

*Schnelle Lösungen
für typische CSS-Probleme*

3. Auflage
Behandelt CSS 2.1 & CSS3



CSS Kochbuch

O'REILLY®

*Christopher Schmitt
& Jørgen W. Lang*

3. Auflage

CSS Kochbuch

Christopher Schmitt & Jørgen W. Lang

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:

© 2012 by O'Reilly Verlag GmbH & Co. KG

1. Auflage 2005

2. Auflage 2007

3. Auflage 2012

Diese Ausgabe des Buches ist eine Aktualisierung, Überarbeitung und Erweiterung der 3. englischen Auflage des Buches, die 2009 unter dem Titel *CSS Cookbook, 3rd Edition* bei O'Reilly Media, Inc. erschienen ist.

Die Darstellung eines Grizzlybären im Zusammenhang mit dem Thema CSS ist ein Warenzeichen von O'Reilly Media, Inc.

Bibliografische Information der Deutschen Bibliothek

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

Lektorat: Inken Kiupel, Köln

Fachliche Unterstützung: Jens Grochtdreis, Mainz, und Jens O. Meiert, Santa Clara, USA

Korrektorat: Sibylle Feldmann, Düsseldorf

Satz: Reemers Publishing Services GmbH, Krefeld; www.reemers.de

Produktion: Geesche Kieckbusch, Hamburg

Belichtung, Druck und buchbinderische Verarbeitung:

Druckerei Kösel, Krugzell; www.koeselbuch.de

ISBN 978-3-89721-324-1

Dieses Buch ist auf 100% chlorfrei gebleichtem Papier gedruckt.

Vorwort	XIII
Einführung	XV
1 Allgemeines	1
1.1 Eine Webseite mit Stildefinitionen versehen.	1
1.2 Den Aufbau einer CSS-Regel verstehen.	6
1.3 Verschiedene Selektoren für die Zuweisung von Stilen verwenden	9
1.4 Selektoren gruppieren	12
1.5 Alle Elemente auf einmal auswählen.	13
1.6 Bestimmte Elemente nach ihrem Namen auswählen	15
1.7 Nachfahren-elemente auswählen	16
1.8 Kindelemente auswählen	18
1.9 Benachbarte Geschwister-elemente auswählen	20
1.10 Regeln für allgemeine Geschwister-elemente definieren	22
1.11 Elemente nach ihren Attributen auswählen	23
1.12 Elemente aufgrund von Klassen- und ID-Attributen auswählen.	26
1.13 Mehrere Klassen auf einmal angeben	30
1.14 Regeln für unbenannte Dokument-teile definieren	30
1.15 Werte für die neue Pseudoklasse :nth-child() und ihre Verwandten richtig angeben	35
1.16 Das Boxmodell begreifen	38
1.17 Darstellungsrollen verstehen.	40
1.18 Die Vererbung nutzen	44
1.19 Die Reihenfolge von Stylesheets: die Kaskade	46
1.20 Darstellungskonflikte anhand der Spezifität lösen	48
1.21 Regeln mit besonderer Priorität versehen	51

1.22	Dokumenttypen und ihre Auswirkungen auf die Browserdarstellung	52
1.23	Stylesheets für verschiedene Ausgabemedien definieren	53
1.24	Den Inhalt eines Stylesheets richtig strukturieren	55
2	Grundrezepte	57
2.1	Längeneinheiten verstehen	57
2.2	Höhen und Breiten definieren	60
2.3	Farben definieren	61
2.4	Externe Ressourcen einbinden	64
2.5	Außenabstände definieren	66
2.6	Innenabstände definieren	69
2.7	Elemente mit einem Rahmen versehen	70
2.8	Hintergrundbilder verwenden	73
2.9	Die Wiederholung von Hintergrundbildern steuern	75
2.10	Hintergrundbilder positionieren	76
2.11	Das Hintergrundbild verankern	78
2.12	Mehrere Hintergrundbilder für ein Element definieren	79
2.13	Hintergrundbilder auf den gesamten Ansichtsbereich ausdehnen	82
2.14	Definitionen für Hintergründe in einer Deklaration zusammenfassen	86
2.15	Durchscheinende Hintergründe definieren	87
2.16	Elemente positionieren	89
2.17	Elemente absolut positionieren	90
2.18	Elemente relativ positionieren	92
2.19	Elemente ineinander positionieren	93
2.20	Elemente im Ansichtsbereich verankern	95
2.21	Elemente voreinander anordnen	96
2.22	Elemente scrollbar machen	98
2.23	Elemente umfließen lassen	100
2.24	Elemente horizontal zentrieren	103
2.25	Elemente vertikal zentrieren	106
3	Typografie für das Web	109
3.1	Schriftarten festlegen	109
3.2	Schriftgrößen und -maße festlegen	113
3.3	Die Schriftfarbe ändern	116
3.4	Das Schriftgewicht ändern	117
3.5	Den Schriftstil anpassen	118
3.6	Text in Kapitälchen darstellen	120
3.7	Text in Groß- und Kleinschreibung darstellen	121

