das `on`-Präfix der Events weglassen, wenn wir diese Methode verwenden.

```

6
7   var img = $("<img></img>" .attr("src", src);
8   $(this).html(img);
9   if (event.preventDefault) event.preventDefault();
10  return(false);
11 });

```

In Zeile 5 lesen wir mit der Methode `getData()` die übergebene Bildadresse aus und erstellen anschließend ein neues Bildelement, das wir in unseren Inhaltsbereich schieben.

Wir müssen das standardmäßige `ondrop`-Event abbrechen, damit es nicht ausgelöst wird, wenn der Benutzer das Element auf dem Ziel ablegt. Dazu müssen wir sowohl `preventdefault()` als auch `return false` einsetzen. Für den Internet Explorer brauchen wir `return false`, für die anderen Browser `preventDefault()`.

Wenn wir das direkt in Chrome oder Safari testen, funktioniert es nicht so recht. Wir müssen mindestens das `ondragover`-Element überschreiben. Sonst reagiert unser `ondrag`-Event nicht. Dazu verwenden wir den folgenden Code:

html5drag/application.js

```

target.bind('dragover', function(event) {
    if (event.preventDefault) event.preventDefault();
    return false;
});

```

Auch hier brechen wir wieder das Standard-Event ab, wie schon beim `ondrop`-Event. Dasselbe machen wir für das `ondragend`-Event.

html5drag/application.js

```

contacts.bind('dragend', function(event) {
    if (event.preventDefault) event.preventDefault();
    return false;
});

```

Dadurch werden alle Browser-Events unterbunden, die standardmäßig ausgelöst werden, wenn der Benutzer mit dem Ziehen eines Elements aufhört. Das von uns definierte `ondrop`-Event wird dadurch jedoch nicht beeinflusst.

Stilregeln ändern

Wir möchten die Benutzer darauf hinweisen, wenn sie ein Element über das Ablageziel ziehen. Das können wir über die Methoden `ondragenter` und `ondragleave` machen.

html5drag/application.js

```
target.bind('dragenter', function(event) {
    $(this).addClass('hover');
    if (event.preventDefault) event.preventDefault();
    return false;
});

target.bind('dragleave', function(event) {
    $(this).removeClass('hover');
    if (event.preventDefault) event.preventDefault();
    return false;
});
```

Durch diesen Code wird beim Auslösen der beiden Events die hover-Klasse aus unserem Stylesheet jeweils auf unser Zielelement angewendet bzw. entfernt.

Dateien ziehen

Das Ziehen von Text und Elementen auf der Seite ist nur der Anfang. Laut Spezifikation können Entwickler auch Oberflächen erstellen, die Dateien vom Computer des Benutzers per Drag-and-Drop in Empfang nehmen. Um ein Bild hochzuladen oder eine Datei anzuhängen, muss die Datei nur mit der Maus auf einem bestimmten Ziel abgelegt werden. Googles Gmail unterstützt das beispielsweise, wenn Sie Firefox 3.6 oder Chrome 5 verwenden.

Wenn Sie tiefer in dieses Thema einsteigen möchten, sollten Sie den ausgezeichneten Artikel⁸ von Leslie Michael Orchard lesen.

Nicht alles ist gut

Das Verhalten in den verschiedenen Browsern ist – vorsichtig ausgedrückt – inkonsistent. Mit dem IE 8 funktioniert es, aber nur, solange wir den Datentyp für setData() nicht auf Url statt auf Text festlegen.

Um außerdem in Safari 4 auch Drag-and-Drop für andere Elemente als Bilder oder Links zu unterstützen, müssen wir das folgende zusätzliche CSS in unser Stylesheet einfügen:

```
#contents li{
    -webkit-user-drag
}
```

8 <http://decafbad.com/blog/2009/07/15/html5-drag-and-drop>

In diesem Buch haben wir konsequent dargestellt, wie wichtig es ist, Stilregeln und Inhalt getrennt voneinander zu halten. Der obige Code läuft diesem Konzept eindeutig zuwider.

Versuchen Sie nicht, Text in Formularfelder zu ziehen. Moderne Browser unterstützen das zwar bereits, aber es gibt keine gute Möglichkeit, dieses Verhalten zu überschreiben.

Nach dem aktuellen Stand erzielen wir mit viel weniger Code wesentlich bessere Resultate, wenn wir eine JavaScript-Bibliothek verwenden, die Drag-and-Drop unterstützt, zum Beispiel jQuery UI.⁹

Selbst mit einer solchen Bibliothek bleibt aber immer noch ein Punkt problematisch: Barrierefreiheit. Die Spezifikation sagt nichts dazu, wie wir mit Benutzern umgehen sollen, die keine Maus bedienen können. Wenn wir für unsere Oberflächen eine Drag-and-Drop-Funktionalität implementieren, müssen wir auch eine zweite Möglichkeit entwickeln, für die kein JavaScript und keine Maus erforderlich ist. Und diese Methode hängt wiederum davon ab, was wir erreichen möchten.

Die Spezifikation bietet eine Menge Potenzial, für manche Punkte besteht aber noch Klärungsbedarf. Verwenden Sie sie, soweit es Sinn macht. Aber zwingen Sie Ihren Benutzern unter keinen Umständen etwas auf, das sie nicht verwenden können.

11.4 WebGL

Wir haben in diesem Buch über den 2D-Kontext des canvas-Elements gesprochen. Aber es ist eine andere Spezifikation unterwegs, die beschreibt, wie Sie mit 3D-Objekten arbeiten können. Die WebGL-Spezifikation¹⁰ ist kein Teil von HTML5, aber Apple, Google, Opera und Mozilla sind Teil der Arbeitsgruppe und haben eine gewisse Unterstützung in ihren Browsern implementiert.

Die Arbeit mit 3D-Grafik geht weit über den Rahmen dieses Buchs hinaus, aber auf der Website „Learning WebGL“¹¹ finden Sie tolle Beispiele und Anleitungen.

⁹ <http://docs.jquery.com/UI/Draggable>

¹⁰ <http://cvs.khronos.org/svn/repos/registry/trunk/public/webgl/docs/specs/WebGL-spec.html>

¹¹ <http://learningwebgl.com/blog/?p=11>

11.5 Indexed Database API

Wir haben in diesem Buch über zwei Methoden zum Speichern von Daten auf dem Client gesprochen: Web Storage und Web SQL Storage. Die Mozilla Foundation war mit der Web SQL-Spezifikation nicht einverstanden, weil sie es für keine gute Idee hielt, die Spezifikation auf eine bestimmte SQL-Engine zu stützen. Sie hat eine neue Spezifikation mit dem Namen Indexed Database API eingeführt, die als eigener Standard geplant ist.¹²

Die Indexed Database API ist ähnlich wie die Web Storage APIs, zum Beispiel `localStorage` und `sessionStorage`, ein Schlüssel-/Wertspeicher, bietet aber Methoden für die Ausführung erweiterter Abfragen. Als dieses Buch geschrieben wurde, gab es leider noch keine Implementierungen dieser Spezifikation. Es lohnt sich daher nicht, auf Implementierungsdetails einzugehen, weil sich diese wahrscheinlich bis zum Zeitpunkt der endgültigen Implementierung noch ändern. Firefox 4 und Chrome 7 sollen die Spezifikation wohl unterstützen.

Diese Spezifikation sollten Sie im Auge behalten, weil sich Web SQL in einer Sackgasse befindet und Mozilla mehrfach bekannt gegeben hat, dass es keine Pläne gibt, WebSQL jemals in Firefox zu implementieren. Mozilla fühlt sich nicht wohl mit dem SQL-Dialekt und der Vorstellung, dass die Spezifikation auf einer bestimmten Datenbankimplementierung basiert. Die Web SQL-Spezifikation verwendet den Dialekt der SQLite-Datenbank, die sich unabhängig von der Spezifikation ändern kann. Es ist sehr wahrscheinlich, dass die Spezifikation in den Internet Explorer implementiert wird, weil Microsoft Interesse an ihrer Entwicklung hat.¹³

The image shows a web form titled "Project Information". It contains the following fields and controls:

- Name:** A text input field.
- Priority:** A text input field with a red-bordered warning box overlaid on it. The warning box contains the text "You have to specify a value".
- Estimated Hours:** A text input field with a small up/down arrow control to its right.
- Start date:** A date/time picker showing "2010-12-01" with a calendar icon to its right.
- Email contact:** A text input field.

Abbildung 11.2: Opera zeigt eine hervorgehobene Warnung.

¹² <http://www.w3.org/TR/IndexedDB/>

¹³ <http://hacks.mozilla.org/2010/06/beyond-htm15-database-apis-and-the-road-to-indexeddb/>

11.6 Clientseitige Formularvalidierung

Die HTML-Spezifikation sieht mehrere Attribute vor, über die wir clientseitig Benutzereingaben validieren können. Auf diese Weise können wir einfache Eingabefehler abfangen, bevor der Benutzer die Anfrage an den Server schickt. Mit JavaScript geht das schon seit Jahren, aber in HTML5-Formularen können wir jetzt neue Attribute verwenden, um das gewünschte Verhalten festzulegen.

Mit dem Attribut `required` können wir sicherstellen, dass Benutzer erforderliche Formularfelder ausfüllen:

```
html5validation/index.html
```

```
<label for="name">Name</label>
<input type="text" name="name" autofocus required id="name">
```

Falls das Feld leer ist, kann der Browser die Übermittlung des Formulars verhindern und eine hübsche Fehlermeldung anzeigen. Und wir müssen dafür nicht eine einzige Zeile JavaScript schreiben. Opera macht das bereits, wie Sie in Abbildung 11.2 sehen können.

Damit scheitern Benutzer früher und müssen nicht erst auf eine Serverantwort warten, um herauszufinden, dass sie einen Fehler gemacht haben. Dieses Verhalten kann natürlich deaktiviert, nicht verfügbar oder einfach nicht korrekt implementiert sein. Entsprechend müssen Sie unbedingt für eine serverseitige Strategie für die Datenvalidierung sorgen. Sie sollten sich definitiv schon jetzt Gedanken darüber machen, weil Sie dann auf einfache Weise die erforderlichen Felder ausfindig machen und die Oberfläche mit CSS so stylen können, dass sich diese Felder von den übrigen entsprechend abheben.

Mit dem Attribut `pattern` können Sie noch einen Schritt weiter gehen. Damit können Sie einen regulären Ausdruck mit den Kriterien angeben, denen der Inhalt entsprechen muss.

```
html5validation/index.html
```

```
<input type="text" name="acctnumber" id="acctnumber" required
      pattern="[1-9]+[0-9]*$">
```

Zwar wird das von keinem aktuellen Browser unterstützt, aber es wäre wirklich einfach, dieses Markup als Basis für eine JavaScript-Validierungsbibliothek zu verwenden.

11.7 Vorwärts!

Wir leben in einer spannenden Zeit für Entwickler. Dieses Buch kratzt gerade mal an der Oberfläche dessen, was die Zukunft für Entwickler bereithält. Es steckt noch so viel mehr in den Spezifikationen, und ich möchte Sie ermuntern, tiefer einzusteigen. Ich hoffe, dass Sie die Dinge anwenden, die Sie hier gelernt haben, und weiterhin basteln und erforschen – und dabei die verschiedenen Spezifikationen mitverfolgen.

Und jetzt legen Sie los, und schreiben Sie etwas Tolles!

Anhang A

Kurzreferenz

In den folgenden Beschreibungen wird die Unterstützung durch die jeweiligen Browser in eckigen Klammern mit einem Kurzcode für den Browser zusammen mit der mindestens erforderlichen Versionsnummer angegeben. Die Codes lauten: *C* für Google Chrome, *F* für Firefox, *IE* für Internet Explorer, *O* für Opera, *S* für Safari, *IOS* für iOS-Geräte mit Mobile Safari und *A* für Android-Browser.

A.1 Neue Elemente

Verwendet in *Einen Blog mit semantischem Markup neu definieren* auf Seite 26.

`<header>`

Definiert den Kopfbereich einer Seite oder eines Abschnitts.
[C5, F3.6, IE8, S4, O10]

`<footer>`

Definiert den Fußbereich einer Seite oder eines Abschnitts.
[C5, F3.6, IE8, S4, O10]

`<nav>`

Definiert einen Navigationsbereich einer Seite oder eines Abschnitts. [C5, F3.6, IE8, S4, O10]

`<section>`

Definiert einen logischen Bereich einer Seite oder eine Gruppierung von Inhalten. [C5, F3.6, IE8, S4, O10]

`<article>`

Definiert einen Artikel oder einen abgeschlossenen Inhaltsteil.
[C5, F3.6, IE8, S4, O10]

<aside>

Definiert sekundäre oder verwandte Inhalte.
[C5, F3.6, IE8, S4, O10]

<meter>

Beschreibt einen Wert innerhalb eines Bereichs.
[C5, F3.5, S4, O10]

<progress>

Steuerelement, das den Fortschritt zu einem Ziel hin in Echtzeit darstellt. *[Zum Zeitpunkt der Veröffentlichung nicht unterstützt.]*

A.2 Attribute

Benutzerdefinierte Datenattribute

Mit dem data-Muster können benutzerdefinierte Datenattribute zu beliebigen Elementen hinzugefügt werden. *[Alle Browser unterstützen das Auslesen der Attribute über die JavaScript-Methode `getAttribute()`.]*

Verwendet im Abschnitt *Pop-up-Fenster mit benutzerdefinierten Datenattributen* auf Seite 39.

In-Place-Editing [`<p contenteditable>lorem ipsum</p>`]

Unterstützung von In-Place-Editing von Inhalten im Browser.
[C4, S3.2, IE6, O10.1]

Verwendet im Abschnitt *In-Place-Editing mit contenteditable* auf Seite 65.

A.3 Formulare

Verwendet im Abschnitt *Daten mit neuen Eingabefelder beschreiben* auf Seite 48.

E-Mail-Feld [`<input type="email">`]

Zeigt ein Formularfeld für E-Mail-Adressen an. [O10.1, IOS]

URL-Feld [`<input type="url">`]

Zeigt ein Formularfeld für URLs an. [O10.1, IOS]

Telefonfeld [`<input type="tel">`]

Zeigt ein Formularfeld für Telefonnummern an. [O10.1, IOS]

Suchfeld [`<input type="search">`]

Zeigt ein Formularfeld für Suchwörter an. [C5, S4, O10.1, IOS]

Slider (Bereich) [`<input type="range">`]

Zeigt ein Slider-Steuerelement an. [C5, S4, O10.1]

Zahlen [`<input type="number">`]

Zeigt ein Formularfeld für Zahlen an, häufig eine Spinbox.
[C5, S5, O10.1, IOS]

Datumsfelder [`<input type="date">`]

Zeigt ein Formularfeld für Datumswerte an. Unterstützt `date`, `month` oder `week`. [C5, S5, O10.1]

Datumswerte mit Uhrzeit [`<input type="datetime">`]

Zeigt ein Formularfeld für Datumswerte mit Uhrzeit an. Unterstützt `datetime`, `datetime-local` oder `time`. [C5, S5, O10.1]

Farbe [`<input type="color">`]

Zeigt ein Feld zum Auswählen von Farben an. [C5,S5] (Chrome 5 und Safari 5 verstehen das Color-Feld, zeigen aber kein spezielles Steuerelement an.)

A.4 Attribute für Formularfelder

Autofokus-Unterstützung [`<input type="text" autofocus>`]

Ermöglicht das Festlegen des Fokus auf ein bestimmtes Formularelement. [C5, S4]

Verwendet im Abschnitt *Mit autofocus zum ersten Feld springen* auf Seite 56.

Unterstützung für Platzhalter [`<input type="email" placeholder="me@example.com">`]

Unterstützt die Anzeige von Platzhaltertext in einem Formularfeld. [C5, S4, F4]

Verwendet im Abschnitt *Tipps mit Platzhaltertext* auf Seite 58.