3.8	Zeilenhöhen ändern	122
3.9	Den Abstand zwischen Buchstaben und Wörtern anpassen.	123
3.10	Die Kurzschrift Eigenschaft font verwenden	125
3.11	Webfonts verwenden	128
3.12	Die Textausrichtung festlegen.	133
3.13	Zeilenumbrüche steuern	134
3.14	Text mit Über-, Unter- oder Durchstreichungen versehen	138
3.15	Grafische Unterstreichungen verwenden	140
3.16	Sonderzeichen aus einem anderen Zeichensatz einfügen	142
3.17	Typografisch korrekte Anführungszeichen verwenden.	143
3.18	Hängende Initialen zu Beginn eines Absatzes einfügen	146
3.19	Bilder als Initialen einsetzen	148
3.20	Stile für die erste Absatzzeile definieren	150
3.21	Überschriften mit Grafiken überlagern	151
3.22	Die Gestaltung von ausgewähltem Text anpassen	154
3.23	Text mit einem Schatten versehen	155
3.24	Hoch- und tiefgestellten Text ohne Grundlinienversatz darstellen.	158
3.25	Die Schreibrichtung festlegen	160
3.26	Pull Quotes gestalten.	161
4	Bilder	165
4.1	Bilder mit Rahmen versehen	165
4.2	Rahmen um Bilder nicht darstellen.	167
4.3	Die Rahmenfarbe ändern, wenn sich der Mauszeiger über dem Bild befindet	168
4.4	Skalierbare Bilder	170
4.5	Bilder mit einer Bildunterschrift versehen.	172
4.6	Text durch Grafiken ersetzen	178
4.7	Bilder als Panorama präsentieren	182
4.8	Verschiedene Bildformate miteinander kombinieren	184
4.9	Sprites zum Speichern von Icons verwenden	187
5	Listen	190
5.1	Browserübergreifende Einrückungen	191
5.2	Spezielle Aufzählungszeichen verwenden	192
5.3	Eigene textbasierte Aufzählungszeichen für Listeneinträge verwenden	194
5.4	Aufzählungszeichen mit einer eigenen Farbe versehen.	197
5.5	Eigene Grafiken als Aufzählungszeichen verwenden	199
5.6	Das Aufzählungszeichen innerhalb des Listeneintrags darstellen.	201

5.7	Die Kurzschrifteigenschaft list-style verwenden	202
5.8	Trennlinien für Listeneinträge erstellen	203
5.9	Listeneinträge nebeneinander darstellen	205
5.10	Stile für Definitionslisten	207
5.11	Stile für den ersten oder letzten Eintrag einer Liste definieren	209
6	Links und Navigation	211
6.1	Die Darstellung der verschiedenen Linkzustände anpassen	212
6.2	Unterschiedliche Linkfarben für bestimmte Teile einer Seite	215
6.3	Icons am Ende von Links einfügen	216
6.4	Tooltips mit CSS gestalten	218
6.5	Dokumentteile nur bei Bedarf anzeigen	222
6.6	Den anklickbaren Bereich eines Links vergrößern	224
6.7	Listen als Navigationsmenüs gestalten	226
6.8	Die aktuelle Seite hervorheben	230
6.9	Eine Breadcrumb-Navigation gestalten	232
6.10	Karteireiter-Navigationsmenüs	235
6.11	Ein vertikales Navigationsmenü mit mehreren Ebenen erstellen	239
6.12	Ein horizontales Navigationsmenü mit mehreren Ebenen erstellen	243
7	Formulare	251
7.1	Stile für Formulare definieren	252
7.2	Stile für fieldset-Elemente definieren	254
7.3	legend-Elemente mit Stildefinitionen versehen	256
7.4	Die Platzierung von legend-Elementen verändern	261
7.5	label-Elemente gestalten	263
7.6	Texteingabefelder mit CSS gestalten	265
7.7	Stildefinitionen für textarea-Elemente zuweisen	268
7.8	Stile für Checkboxes und Radiobuttons definieren	270
7.9	Stildefinitionen für Auswahllisten und Aufklappmenüs	272
7.10	Stildefinitionen für Buttons	274
7.11	Grafikbasierte Buttons	277
7.12	Buttons wie einfachen HTML-Text darstellen	279
7.13	Eingabefelder hervorheben	281
7.14	Benötigte Formularfelder hervorheben	283
7.15	Formularelemente zeilenweise darstellen	285
7.16	Serviovorschlag: ein mehrspaltiges Formular	288