`required` [`<input type="email" required >`]

Macht ein Feld zum Pflichtfeld. [C5, S5, O10.6]

Verwendet im Abschnitt 11.6, *Clientseitige Formularvalidierung*, auf Seite 238.

`pattern` [`<input type="text" pattern="^[1-9]+[0-9]*$">`]

Validiert Formularfelddaten anhand eines bestimmten regulären Ausdrucks. [C5, S5, O10.6]

Verwendet im Abschnitt 11.6, *Clientseitige Formularvalidierung*, auf Seite 238.

A.5 Barrierefreiheit

`role` [`<div role="document">`]

Kennzeichnet die Zuständigkeit eines Elements für Screenreader. [C3, F3.6, S4, IE8, O9.6]

Verwendet im Abschnitt *Navigationshinweise mit ARIA-Rollen* auf Seite 101.

`aria-live` [`<div aria-live="polite">`]

Kennzeichnet einen Bereich, der automatisch aktualisiert wird, zum Beispiel über Ajax. [F3.6 (Windows), S4, IE8]

Verwendet im Abschnitt *Barrierefreie aktualisierbare Bereiche erstellen* auf Seite 106.

`aria-atomic` [`<div aria-live="polite" aria-atomic="true">`]

Gibt an, ob der gesamte Inhalt eines Live-Bereichs vorgelesen werden soll oder nur die Elemente, die sich verändert haben. [F3.6 (Windows), S4, IE8]

Verwendet im Abschnitt *Barrierefreie aktualisierbare Bereiche erstellen* auf Seite 106.

A.6 Multimedia

`<canvas>` [`<audio src="drums.mp3"></audio>`]

Unterstützt die Erstellung vektorbasierter Grafiken mit JavaScript. [C4, F3, IE9, S3.2, O10.1, IOS3.2, A2]

Verwendet in Kapitel 6, *Zeichnen mit dem canvas-Element*, auf Seite 113.

`<audio>` [`<audio src="drums.mp3"></audio>`]

Audio nativ im Browser wiedergeben. [C4, F3.6, IE9, S3.2, O10.1, IOS3, A2]

Verwendet im Abschnitt *Mit Audio arbeiten* auf Seite 138.

`<video>` [`<video src="tutorial.m4v"></video>`]

Native Wiedergabe von Video im Browser. [C4, F3.6, IE9, S3.2, O10.5, IOS3, A2]

Verwendet im Abschnitt *Video einbetten* auf Seite 142.

A.7 CSS3

Verwendet in Abschnitt 11.1, *CSS3-Übergänge*, auf Seite 226.

`:nth-of-type [p:nth-of-type(2n+1){color:red;}]`

Findet alle n Elemente eines bestimmten Typs.

[C2, F3.5, S3, IE9, O9.5, IOS]

Verwendet im Abschnitt *Tabellen mit Pseudoklassen stylen* auf Seite 75.

`:first-child [p:first-child{color:blue;}]`

Findet das erste Kindelement. [C2, F3.5, S3, IE9, O9.5, IOS3, A2]

Verwendet im Abschnitt *Tabellen mit Pseudoklassen stylen* auf Seite 75.

`:nth-child [p:nth-child(2n+1){color:red;}]`

Findet ein bestimmtes Kindelement, von vorne gesucht.

[C2, F3.5, S3, IE9, O9.5, IOS3, A2]

Verwendet im Abschnitt *Tabellen mit Pseudoklassen stylen* auf Seite 75.

`:last-child [p:last-child{color:blue;}]`

Findet das letzte Kindelement. [C2, F3.5, S3, IE9, O9.5, IOS3, A2]

Verwendet im Abschnitt *Tabellen mit Pseudoklassen stylen* auf Seite 75.

`:nth-last-child [p:nth-last-child(2){color:red;}]`

Findet ein bestimmtes Kindelement, von hinten gesucht.

[C2, F3.5, S3, IE9, O9.5, IOS3, A2]

Verwendet im Abschnitt *Tabellen mit Pseudoklassen stylen* auf Seite 75.

`:first-of-type [p:first-of-type{color:blue;}]`

Findet das erste Element des angegebenen Typs.

[C2, F3.5, S3, IE9, O9.5, IOS3, A2]

Verwendet im Abschnitt *Tabellen mit Pseudoklassen stylen* auf Seite 75.

`:last-of-type [p:last-of-type{color:blue;}]`

Findet das letzte Element des angegebenen Typs.

[C2, F3.5, S3, IE9, O9.5, IOS3, A2]

Verwendet im Abschnitt *Tabellen mit Pseudoklassen stylen* auf Seite 75.

Unterstützung von Spalten `[#content{ column-count:2; column-gap:20px; column-rule:1pxsolid #ddccb5;}]`

Unterteilt einen Inhaltsbereich in mehrere Spalten.

[C2, F3.5, S3, O9.5, IOS3, A2]

Verwendet im Abschnitt *Mehrspaltige Layouts* auf Seite 89.

`:after [span.weight:after{ content: "lbs"; color:#bbb;}]`

Wird zusammen mit `content` verwendet, um Inhalte nach dem angegebenen Element einzufügen. [C2, F3.5, S3, IE8, O9.5, IOS3, A2]
Verwendet im Abschnitt *Links ausdrucken mit :after und content* auf Seite 85.

`Media Queries [media="only all and(max-width:480)"]`

Anwendung von Stilregeln auf der Grundlage von Geräteeinstellungen. [C3, F3.5, S4, IE9, O10.1, IOS3, A2]
Verwendet im Abschnitt *Benutzeroberflächen für mobile Geräte mit Media Queries* auf Seite 95.

`border-radius [border-radius:10px;]`

Rundet die Ecken von Elementen ab. [C4, F3, IE9, S3.2, O10.5]
Verwendet im Abschnitt *Scharfe Ecken abrunden* auf Seite 151.

`RGBA-Unterstützung [background-color:rgba(255,0,0,0.5);]`

Verwendet RGB-Farben statt Hexcodes für die Transparenz. [C4, F3.5, IE9, S3.2, O10.1]
Verwendet im Abschnitt *Schatten, Verläufe und Transformationen* auf Seite 159.

`box-shadow [box-shadow: 10px 10px 5px #333;]`

Versieht Elemente mit einem Schlagschatten. [C3, F3.5, IE9, S3.2, O10.5]
Verwendet im Abschnitt *Schatten, Verläufe und Transformationen* auf Seite 159.

`Rotation: [transform: rotate(7.5deg);]`

Rotiert beliebige Elemente. [C3, F3.5, IE9, S3.2, O10.5]
Verwendet im Abschnitt *Schatten, Verläufe und Transformationen* auf Seite 159.

`Verläufe: [linear-gradient(top,#fff,#efefef);]`

Erstellt Verläufe, die als Bilder verwendet werden können. [C4, F3.5, S4]
Verwendet im Abschnitt *Schatten, Verläufe und Transformationen* auf Seite 159.

`@font-face [@font-face{font-family:AwesomeFont;
src:url(http://example.com/awesomeco.ttf);font-weight:bold;}]`

Ermöglicht die Verwendung spezieller Schriften über CSS. [C4, F3.5, IE5+, S3.2, O10.1]
Verwendet im Abschnitt *Echte Schriften* auf Seite 169.

A.8 Clientseitige Speicherung

localStorage

Speichert Daten in Schlüssel-/Wertpaaren, die an eine Domain gebunden sind und über Browsersitzungen hinweg erhalten bleiben. [C5, F3.5, S4, IE8, O10.5, IOS, A]

Verwendet im Abschnitt *Einstellungen mit localStorage speichern* auf Seite 180.

sessionStorage

Speichert Daten in Schlüssel-/Wertpaaren, die an eine Domain gebunden sind und am Ende einer Browsersitzung gespeichert werden. [C5, F3.5, S4, IE8, O10.5, IOS, A]

Verwendet im Abschnitt *Einstellungen mit localStorage speichern* auf Seite 180.