8	Tabellen	301
8.1	Rahmen und Innenabstände für Zellen einstellen	301
8.2	Den Zellzwischenraum einstellen	305
8.3	Leerraum zwischen Bildern und dem unteren Zellenrand entfernen	306
8.4	Tabellenzellen individuell gestalten	308
8.5	Leere Tabellenzellen verstecken	312
8.6	Stildefinitionen für die Tabellenüberschriften	313
8.7	Stildefinitionen für Tabellenbeschriftungen	314
8.8	Rollover-Effekte für Tabellenzeilen	315
8.9	Tabellezeilen individuell gestalten	317
8.10	Tabellenspalten gestalten	320
8.11	Abwechselnde Hintergrundfarben für Tabellezeilen	324
8.12	Tabellen mit einer festen Breite versehen	326
8.13	Servervorschlag: eine Tabelle mit Stil	328
9	Drucken	333
9.1	Spezielle Stylesheet-Regeln für das Ausdrucken angeben	334
9.2	Die richtigen Längeneinheiten und Werte für das Ausdrucken verwenden	336
9.3	Eine Seite für Schwarz-Weiß-Ausdrucke vorbereiten	337
9.4	Nach einem Link den URI anzeigen	338
9.5	Sonderzeichen vor Links einfügen	340
9.6	Seitenumbrüche anlegen	341
9.7	Überschriften von Tabellenspalten auf jeder Seite mitdrucken	343
9.8	Ein Webformular für den Ausdruck aufbereiten	344
9.9	Bestimmte Dokumentteile nicht mit ausdrucken	350
9.10	Das Ausdrucken von Hintergrundbildern selbst steuern	352
10	Mobile Geräte	355
10.1	Media Queries verwenden, um mobile und Desktop-Browser zu unterscheiden	355
10.2	Längs- oder Querformat ermitteln	360
10.3	Die Größe und Skalierung des Ansichtsbereichs steuern	361
10.4	Stile für Mobilgeräte zuerst laden	366
10.5	Das Laden von Stylesheets beschleunigen	367
10.6	Mobile Seiten mit CSS3 schneller machen	368
10.7	Nicht benötigte Elemente ausblenden	370
10.8	Seitenlayouts linearisieren	371
10.9	Tabellen linearisieren	373

10.10	Links als Buttons darstellen.	379
10.11	Testen	383
11	Seitenlayouts.	384
11.1	Flexible zweispaltige Layouts erstellen	384
11.2	Zweispaltige Layouts mit festen Breiten erstellen	388
11.3	Flexible mehrspaltige Layouts mit Floats erstellen	391
11.4	Spalten mit Floats in beliebiger Reihenfolge darstellen.	394
11.5	Faux Columns einsetzen	399
11.6	Den »heiligen Gral« verstehen.	403
11.7	Den Fußteil an der Unterkante des Browserfensters verankern	406
11.8	Layouts mit einem »Sticky Footer« versehen	408
11.9	Grid-System benutzen	411
11.10	Inhalte mit CSS3 auf mehrere Spalten verteilen	415
11.11	Ausblick.	418
12	Design mit CSS	421
12.1	Rollover-Effekte ohne Bilder erzeugen	421
12.2	Ein Farbschema für eine Website erstellen	424
12.3	Für ausreichenden Farbkontrast sorgen	427
12.4	Farbverläufe mit CSS3 definieren	429
12.5	Transparenzen erzeugen	434
12.6	Abgerundete Ecken mit CSS3 definieren.	438
12.7	Schattenwürfe mit CSS3 erzeugen.	442
12.8	Realistische Schattenwürfe im Internet Explorer vor Version 9 erzeugen	445
12.9	Elemente umformen	448
12.10	Einfache Animationen erstellen.	454
12.11	Alternative Cursorsymbole definieren.	461
12.12	Modulare Skalen verwenden	463
12.13	Ein Grundlinienraster verwenden	465
12.14	Quasi zufällige Hintergrundmuster erzeugen	470
13	Tipps und Tricks	474
13.1	Überprüfen, ob ein Stylesheet gültig ist.	475
13.2	Ein Design auf mehreren Plattformen testen.	477
13.3	Browsererweiterungen zur Fehlersuche benutzen.	479
13.4	Bekannte Browser-Bugs und CSS-Probleme erkennen und beheben	481
13.5	Websites mit einem Textbrowser testen	483

13.6 Die richtige Zeichenkodierung angeben	483
13.7 Inhalte mit CSS erzeugen	486
13.8 Das Boxmodell anpassen	489
13.9 Float-Containern per Clearfix ihre Höhe wiedergeben.	491
13.10 HTML-Elemente dynamisch hervorheben	495
13.11 Elemente nur in Hilfsgeräten darstellen	497
13.12 Inline-Stile überschreiben	499
13.13 Stylesheets mit Conditional Comments nur für den Internet Explorer definieren	501
13.14 Stile für HTML5-Elemente im IE definieren	504
13.15 hasLayout-Probleme beheben	506
13.16 Reset-Stylesheets benutzen	509
13.17 Die Möglichkeiten von CSS mit Präprozessoren erweitern.	511
13.18 Älteren Browsern mit Polyfills auf die Sprünge helfen	513
13.19 CSS3-Eigenschaften in der Präfixschreibweise angeben	516
13.20 Daten-URIs verwenden	517
A Ressourcen	521
Index	529