Web SQL Databases

Vollwertig relationale Datenbanken für das transaktionsgestützte Erstellen von Tabellen und das Einfügen, Aktualisieren, Löschen und Auswählen von Daten; an eine Domain gebunden und sitzungspersistent. [C5, S3.2, O10.5, IOS3.2, A2]

Verwendet im Abschnitt *Daten in einer clientseitigen relationalen Datenbank speichern* auf Seite 186.

A.9 Zusätzliche APIs

Offline-Webanwendungen

Definition von Dateien, die für die Offline-Verwendung zwischengespeichert werden, wodurch Anwendungen auch ohne aktive Internetverbindung ausgeführt werden können.

[C4, S4, F3.5, O10.6, IOS3.2, A2]

Verwendet im Abschnitt *Offline arbeiten* auf Seite 198.

History

Verwaltet den Browserverlauf. [C5, S4, IE8, F3, O10.1, IOS3.2, A2]

Verwendet im Abschnitt *Den Verlauf erhalten* auf Seite 203.

Cross-Document Messaging

Versenden von Benachrichtigungen zwischen Fenstern mit Inhalten von verschiedenen Domains. [C5, S5, F4, IOS4.1, A2]

Verwendet im Abschnitt *Über Domains hinweg kommunizieren* auf Seite 206.

Web Sockets

Herstellen von zustandsbezogenen Verbindungen zwischen einem Browser und einem Server. [C5, S5, F4, IOS4.2]

Verwendet im Abschnitt *Chatten mit Web Sockets* auf Seite 213.

Geolocation

Ruft den Breiten- und Längengrad des Client-Browsers ab. [C5, S5, F3.5, O10.6, IOS3.2, A2]

Verwendet im Abschnitt *Finden Sie sich selbst: Geolocation* auf Seite 220.

Web Workers

Hintergrundverarbeitung von JavaScript. [C3, S4, F3.5, O10.6]

Verwendet im Abschnitt *Web Workers* auf Seite 229.

3D-Canvas mit WebGL

3D-Objekte mit dem canvas-Element. [C5, F4]

Verwendet im Abschnitt *WebGL* auf Seite 236.

Drag-and-Drop

API für Drag-and-Drop-Interaktion. [C3, S4, F3.5, IE6, A2]

Verwendet im Abschnitt *Native Unterstützung für Drag-and-Drop* auf Seite 230.

Anhang B

jQuery-Einführung

Es ist eine schwierige Aufgabe, JavaScript so sauber und prägnant zu schreiben, dass es in allen wichtigen Browsern gut funktioniert. Es gibt viele Bibliotheken, die das wesentlich erleichtern, und jQuery ist eine der beliebtesten davon. Einfach in der Verwendung und mit einer breiten Palette vorhandener Bibliotheken ist jQuery eine gute Wahl für einfache Ausweidlösungen.

In diesem Anhang werden die Teile der jQuery-Bibliothek vorgestellt, die wir in diesem Buch verwenden. Dies soll weder ein Ersatz für die ausgezeichnete Dokumentation von jQuery¹, noch eine abschließende Liste der verfügbaren Funktionen und Methoden sein, Ihnen aber einen guten Einstieg bieten.

B.1 jQuery laden

Sie können sich die jQuery-Bibliothek von der jQuery-Website² holen und das jQuery-Skript direkt einbinden, aber wir laden jQuery von den Google-Servern:

```
jquery/simple_selection.html
```

```
<script type="text/javascript"
  charset="utf-8"
  src="http://ajax.googleapis.com/ajax/libs/jquery/1.4.2/jquery.min.js">
</script>
```

1 <http://docs.jquery.com>

2 <http://www.jquery.com>

Browser können immer nur ein paar Verbindungen gleichzeitig zu einem Server machen. Wenn wir unsere Bilder und Skripten auf mehrere Server verteilen, können die Benutzer unsere Seiten schneller laden. Es hat noch einen weiteren Vorteil, das Google-Netzwerk für die Bereitstellung von Inhalten zu nutzen: Da andere Websites auch die jQuery-Bibliothek von Google einbinden, haben unsere Besucher die Bibliothek unter Umständen bereits im Cache ihres Browsers. Wie Sie wahrscheinlich bereits wissen, verwenden Browser die vollständige URL, um zu erkennen, ob sie eine Kopie zwischengespeichert haben. Wenn Sie vorhaben, mit jQuery auf einem Laptop oder Computer ohne ständige Internetverbindung zu arbeiten, sollten Sie eine lokale Kopie einbinden.

B.2 jQuery-Grundlagen

Nachdem Sie die jQuery-Bibliothek in Ihre Seite geladen haben, können Sie mit der Arbeit an Elementen beginnen. jQuery stellt die Funktion `jQuery()` bereit, die das Herzstück der Bibliothek bildet. Wir verwenden diese Funktion, um Elemente mithilfe von CSS-Selektoren zu erfassen und in jQuery-Objekten zusammenzufassen, damit wir sie manipulieren können. Es gibt auch eine Kurzversion der `jQuery()`-Funktion: `$()`. Das ist die Version, die wir in diesem Buch verwenden. Für den Rest des Anhangs nenne ich diese Funktion „die jQuery-Funktion“. Und so funktioniert sie:

Wenn Sie ein `h1`-Tag auf einer Seite suchen möchten, machen Sie das so:

```
jquery/simple_selection.html
```

```
$("#h1");
```

Wenn Sie alle Elemente mit der Klasse `important` suchen, geht das so:

```
jquery/simple_selection.html
```

```
$(".important");
```

Sehen Sie sich das noch mal genau an. Der einzige Unterschied zwischen den beiden Beispielen ist der CSS-Selektor, den wir verwenden. Die jQuery-Funktion liefert ein jQuery-Objekt zurück – ein spezielles JavaScript-Objekt mit einem Array der DOM-Elemente, die dem Selektor entsprechen. Dieses Objekt verfügt über viele nützliche vordefinierte Methoden, mit denen wir die ausgewählten Elemente manipulieren können. Sehen wir uns einige davon genauer an.

B.3 Methoden zum Verändern von Inhalten

Bei der Arbeit in diesem Buch verwenden wir mehrere jQuery-Methoden, um unsere HTML-Inhalte zu verändern.

Ein- und Ausblenden

Mit den Methoden `hide()` und `show()` ist es einfach, Elemente der Benutzeroberfläche ein- bzw. auszublenden. So können wir ein oder mehrere Seitenelemente ausblenden:

```
jquery/simple_selection.html
```

```
$("#h1").hide();
```

Um die Elemente anzuzeigen, rufen wir einfach die Methode `show()` auf. In diesem Buch verwenden wir die `hide()`-Methode beispielsweise, um Seitenabschnitte auszublenden, die nur angezeigt werden sollen, wenn JavaScript deaktiviert ist, etwa Transkriptionen und andere alternative Inhalte.

html, val und attr

Mit der `html()`-Methode können wir den Inhalt des angegebenen Elements abfragen oder festlegen:

```
jquery/methods.html
```

```
$("#h1").html("Hello World!");
```

Damit legen wir den Inhalt zwischen dem öffnenden und dem schließenden `h1`-Tag auf „Hello World“ fest.

Mit der Methode `val()` können wir den Wert eines Formularfelds festlegen und auslesen. Sie funktioniert genauso wie die `html()`-Methode.

Attribute von Elementen können wir mit der `attr()`-Methode auslesen und festlegen.

append, prepend und wrap

Die `append()`-Methode fügt nach den vorhandenen Elementen ein neues Kindelement ein. Gehen wir davon aus, dass wir ein einfaches Formular mit einer leeren ungeordneten Liste haben:

jquery/methods.html

```

<form id="add">
  <label for="task">Task</label>
  <input type="text" id="task">
  <input type="submit" value="Add">
</form>
<ul id="links">
</ul>

```

Wir können dann neue Elemente in der Liste erstellen, indem wir die neuen Elemente beim Übermitteln des Formulars anfügen:

jquery/methods.html

```

$(function(){
  $("#add").submit(function(event){
    event.preventDefault();
    var new_element = $("<li>" + $("#email").val() + "</li>");
    $("#links").append(new_element);
  });
});

```

Die `prepend()`-Methode funktioniert genauso wie die `append()`-Methode, fügt das neue Element aber vor den vorhandenen ein. Die `wrap()`-Methode verpackt das ausgewählte Element innerhalb des durch das jQuery-Objekt angegebenen Elements:

jquery/methods.html

```
var wrapper = $("#message").wrap("<div><h2>Message</h2></div>").parent();
```

Mit diesen Techniken werden wir einige komplexe Strukturen programmgesteuert erstellen.

CSS und Klassen

Mit der `css()`-Methode können wir Stilregeln für Elemente definieren:

jquery/methods.html

```
$("#label").css("color", "#f00");
```

Wir können die Regeln einzeln definieren oder aber mit einem JavaScript-Hash mehrere CSS-Regeln auf einmal auf das Element anwenden:

jquery/methods.html

```

$("h1").css({ "color" : "red",
              "text-decoration" : "underline" }
);

```

Allerdings ist es keine gute Idee, Stile und Skripten zu vermischen. Wir können die jQuery-Methoden `addClass()` und `removeClass()` einsetzen, um Klassen hinzuzufügen und entfernen, wenn bestimmte Ereignisse auftreten. Anschließend können wir diesen Klassen Stilregeln zuweisen. So können wir die Hintergrundfarbe unserer Formularfelder ändern, wenn sie den Fokus erhalten oder verlieren, indem wir jQuery-Events mit Klassen kombinieren:

jquery/methods.html

```

$("input").focus(function(event){
    $(this).addClass("focused");
});

$("input").blur(function(event){
    $(this).removeClass("focused");
});

```

Dies ist ein triviales Beispiel, das in CSS3 auch durch die Pseudoklasse `:focus` ersetzt werden kann, die aber in manchen Browsern nicht unterstützt wird.

Verkettungen

Methoden von jQuery-Objekten geben jQuery-Objekte zurück. Das bedeutet, dass wir Methoden unendlich lange miteinander verketteten können:

jquery/simple_selection.html

```

$("h2").addClass("hidden").removeClass("visible");

```

Sie sollten es damit aber nicht übertreiben, weil Ihr Code dadurch schwieriger zu verstehen ist.

B.4 Elemente erstellen

Hin und wieder müssen wir neue HTML-Elemente erstellen, damit wir sie in unser Dokument einsetzen können. Auch dafür können wir die `jQuery()`-Methode verwenden:

`jquery/create_elements.html`

```
var input = $("input");
```

Das geht zwar auch mit `document.createElement("input");`, aber mit der `jQuery`-Funktion können wir gleich weitere Funktionen aufrufen:

`jquery/create_elements.html`

```
var element = $("

Hello World</p>");
element.css("color", "#f00").insertAfter("#header");


```

Das ist ein weiteres Beispiel dafür, wie uns Verkettungen in `jQuery` beim schnellen Aufbau und der Manipulation von Strukturen helfen können.

B.5 Events

Häufig müssen wir Events auslösen, wenn Benutzer mit unserer Seite interagieren. Mit `jQuery` ist das ganz einfach. Viele gebräuchliche Events sind in `jQuery` einfach Methoden des `jQuery`-Objekts, die eine Funktion als Parameter entgegennehmen. So können wir beispielsweise alle Links mit der Klasse `popup` auf einer Seite in einem neuen Fenster öffnen lassen:

`jquery/popup.html`

```
1 var links = $("#links a");
2 links.click(function(event){
3     var address = $(this).attr('href');
4     event.preventDefault();
5     window.open(address);
6 });
```

In unserem `jQuery`-Event-Handler können wir auf das jeweilige Element über das Schlüsselwort `this` zugreifen. In Zeile 3 übergeben wir `this` an die `jQuery`-Funktion, damit wir die `attr()`-Methode schnell aufrufen und die Zieladresse des Links auslesen können.

Mit der Funktion `preventDefault()` verhindern wir, dass das ursprüngliche Event ausgelöst wird, damit es uns nicht ins Handwerk pfuscht.

Bind

Einige Events werden von jQuery nicht direkt unterstützt. Diese können wir aber mit der `bind()`-Methode verarbeiten. Wenn wir beispielsweise den Drag-and-Drop-Teil der HTML5-Spezifikation implementieren, müssen wir das `ondragover`-Event abbrechen. Dafür verwenden wir `bind()`:

jquery/bind.html

```
target = $("#droparea")
target.bind('dragover', function(event) {
  if (event.preventDefault()) event.preventDefault();
  return false;
});
```

Beachten Sie, dass wir das Präfix `on` des Events weglassen, das wir überwachen möchten.

Ursprüngliches Event

Wenn wir jQuery-Event-Funktionen wie `bind()` oder `click()` verwenden, verpackt jQuery das JavaScript-Event in ein eigenes Objekt und kopiert nur *manche* seiner Eigenschaften. Manchmal müssen wir deshalb auf das eigentliche Event zugreifen, um auf die Eigenschaften zuzugreifen, die nicht geklont wurden. jQuery bietet uns über die treffend `originalEvent` genannte Eigenschaft Zugriff auf das Original-Event. So können wir zum Beispiel auf die `data`-Eigenschaft des `onmessage`-Events zugreifen:

```
$(window).bind("message",function(event){
  var message_data = event.originalEvent.data;
});
```

Mithilfe dieser Technik können Sie auf alle Eigenschaften oder Methoden des ursprünglichen Events zugreifen.

B.6 Die Funktion `document.ready`

Der Begriff „unobtrusive JavaScript“ („unaufdringliches JavaScript“) bezieht sich auf JavaScript, das vollständig vom Inhalt getrennt ist. Anstatt `onclick`-Attribute zu unseren HTML-Elementen hinzuzufügen, verwenden wir Event-Handler wie diejenigen, über die wir im Abschnitt *Events* auf Seite 254 gesprochen haben. So können wir unser Dokument um „unaufdringliches“ Verhalten erweitern, ohne das Dokument selbst zu verändern. Unser HTML ist dann nicht darauf angewiesen, dass unsere Benutzer JavaScript aktiviert haben.

Ein Nachteil dieser Methode ist, dass JavaScript die Elemente unserer Seite nicht „sehen“ kann, bis sie deklariert wurden. Wir könnten unseren JavaScript-Code in einen script-Block am unteren Rand der Seite stellen, nachdem alles gerendert wurde. Aber dieser Ansatz ist nicht über mehrere Seiten hinweg einsetzbar.

Wir können unseren Code in den JavaScript-Event-Handler `window.onload()` schreiben, aber dieses Event wird erst ausgelöst, nachdem alle Inhalte geladen wurden. Dadurch könnte eine Verzögerung entstehen, in der unsere Benutzer interagieren, bevor die Events verknüpft wurden. Wir brauchen eine Möglichkeit, unsere Events einzubinden, nachdem das DOM geladen wurde, aber bevor es angezeigt wird.

Die jQuery-Funktion `document.ready` ermöglicht genau das, und zwar auf allen Browsern:

`jquery/ready.html`

```
$(document).ready(function() {
    alert("Hi! I am a popup that displays when the page loads");
});
```

Es gibt auch eine kürzere, kompaktere Version, die wir in unserem gesamten Code verwenden:

`jquery/ready.html`

```
$(function() {
    alert("Hi! I am a popup that displays when the page loads");
});
```

Wir verwenden dieses Muster in fast jedem Beispiel in diesem Buch, damit wir unsere Projekte mit einfachen, unaufdringlichen Ausweichlösungen ausstatten können.

Das ist nur ein kleiner Ausschnitt dessen, was mit jQuery möglich ist. Neben den Methoden zum Manipulieren von Dokumenten bietet jQuery auch Methoden zum Serialisieren von Formularen, für Ajax-Anfragen und Dienstfunktionen, die Schleifen und das Durchlaufen des DOM wesentlich einfacher machen. Sobald Sie damit vertrauter sind, finden Sie sicherlich viele weitere Einsatzmöglichkeiten in Ihren Projekten.