Vorwort

Um ein richtig gutes Essen zuzubereiten, brauchen Sie vor allem eines: hervorragende Zutaten. Diese Einschätzung wird Ihnen jeder gute Koch bestätigen. Der Autor Christopher Schmitt hat die nötigen Einkäufe für Sie erledigt. Dank der Zusammenstellung von hundert Rezepten in diesem Buch haben Sie alles, was Sie brauchen, um flexible Webseiten mit Stil zu erstellen.

Als ich zum ersten Mal versuchte, mich in der wunderbaren Welt von CSS zurecht zu finden, war die Trial-and-Error-Methode mein wichtigstes Werkzeug, um die kreativen Möglichkeiten von CSS zu entdecken. Ich dachte bei mir: »Hm, die Liste würde ich gerne als Navigationsleiste darstellen.« Oder: »Ich muss die Elemente eines Webformulars für einen Kunden gestalten.« Ich verbrachte viele Stunden (oder gar Tage) damit, verschiedene CSS-Regeln zu erstellen, wieder zu entfernen und endlose Kombinationen durchzuspielen. Dieses Ausprobieren »auf gut Glück« funktionierte (manchmal), und auch wenn Außenstehende es sogar als Spaß bezeichnen könnten, hat der ganze Prozess doch ganz schön lange gedauert.

Damals wünschte ich mir ein Buch wie dieses. Anstatt die Lösung zum Gestalten aller Elemente unverhofft durch einen Zufall zu finden, hätte ich auch kurz das CSS Kochbuch durchblättern können, das richtige Rezept herausuchen und direkt mit dem Kochen anfangen können. All die Raterei wäre nicht mehr nötig gewesen, und ich hätte meine Zeit mit den Dingen verbringen können, die ich am besten kann: kreativ zu sein.

Der modulare Aufbau dieses Buchs macht es zu einem unverzichtbaren Referenzwerk für Designer und Entwickler gleichermaßen. Konfrontiert mit Problemen wie den typografischen Fallstricken im Web, der Gestaltung von Links oder ganzen Seitenlayouts, erklärt Christopher einerseits klar und verständlich, welche Stildefinitionen nötig sind, um die gestellten Anforderungen zu erfüllen, aber auch, welche Eigenheiten es bei den verschiedenen Browsern zu beachten gilt. Zusätzlich enthält das Buch hilfreiche Workarounds für die täglichen Probleme mit CSS und das nötige Wissen, um als Webdesigner erfolgreich zu sein.

Dieses Buch ist deshalb so nützlich, weil es den echten Problemen die richtigen Werkzeuge gegenüberstellt, um wirkliche Ergebnisse zu erzielen. Sie haben vielleicht schon davon gehört, dass CSS Ihr Leben und die Erstellung und Pflege von Webseiten erleichtern kann. Jetzt sind Sie dran, und mit diesem Buch haben Sie eine Ausrede weniger, es nicht zu tun.

Ich rate Ihnen, Platz auf Ihrem Schreibtisch zu schaffen, denn das CSS Kochbuch wird Ihnen ein ständiger Begleiter bei Ihrer täglichen Arbeit sein. Ich hoffe, dass es immer in Reichweite ist.

– Dan Cederholm, Gründer von SimpleBits (<http://www.simplebits.com>) über die 3., englische Auflage dieses Buches

Jedes Buch erzählt eine Geschichte – selbst eines über Webdesign.

In diesem Buch geht es um *Cascading Style Sheets*, kurz CSS. CSS besitzt eine einfache, standardisierte Syntax, die Webdesignern eine umfassende Kontrolle über die Darstellung einer Website ermöglicht. CSS ist ein wesentlicher Bestandteil des heutigen Webdesigns.

Verglichen mit den Möglichkeiten der 1990er-Jahre, können Webdesigner mit CSS das Design einer Site wesentlich präziser kontrollieren, und das mit einem deutlich geringeren Zeitaufwand. Dabei gehen die Möglichkeiten von CSS deutlich über die des traditionellen Webdesigns hinaus und schaffen ein Maß an Interaktivität, das noch vor wenigen Jahren nur mithilfe von JavaScript oder Flash möglich war.