Anhang C

Audio und Video kodieren

Die Kodierung von Audio und Video für die Verwendung mit den HTML5-Tags `audio` und `video` ist ein komplexes Thema, das den Rahmen dieses Buchs sprengen würde. Aber dieser kurze Anhang soll Ihnen die Richtung weisen, falls Sie einmal eigene Inhalte vorbereiten müssen.

C.1 Audio kodieren

Sie müssen Ihre Audiodateien in den beiden Formaten MP3 und Vorbis bereitstellen, um ein möglichst breites Publikum zu erreichen. Dafür müssen Sie eine Reihe von Werkzeugen verwenden.

Für MP3-Dateien bietet Lame die beste Qualität. Sie sollten eine variable Bitrate verwenden. So erreichen Sie eine qualitativ hochwertige Kodierung:

```
lame in.wav out.mp3 -V2 --vbr-new -q0 --lowpass 19.7
```

Für Vorbis verwenden Sie Oggenc zur Audiokodierung. So erhalten Sie eine gut klingende Vorbis-Datei mit variabler Bitrate:

```
oggenc -q 3 inputfile.wav
```

Mehr über die Kodierung von MP3 und Vorbis erfahren Sie bei Hydrogen Audio.¹ Die Informationen sind ausgezeichnet, aber Sie müssen mit den Einstellungen experimentieren, damit Sie für Sie und Ihre Zuhörer die richtigen Werte finden.

C.2 Video für das Web kodieren

Wenn Sie mit HTML5-Video jede Plattform erreichen möchten, müssen Sie Ihre Videodateien in mehreren Formaten kodieren. Die Kodierung mit H.264, Theora und VP8 ist zeitintensiv – sowohl bei der Einrichtung eines Open Source-Encoders wie etwa FFMpeg² als auch bei der eigentlichen Kodierung. Die richtige Kodierung von Videos sprengt den Rahmen dieses Buchs. Wir haben gar nicht genug Zeilen, um den folgenden Befehl zu erklären, der eine Datei in VP8 mit dem WebM-Container konvertiert:

```
ffmpeg -i blur.mov
       -f webm -vcodec libvpx_vp8 -acodec libvorbis
       -ab 160000 -sameq
       blur.webm
```

Wenn Sie sich nicht selbst mit den Einstellungen herumärgern möchten, können Sie Ihre Videos mit dem Webservice Zencoder³ in alle für HTML5 erforderlichen Formate kodieren. Sie hinterlegen einfach Ihre Videos auf Amazon S3 oder einer anderen öffentlichen URL und können dann über die Weboberfläche oder API-Aufrufe Aufträge einrichten, um diese Videodateien in mehrere Formate zu kodieren. Zencoder holt die Videodateien ab, kodiert sie und überträgt die neuen Videos zurück an Ihren Server. Der Service ist nicht kostenlos, liefert aber hervorragende Ergebnisse und spart Ihnen eine Menge Zeit, wenn Sie viele Inhalte kodieren müssen.⁴

Wenn Sie einfach nur mit den Formaten experimentieren möchten, ist Miro Video Converter⁵ eine nette Option. Die Software ist Open Source und bietet viele Voreinstellungen für die Konvertierung Ihrer Videodateien in mehrere Formate.

1 Lame finden Sie unter http://wiki.hydrogenaudio.org/index.php?title=Lame#Quick_start_.28short_answer.29, Vorbis unter http://wiki.hydrogenaudio.org/index.php?title=Recommended_Ogg_Vorbis

2 <http://www.ffmpeg.org/>

3 <http://www.zencoder.com/>

4 Um ganz ehrlich zu sein: Ich kenne einige Entwickler bei Zencoder. Aber ich würde den Service auch empfehlen, wenn ich dort niemand kennen würde.

5 <http://mirovideoconverter.com/>

Anhang D

Ressourcen

D.1 Ressourcen im Web

Apple - HTML5

<http://www.apple.com/html5/>

Apples Seite über HTML5, Webstandards und darüber, wie sie von Safari 5 unterstützt werden.

CSS3.Info

<http://www.css3.info/>

Jede Menge Hintergrundwissen und Beispiele zu den verschiedenen Modulen von CSS3.

Font Squirrel

<http://www.fontsquirrel.com>

Lizenzfreie Schriften in verschiedenen Formaten für die Verbreitung im Web.

HTML5

<http://www.w3.org/TR/html5/>

Die HTML5-Spezifikation des W3C.

HTML5 - Mozilla Developer Center

<https://developer.mozilla.org/en/html/html5>

Die Seite des Mozilla Developer Center über HTML5

Web Socket-Server mit Node.js implementieren

<http://www.web2media.net/laktek/2010/05/04/implementing-web-socket-servers-with-node-js/>

Hier wird erklärt, wie man mit Node.js einen Web Sockets-Server schreibt.

Probefahrt mit Microsoft IE9

<http://windows.microsoft.com/de-DE/internet-explorer/products/ie-9/home>

Demonstration der HTML5-Funktionen (und ähnlicher) in Internet Explorer 9.

Ruby und Web Sockets - TCP für den Browser

<http://www.igvita.com/2009/12/22/ruby-websockets-tcp-for-the-browser/>

Informationen über em-websocket, eine Ruby-Bibliothek zum Erstellen von Web Sockets-Servern

Flash-Richtliniendatei einrichten

http://www.lightsphere.com/dev/articles/flash_socket_policy.html

Ausführliche Beschreibung von Flash Socket Policy-Dateien.

Typekit

<http://www.typekit.com>

Service, mit dem Sie lizenzierte Schriften über eine einfache JavaScript-API auf Ihrer Website verwenden können.

Unit Interactive: „Better CSS Font Stacks“...

<http://unitinteractive.com/blog/2008/06/26/better-css-font-stacks/>

Diskussion über Font Stacks, mit einigen tollen Beispielen.

Video für Alle!

http://camendesign.com/code/video_for_everybody

Informationen über HTML5-Video, mit Code zum Abspielen von Videos auf allen Browsern.

Video.js

<http://videojs.com>

JavaScript-Bibliothek, die beim Abspielen von HTML5-Videos hilft.

Wo kann ich ... verwenden?

<http://caniuse.com/>

Browserkompatibilitätstabellen für HTML5, CSS3 und verwandte Technologien.

Anhang E

Literaturverzeichnis

[Hog09]

Brian P. Hogan. *Web Design For Developers*.
The Pragmatic Programmers, 1. Auflage, Oktober 2009.

[HT03]

Andrew Hunt und David Thomas.
Der Pragmatische Programmierer, Hanser Fachbuch,
1. Auflage, März 2003.

[Zel08]

Jeffrey Zeldman. *Webdesign mit Webstandards*.
Addison-Wesley, München, 1. Auflage, Mai 2008.