Für die Erstellung von Stylesheets brauchen Sie keine spezielle Hard- oder Software. Die Grundvoraussetzungen sind ein einigermaßen aktueller Computer, ein moderner Browser wie z.B. Firefox, Google Chrome oder Internet Explorer ab Version 9 (um nur ein paar zu nennen) und Ihr bevorzugter Code-Editor. Hierfür können Sie alles einsetzen – vom einfachen Textprogramm wie Windows Notepad oder TextEdit unter Macintosh bis hin zum komplett ausgestatteten WYSIWYG-Editor wie beispielsweise Adobe Dreamweaver in der Codeansicht.

Bevor es richtig losgeht, hier noch ein wenig Geschichte: Manche Menschen sind der Meinung, das Webdesign hätte seinen Anfang genommen, als Tim Berners-Lee, der Erfinder des World Wide Web, die ersten Websites erstellt hat. Andere glauben, dass das Web seinen Anfang nahm, als Netscape HTML um das center-Tag erweiterte.

Auch wenn es vielleicht zunächst widersinnig erscheint, bin ich persönlich der Meinung, dass die Geburtsstunde des Webdesigns mit der Veröffentlichung der ersten Bücher zu diesem Thema zusammenfiel. Eines der ersten Bücher, die dabei halfen, den Dot-com-Boom der 90er anzuschieben, war das erste vierfarbige Buch über Webgrafiken. *Designing Web Graphics* von Lynda Weinman wurde im Januar 1996 veröffentlicht. Ein paar Monate später im gleichen Jahr folgte dann *Creating Killer Web Sites* von David Siegel. Diese beiden Bücher trugen meiner Meinung nach genauso stark dazu bei, die Webrevolution auf den Weg zu bringen, wie die Erfindung der Technologien selbst.

Obwohl die in diesen Büchern beschriebenen Methoden damals auf dem neuesten Stand waren, wirken sie im Licht der heutigen Möglichkeiten veraltet. Kein Wunder, in den letzten 15 Jahre ist viel passiert, sowohl, was das Design angeht, als auch die Webstandards betreffend.

In seinen Anfangstagen war CSS weder besonders robust noch sonderlich gut in die damals aktuellen Browser implementiert. Das ist inzwischen zum Glück anders: Während der letzten fünfzehn Jahre wurde CSS permanent weiterentwickelt und verbessert. Und so sind Webdesigner und Browserbenutzer eigentlich erst jetzt, mit der Einführung standardfreundlicher Browser, in der Lage, CSS und seine Möglichkeiten voll auszuschöpfen. Sogar das Sorgenkind unter den Browsern, der Internet Explorer, verhält sich inzwischen halbwegs standardkonform, und es ist zu hoffen, dass die künftigen Versionen in dieser Hinsicht noch stärker sein werden. Dennoch ist es zurzeit (Stand: März 2012) noch so, dass andere Browser dem IE insbesondere im Hinblick auf die Unterstützung von CSS3 und HTML5 noch ein Stück voraus sind.

Das *CSS Kochbuch* ist eine Sammlung von CSS-basierten Lösungen für Probleme, die häufig im Webdesign auftreten. Es soll Webdesignern und Entwicklern dabei helfen, die vielen mit CSS realisierbaren Designs und Techniken möglichst gut umzusetzen. Wenn Sie es ernst meinen und Websites erstellen möchten, die auf der Höhe der Zeit sind, sollten Sie CSS und das *CSS Kochbuch* einsetzen. Mit etwas Glück können Sie so selbst an der Geschichte des Webs mitschreiben.

Für diese Ausgabe des CSS Kochbuchs wurde die englische 3. Auflage des Buches herangezogen und einer gründlichen Überarbeitung unterzogen. Die besten Rezepte jener Auflage wurden ausgewählt und ergänzt um Rezepte zu aktuellen Themen, insbesondere im Zusammenhang mit CSS3. Den Besonderheiten, denen Webworker beim Design für mobile Endgeräte begegnen, wurde darüber hinaus ein eigenes Kapitel gewidmet.

Außerdem haben wir Techniken, die einem bei der Arbeit mit CSS immer wieder begegnen, in einem eigenen Kapitel zusammengefasst: In Kapitel 2, *Grundrezepte*, finden Sie eine Reihe von Rezepten, auf die später immer wieder Bezug genommen wird. So wird nur ein Mal an zentraler Stelle erklärt, wie beispielsweise ein Element mit einem Hintergrund versehen werden kann, und in den folgenden Kapiteln können wir uns dann auf die Lösung konkreter Probleme konzentrieren, die im Zusammenhang damit auftreten.

An wen sich dieses Buch richtet

Dieses Buch richtet sich an Webdesigner und -entwickler, die Grundkenntnisse in HTML mitbringen und die sich nun ganz auf das Design von Websites mithilfe von CSS konzentrieren möchten. Wenn browserseitig bereits eine breite CSS3-Unterstützung vorhanden ist, setzen wir in unseren Lösungen auf CSS3. Ist dem nicht so, basiert die jeweils beschriebene Lösung auf CSS 2.1.