Index

A

- AAC (Advanced Audio Coding) 136
- Abrundung aufrufen 157
- Abwärtskompatibilität 13
- Accessibility for Rich Internet Applications (WIA-ARIA) 99, 101
- addClass()-Methode 253
- addToNotesList()-Methode 194
- Advanced Audio Coding (AAC) 136
- Ajax API (Google) 223
- :after 85
- Anwendungen 10
- Apache, Caching 200
- APIs
 - Browserverlauf 203
 - Cross-Document Messaging 206, 212
 - Geolocation 220, 224
 - Kurzreferenz der Funktionen 248
 - Web Sockets 219
- append()-Methode 251–252
- Apple 10, 136
 - siehe auch* Safari (Apple)
- ARIA-Rollen 101
- article-Tag 25, 31
- aside-Tag 25, 32
- assertive-Methode 108
- attr()-Methode 251
- Attribute
 - autocomplete 59
 - autofocus 56
 - benutzerdefinierte Datenattribute 39–42
 - Formularfeld 243
 - für die Darstellung 17
 - ID 28
 - Kurzreferenz 241
 - longdesc 18
 - placeholder 58
 - profile 18
 - siehe auch* contenteditable-Attribut
- Audio
 - Auswechlösung 140
 - Barrierefreiheit 148
 - Codecs 133
 - siehe auch* Video
- Audio und Video kodieren 257
- Aufforderungen zur Interaktion 226
- Auswechlösung
 - Audio 140
 - ausdruckbare Links 87
 - Barrierefreiheit 128
 - benutzerdefinierte Datenattribute 42
 - Browserverlauf 205
 - canvas-Element 121
 - clientseitige relationale Datenbank 196
 - contenteditable-Attribut 67

- Cross-Domain Messaging 211
- CSS3-Spalten 92
- Ecken abrunden 154
- Geolocation 222
- Internet Explorer 165
- localStorage 183
- Media Queries 97
- placeholder-Attribut 60
- Schriften 174
- Selektoren 82
- semantisches Markup 37
- Video 143
- Web Sockets 217
- autocomplete-Attribut 59
- autofocus-Attribut 56

B

- Balkendiagramm, aus HTML erzeugen 124
- Barrierefreiheit
 - Ausweichlösung 128
 - Kurzreferenz 244
 - Überblick 12, 15, 99
- bedingter Kommentar 37
- beginPath()-Methode 118
- benutzerdefinierte Datenattribute 10, 25
- Benutzeroberflächen
 - mit CSS3 73–98
 - Notizen 187
 - scharfe Ecken abrunden 151–158
 - Schatten, Verläufe und Transformationen 159–168
 - Schriften 169–174
 - strukturelle Tags und Attribute 23–42
 - Überblick 11, 149
 - Webformulare 45–70
- Bildschirmlesegeräte 99
- bind()-Methode 255
- Blog, neu definiert mit semantischem Markup 36
- box-shadow-Eigenschaft 161
- Browser
 - spezifische Browser 97
 - Unterstützung von Selektoren 97
 - Verlauf mit APIs verwalten 203
- Browser-spezifische Selektoren 153

C

- Cache
 - Apache 200
 - definieren mit Manifestdatei 198
- canPlayType()-Methode 141
- canvas-Element 113, 115
 - Statistiken grafisch darstellen mit RGraph 122–130
- canvas-Tag 113–115
- Chrome (Google)
 - Browserververlauf 205
 - Farbverläufe in 161
 - localStorage 183
 - Offline-Unterstützung 198
 - Slider-Widget 50
- Clark, Keith 83
- click()-Methode 255
- ClientLocation()-Methode 223
- clientseitige Daten
 - clientseitige relationale Datenbank 186–197
 - localStorage 180, 185
 - Überblick 177
- clientseitige Formularvalidierung 238
- clientseitige relationale Datenbank
 - CRUD 186
 - Datensätze ändern 193
 - Datensätze suchen 192
 - Notiz-Benutzeroberfläche 187
 - Notizen laden 191
 - Notiz-Tabellen erstellen 190
 - offline arbeiten 198
 - Schaltfläche „New“ aktivieren 195
 - Verbindung herstellen 190
- clientseitige Speicherung 11
- Codecs *siehe* Container und Codecs
- color
 - input type 52
- Container und Codecs
 - Advanced Audio Coding (AAC) 136
 - als Team 137
 - Audio-Codecs 135

- H.264 134
- MP3 136
- Theora 135
- Video-Codecs 133
- VP8 135
- contenteditable-Attribut 65, 67
 - Ausweidlösung 68
 - Bearbeitungsseiten erstellen 68
 - Daten ablegen 67
 - Profilformular 65
 - Überblick 65
- Cookies
 - JavaScript und 184
 - Überblick 177
- Create, Retrieve, Update, and Delete (CRUD) 186
- Cross-Document Messaging
 - Überblick 201
- Cross-Domain Messaging
 - Ausweidlösung 211
 - Nachrichten empfangen 211
 - Nachrichten senden 208
 - Support-Website 209
 - Webserver 207
- CSS
 - Inhalte erzeugen mit 85
- css()-Methode 252
- CSS3
 - Funktionen 73
 - Herausforderungen 15
 - Kurzreferenz 246
 - Übergänge 226
 - Vorteile von 9
 - Zukunft 19

D

- Darstellung (Benutzeroberflächen)
 - scharfe Ecken abrunden 151–158
 - Schatten, Verläufe und Transformationen 159–168
 - Schriften 169–174
 - Überblick 149
- Dateien
 - Manifest 198
 - MP3 131, 257

- Daten
 - Attribute 12
 - localStorage 180
 - mit HTML beschreiben 123
 - speichern 67
 - siehe auch* clientseitige Daten
- Datenbanken 190
 - siehe auch* clientseitige relationale Datenbank
- Datensätze
 - ändern 193
 - suchen 192
- Datumswerte 51
- DirectX-Filter (Microsoft) 165
- Divitis 23
- Doctype
 - Deklaration 13
 - HTML5 28
- document.ready-Funktion 256
- Don't Repeat Yourself (DRY) 68
- Drag-and-Drop-Events 232
- Drag-and-Drop-Implementierung (Microsoft) 230
- draggable-HTML5-Attribut 233
- DRY (Don't Repeat Yourself) 68

E

- ease-in-Kurve 228
- ease-in-out-Kurve 228
- Ecken abrunden
 - Abrundung aufrufen 157
 - Ausweidlösung 154
 - formCorners-Plugin 156
 - Internet Explorer 157
 - Überblick 151
 - Unterstützung ermitteln 154
- Effekte (visuelle) 12
- Einbetten
 - Audio 138–141
 - Container und Codecs 134
 - Geschichte 133
 - Überblick 131
 - Video 142, 148
- Eingabefelder
 - Ausweidlösung 53
 - Datumswerte 51

- E-Mail 51
- Farbe 52
- Farbwähler ersetzen 53
- grundlegendes Formular 48
- Modernizr 55
- Slider erstellen 49
- Spinbox 50
- Überblick 48
- URL 52
- Einstellungen
 - anwenden 182
 - speichern mit localStorage 185
 - speichern und laden 181
- Elemente
 - ablegen 233
 - block 24
 - drehen 163, 165
 - Kurzreferenz 241
 - Logos zeichnen 121
 - Statistiken mit RGraph darstellen 122, 129
 - siehe auch* Tags
- E-Mail-Formularfeld 51
- Embedded OpenType (EOT) 170
- embed-Tag 132
- executeSql()-Methode 191
- ExplorerCanvas-Bibliothek 121, 129

F

- Farbe
 - für canvas festlegen 120
- Farbverlauf 161
- Farbwähler 53
- FFMpeg 258
- Filter (DirectX) 165
- Firefox (Mozilla)
 - Browserverlauf 205
 - Farbverläufe in 161
 - localStorage 183
 - Media Queries 97
 - moz-linear-gradient 161
 - Offline-Unterstützung 198
 - Unterstützung für abgerundete Ecken 151

- Flash (Adobe)
 - browserübergreifende Kompatibilität 133
 - im Vergleich zum canvas-Element 129
 - Verfügbarkeit 133
 - Web Sockets mit 217
- Flash Socket Policy 218
- @font-face-Anweisung 170
- Font Stacks 173, 261
- Fonts *siehe* Schriften
- FontSquirrel-Schriften 172
- footer-Tag 25, 29
- Formate
 - Video 133
- formCorners-Plugin 155
- Formulare
 - siehe* Webformulare
- Funktionen
 - Timing 227

G

- Geolocation 220, 222
- Geschichte des Embedding 132
- getContext-Methode 115
- getScript()-Methode 211, 223
- Google
 - Ajax-API 223
 - Chrome Frame-Plugin 197
 - Einführung von HTML5 und CSS3 durch 18
 - Static Map API 220
 - VP8 135
 - siehe auch* Chrome (Google)
- gradient-Filter (Internet Explorer) 166

H

- handheld-Medientypen 96
- header-Tag 25, 28
- Herausforderungen durch HTML5 und CSS3 15
- hide()-Methode 251
- Hintergründe, Transparenz 163