Das *CSS Kochbuch* eignet sich gut für alle, die CSS für ihre Webprojekte benutzen möchten, aber bisher vor dem Erlernen einer neuen Technologie zurückgeschreckt sind. Wenn Sie solch ein Leser sind, können Sie die hier vorgestellten Lösungen nacheinander benutzen. Verwenden Sie das Buch als Leitfaden und nehmen Sie sich einen neuen Abschnitt vor, wenn Sie so weit sind bzw. wenn Sie sich eine neue Technik aneignen oder einen neuen Trick lernen möchten.

Auch wenn Sie schon CSS-Grundkenntnisse mitbringen, Sie Ihre Design-Kenntnisse aber noch einmal auffrischen möchten, ist dieses Buch hilfreich. Angefangen bei der Webtypografie über das Seitenlayout bis hin zu Designanregungen für komplette Webseiten finden Sie in diesem Buch umfassende und praxisnahe Informationen.

Was Sie schon wissen sollten

Ein paar Dinge setzen wir bei Ihnen, lieber Leser, voraus, beispielsweise, dass Sie bereits eine gewisse Erfahrung mit dem Design oder der Entwicklung von Webseiten besitzen. Hierbei spielt es keine Rolle, ob Sie Hobbydesigner, Student oder Profi sind.

Das *CSS Kochbuch* ist weder eine Einführung in CSS, noch geht es detailliert darauf ein, wie CSS in den verschiedenen Browsern funktionieren sollte. Daher werden diejenigen, die ihren Weg als Webdesigner oder -entwickler gerade erst begonnen haben, dieses Buch vermutlich etwas schwieriger finden als ein allgemeines Buch über die Theorie von CSS. Wenn Sie Literatur über Dinge wie die CSS 2.1-Spezifikation benötigen, sollten Sie sich einmal das Buch *CSS – Das umfassende Handbuch* (O'Reilly) ansehen, das eine solide Ergänzung zu diesem Buch darstellt.

Wenn Sie Programme wie Adobe Dreamweaver ausschließlich im »Design«-Modus verwenden, ohne direkt mit Markup und Stylesheets in der Codeansicht zu arbeiten, könnte es für Sie schwierig werden, den größtmöglichen Nutzen aus diesem Buch zu ziehen. In diesem Fall ist ein gutes Grundlagenbuch vermutlich ein geeigneterer Ausgangspunkt.

Obwohl Sie auch mit grafikbasierten Lösungen CSS-Designs erstellen können, sind einige dieser Werkzeuge nicht in der Lage, mit den in diesem Buch empfohlenen Ansätzen umzugehen. Um von diesem Buch zu profitieren, müssen Sie in der Lage sein, HTML und CSS von Hand zu editieren. Manche der Codebeispiele können zwar auch mit den dialogboxbasierten Webdesignprogrammen umgesetzt werden, allerdings kann es dabei leicht zu Problemen kommen.

Die »Browserhölle«, die vielen Webdesignern in den vergangenen Jahren Probleme bereitet hat und oft mit der browserunabhängigen Entwicklung in Verbindung gebracht wird, ist zwar deutlich abgekühlt, existiert aber immer noch. Sie rührt daher, dass manche Browserhersteller die CSS-Spezifikation individuell interpretieren oder nur teilweise implementieren. Frustration ist allerdings ein natürlicher Teil des Lernprozesses. Die wichtigsten Tugenden, die Sie zum Lernen von CSS benötigen, sind Geduld und ein guter Sinn für Humor.

Die gute Nachricht ist, dass die Hersteller der wichtigsten Browser einen Großteil der Probleme mittlerweile gelöst haben. Die aktuellen Browserversionen implementieren CSS weitgehend korrekt. Wenn Sie versuchen, auch für ältere oder weniger benutzte Versionen browserübergreifende Unterstützung zu bieten, kann dies allerdings immer noch eine große Herausforderung bedeuten. Dennoch überwiegen die Vorteile von CSS, inklusive der größeren Kontrolle über Aussehen und Handhabung von Webseiten («Look and Feel») und der wesentlich leichteren Wartbarkeit selbst von mehreren Sites auf einmal, die »Browserhölle« bei Weitem.

Wir haben in dieser Auflage fast komplett auf JavaScript-basierte Lösungen verzichtet. Dennoch werden Sie im Buch einige Verweise auf so genannte »Polyfills« finden. Dies sind häufig (aber nicht immer) in JavaScript geschriebene Bibliotheken, die Browser um bestimmte CSS3- oder auch HTML5-Fähigkeiten erweitern können.

CSS3 und HTML5

Sofern von den aktuellen Browsern ausreichend unterstützt, bieten wir Ihnen in diesem Buch Lösungen auf der Basis von CSS3 an. Sollten bestimmte Features noch nicht ausreichend unterstützt werden, finden Sie zudem eine Lösung, teilweise auch mehrere weitere Lösungsansätze, die auch mit älteren Browsern funktionieren.

Die Codebeispiele in diesem Buch verwenden weitestmöglich den neuen HTML5-Dokumententyp. Auch wenn HTML5 laut World Wide Web Consortium (W3C) noch »in Arbeit« ist, können Sie es gefahrlos auch mit älteren Browsern verwenden. Mehr hierzu finden Sie in Kapitel 1, *Allgemeines*.

Bei der Schreibweise der HTML-Tags haben wir uns an der Syntax von XHTML5 orientiert. Das heißt:

- Alle HTML-Tags werden kleingeschrieben, auch wenn die Groß- und Kleinschreibung in HTML5 bei Tagnamen unerheblich ist.
- Einzeln stehende Tags (z.B. `img`) werden als »self-closing tags« mit einem abschließenden Schrägstrich versehen, auch wenn dies nicht absolut notwendig ist.
- Es werden öffnende und schließende Tags verwendet, auch wenn dies nicht zwingend vorgeschrieben ist (z.B. bei `li`-Elementen).

Diese Regeln sollen Sie Ihnen dabei helfen, verständlichen, lesbaren und validierenden Code zu schreiben.

Aktuelle Browser

Gelegentlich verwenden wir in diesem Buch den Begriff »aktuelle Browser«. Hiermit meinen wir Browser, die bei der Arbeit an dieser Auflage eine ausreichend große Nutzerbasis aufweisen konnten. Dies sind:

- Firefox ab Version 3.6
- Internet Explorer ab Version 7
- Google Chrome (aktualisiert sich selbst)
- Safari ab Version 4
- Opera ab Version 10.6

Soweit möglich, weisen wir darauf hin, wenn es Probleme mit bestimmten Browsern gibt. In Kapitel 13, *Tipps und Tricks*, finden Sie eine Reihe von Rezepten, die Ihnen dabei Hilfestellung geben, wie Sie Stylesheets vor Browsern verstecken oder die Unterstützung für bestimmte Eigenschaften nachrüsten können.

Das ist jedoch keine Garantie dafür, dass Sie in jedem Browser eine pixelgenaue Darstellung erreichen können. Selbst bei den tabellenbasierten Webdesignmethoden der 90er-Jahre war das nie der Fall.

Benennung von Klassen und IDs

Sie werden in diesem Buch eine Reihe von Beispielen finden, die mit `class`- und `id`-Attributen arbeiten, um bestimmte Dokumentteile zu kennzeichnen. Teilweise haben die Namen beschreibenden Charakter, zum Beispiel `linke_spalte`.

Bitte beachten Sie, dass dies ausschließlich zu Demonstrationszwecken geschieht, um die Codebeispiele leichter verständlich zu machen. Auf einer Website, die im Netz erreichbar ist, die gepflegt und gewartet werden muss, sollten Sie auf Bezeichner dieser Art verzichten.

Verwenden Sie in Ihren Stylesheets stattdessen Klassen- und ID-Namen, die sich auf die *Funktion eines Elements* und nicht auf sein Aussehen oder seine Platzierung im Seitenlayout beziehen.

Dokumentkodierung

Damit es bei der Verwendung von Umlauten und Sonderzeichen (ö, ü, ä, ß usw.) nicht zu Darstellungsproblemen im Browser kommt, sollten Sie Ihre CSS- und HTML-Dateien prinzipiell im UTF-8-Format sichern. Die meisten Editorprogramme bieten hierfür im *Sichern als*-Dialog oder in den entsprechenden Voreinstellungen eine entsprechende Option an.

Inhalte dieses Buchs

Meistens benutze ich ein Buch wie dieses, indem ich es gelegentlich aufschlage, um ein bestimmtes Problem zu lösen. So habe ich übrigens auch die erste Auflage dieses Buchs benutzt, um mein Gedächtnis aufzufrischen. In diesem Sinne ist das Buch gut auf dem Schreibtisch eines Webdesigners oder in dessen Nähe aufgehoben – immer in Reichweite, falls ein CSS- oder Webdesignproblem auftritt. Natürlich können Sie das Buch auch einfach von vorn bis hinten durchlesen. Im Folgenden finden Sie eine Übersicht, was Sie in diesem Buch erwartet:

Kapitel 1, *Allgemeines*, befasst sich mit den CSS-Grundlagen sowie mit einigen Techniken, die mit den gängigen Vorgehensweisen (Best Practices) zusammenhängen. Wir zeigen, wie Sie CSS 2.1- und CSS3-Selektoren verwenden können, um gezielt bestimmte Elemente in einem HTML-Dokument auszuwählen, und erklären, wie das CSS-Boxmodell funktioniert, was Darstellungsrollen sind, wie die Vererbung und die Kaskade funktionieren und vieles mehr.