HTML
 als Balkendiagramm darstellen 124
 Code ändern 83
 Daten beschreiben mit 123
 html()-Methode 251
 HTML5
 Doctype 26
 Herausforderungen 19
 Offline-Unterstützung 198
 Vorteile 9
 Zukunft 15
 HTMLShiv 38
 Hydrogen Audio 257

I

id-Attribut 28
 Indexed Database API 237
 Inhalt
 mit CSS erzeugen 85
 vom Verhalten trennen 39
 insertNote()-Methode 194
 Internet Explorer
 Ausweichlösung 165
 Browserverlauf 205
 Ecken abrunden 151, 157
 Embedded OpenType 171
 gradient-Filter 166
 localStorage 183
 semantisches Markup 37
 Stylesheets 164, 167
 Überblick 15

J

JavaScript
 API für Medieninhalte 146
 benutzerdefinierte Datenattribute 43
 Cookies und 184
 Elemente definieren mit 37
 Tabellen stylen mit 83
 unaufdringliches 252
 jQuery
 Address-Plugin 205
 bind()-Methode 233
 Columnizer-Plugin 92

CSS contra 126
 Einführung 249–256
 Farbwähler ersetzen mit 53
 Grundlagen 83
 Postback-Plugin 211
 Selektor 41
 Website 249
 jQuery()-Methode 253
 jQuery-Bibliothek
 document.ready 255
 Elemente erstellen 254
 Events 254
 Grundlagen 83, 250
 laden 249
 Methoden zum Verändern von Inhalten 251

K

Kompatibilität (Abwärts-) 13
 Kopfzeilen 28

L

:lastchild-child-Selektor 80
 Landmark-Rollen 101
 Linien, auf dem canvas-Element zeichnen 118
 loadNote()-Methode 192
 localStorage
 Ausweichlösung 183
 Einstellungen anwenden 182
 Formulare für Benutzereinstellungen 180
 Überblick 178, 180
 Logos zeichnen 115
 longdesc-Attribut 18

M

Macromedia 132
 Manifestdatei 198
 Markup
 siehe semantisches Markup
 Media Queries
 in Chrome 97

- mobile Benutzeroberflächen
 - erstellen mit 95
- meter-Tag 25, 36
- Methoden 39
- Methoden verketteten 253
- Microsoft
 - DirectX-Filter 165
 - Drag-and-Drop-Implementierung 230
 - Web Open Font Format 171
- Miro Video Converter 258
- mobile Benutzeroberflächen 95
- Mobile Safari-Browser 132
- Modernizr-Bibliothek 55
- Mozilla
 - Einführung von HTML5 und CSS3 durch 19
 - siehe auch* Firefox (Mozilla)
 - Web Open Font Format 171
- moz-linear-gradient-Methode 161
- MP3-Dateien 136, 257
- MP4-Container 137
- Multimedia
 - Kurzreferenz 10, 244
 - Überblick 10

N

- Nachrichten
 - siehe* Cross-Domain Messaging
- nav-Tag 25, 30
- Notizen
 - Benutzeroberfläche 187
 - laden 191
 - Tabellen 190
- :nth-child-Selektor 78
- :nth-last-child-Selektor 81
- :nth-of-type-Selektor 77

O

- Offline-Unterstützung 198
- OGG-Container 137
- onclick-Methode 39
- onclose()-Methode 216
- onmessage()-Methode 211, 216
- onopen()-Methode 216

- Opera
 - Farbwähler 53
 - Kalender-Picker 51
 - Media Queries 97
 - Slider-Widget 50
 - Spinbox 50
 - Web Open Font Format 171

P

- placeholder-Attribut 58, 61
- Plugins
 - formCorners 155
 - Google Chrome Frame 197
 - jQuery Address 205
 - jQuery Columnizer 92
 - jQuery Postback 211
- polite-Methode 108
- postMessage()-Methode 208, 211, 229
- prepend()-Methode 252
- print-Type 86
- profile-Attribut 18
- progress-Tag 25, 36
- Pseudoklassen 75–78
- pushstate()-Methode 204

R

- removeClass()-Methode 253
- rgba-Funktion 164
- Ruby on Rails-Framework 43
- Ruby-basierte Server 207

S

- Safari (Apple)
 - Browsersverlauf 205
 - Farbverläufe in 161
 - localStorage 183
 - Offline-Unterstützung 198
 - Selektoren 153
 - Unterstützung für abgerundete Ecken 151
- save()-Methode 119
- Schatten
 - hinzufügen 161

Schriften

- @font-face-Anweisung 170
- ändern 172
- Ausweidlösung 173
- konvertieren 171
- Rechte und 170
- Überblick 169

section-Tag 25, 31

Seitenleisten 32

selbstschließende Tags 14

Selektoren 78, 81

- browserspezifische 153

- Definition 73

- :last-child 80

- :nth-child 78

- :nth-last-child 81

- :nth-of-type 77–78

- Überblick 12

- Unterstützung durch Browser 97

semantisches Markup

- article-Tag 31

- aside-Tag 25

- Ausweidlösung 37

- Blog neu definiert mit 26, 35

- Doctype 26–27

- footer-Tag 29

- header-Tag 28

- meter-Tag 36

- nav-Tag 30

- progress-Tag 36

- section-Tag 31

- Seitenleisten 32

- stylen 34

- Überblick 29

send()-Methode 217

Server

- siehe* Webserver

serverseitige Speicherung 183

sessionStorage 178, 185

Sharp, Remy 38

show()-Methode 251

showLocation()-Methode 221, 223

Slider 49

Spalten

- aufteilen 89

- Breite festlegen 92

- Text ausrichten 78

Speicherung

- clientseitige 11, 247

- serverseitige 183

spezifische Tags 28

Spinboxen 50

Static Map API (Google) 220

Statistiken grafisch darstellen mit

- RGraph 122

Stilregeln

- ändern 234

- auf Elemente anwenden 34

Stylesheets

- auf Seiten anwenden 159

- Internet Explorer 164, 167

T**Tags 28**

- selbstschließend 14

- veraltet 16

- siehe auch* spezifische Tags

Text

- hinzufügen zum canvas-Element

- 118

- in Spalten ausrichten 78

- Schatten 162

Theora 135

Timing-Funktionen 227

transition-timing-function-

- Eigenschaft 227

transparente Hintergründe 163

Transparenzwerte 167

U

Übergänge (CSS3) 226

Unterstützung

- für abgerundete Ecken ermitteln

- 154

- Offline- 198

updateNote()-Methode 194

URL-Eingabefeld 52

Ursprung verschieben 119

V

- val()-Methode 251
- Validierungsservice des W3C 18
- veraltete Tags 16
- Video
 - Ausweichlösung 143
 - Codecs 134
 - Grenzen von HTML5 147
 - Kodierung 258
 - siehe auch* Audio
- video-Tag 142
- Vorbis-Audio 257
- Vorteile von HTML5 und CSS3 9
- VP8 135

W

- Web Open Font Format 171
- Web Sockets
 - Ausweichlösung 217
 - Chat-Oberfläche 213
 - Server 213, 219
 - Überblick 11, 213
- Web Sockets-JavaScript-Bibliothek 217
- Web SQL Databases 178
- Web Workers 229
- Webentwicklung 9
- Webformulare
 - autofocus-Attribut 56

- clientseitige Validierung 238
- contenteditable-Attribut 65–70
- Daten mit neuen Eingabefeldern beschreiben 48–54
- Felder 46
- grundlegendes Formular erstellen 48
- Kurzreferenz 243
- placeholder-Attribut 58–62
- Überblick 11, 45

- WebGL 236
- WebKit-basierte Browser 161
- WebM-Container 137
- Webserver
 - Cross-Domain Messaging 207
 - sprechen mit 213
 - Web Sockets 219
- WIA-ARIA (Accessibility for Rich Internet Applications) 99, 101
- window.localStorage()-Methode 181
- window.onpopstate()-Event 204
- Windows (Pop-up) 42
- wrap()-Methode 252

Z

- Zebrastreifen 77
- Zencoder 258
- Zukunft von HTML5 und CSS3 19