In Kapitel 2, *Grundrezepte*, finden Sie eine Reihe sehr häufig wiederkehrender Problemstellungen und deren Lösungen, wie z.B. die Definition von Höhen und Breiten, Außen- und Innenabständen, Rahmen und Farben. Wir beschäftigen uns hier ebenso mit der Verwendung von Hintergrundfarben und -bildern wie der Positionierung, Zentrierung und dem Umfließen von Elementen.

Kapitel 3, *Typografie für das Web*, befasst sich mit der Definition verschiedener typografischer Merkmale wie Schriftfamilie, -gewicht und -größe sowie den Abständen zwischen Buchstaben, Wörtern und Zeilen. Natürlich zeigen wir Ihnen hier auch, wie Sie Webfonts in Ihre Seiten einbinden können, wie Sie typografisch korrekte Anführungszeichen, speziell gestaltete Initialen und andere Dinge definieren können.

Kapitel 4, *Bilder*, beschäftigt sich mit CSS-Techniken, die direkt mit der Manipulation von Stilen und Eigenschaften für Webgrafiken zu tun haben.

Kapitel 5, *Listen*, beschreibt, wie die elementaren Listenelemente auf verschiedene Weise mit Stilen versehen werden können. Unter anderem geht es um browserunabhängige Einrückungen, die Verwendung eigener Bilder als Marker für Listeneinträge und noch einiges mehr.

Kapitel 6, *Links und Navigation*, zeigt, wie Sie CSS für die Darstellung eines oder mehrerer Links nutzen können. Das Spektrum reicht von einfachen Lösungen wie dem Entfernen der Unterstreichung bis zu komplexen Dingen wie dem Erzeugen hierarchischer Menüs mit mehreren Ebenen.

Kapitel 7, *Formulare*, zeigt Wege auf, wie Sie die Darstellung von Formularen in Browsern beeinflussen. Die Lösungen behandeln unter anderem die Definition von Stilen für bestimmte Formularelemente und die Gestaltung der Beschriftungen für einzelne Elemente und Elementgruppen. Am Schluss dieses Kapitels finden Sie einen »Serviovorschlag«, der die einzelnen Rezepte in einem praktischen Beispiel zusammenführt.

Kapitel 8, *Tabellen*, befasst sich mit der Definition von Stilen für HTML-Tabellen. CSS kann helfen, tabellenbasierte HTML-Seiten zu vermeiden. Dennoch gibt es Tabellendaten, die Sie mit Stilen versehen möchten, wie beispielsweise Kalender oder statistische Informationen. In diesem Kapitel finden Sie Lösungen für das Festlegen eines Innenabstands in Zellen, das Entfernen von ungewollten Zwischenräumen in Tabellenzellen mit Bildern, das Hervorheben »aktiver« Tabellenzeilen und -spalten, die Definition abwechselnder Hintergründe und anderes.

Kapitel 9, *Drucken*, enthält Informationen über die Definition von Stilen, die beim Druck von Webseiten verwendet werden sollen. Unter anderem finden Sie hier Lösungen für das Definieren eines separaten Stylesheets zum Druck von Webseiten, für das Anlegen von Stilen für Webformulare und für das Einfügen von URLs nach Links.

Kapitel 10, *Mobile Geräte*, beschreibt die Aspekte von CSS, die für die Darstellung von Websites auf Mobilgeräten wichtig sind. Unter anderem finden Sie hier umfassende Informationen zur Verwendung von CSS3-Media Queries, Rezepte zur Verbesserung der Ladegeschwindigkeit in Mobilgeräten sowie zur Linearisierung von Seitenlayouts und Tabellen.

Kapitel 11, *Seitenlayouts*, beschreibt die Verwendung von CSS zur Erstellung von verschiedenen Layouts. Die Lösungen in diesem Kapitel zeigen unter anderem Methoden für die Erstellung mehrspaltiger Seitenlayouts sowie die Verwendung von Grid-Systemen, und selbst den »heiligen Gral« können Sie hier finden!

In Kapitel 12, *Design mit CSS*, finden Sie eine Reihe von Lösungen, die hauptsächlich mit der grafischen Gestaltung von Webseiten zu tun haben. Beispielsweise geht es hier um die Definition von Rollover-Effekten, Farbverläufen, abgerundeten Ecken, Schattenwürfen und einfachen Animationen mit CSS3. Anhand von modularen Skalen und Grundlinienrastern zeigen wir, was Webdesign und Musik gemeinsam haben. Abgerundet wird das Kapitel mit einem Rezept zur Erzeugung quasi zufälliger Hintergrundmuster.

Kapitel 13, *Tipps und Tricks*, enthält Lösungen, die Ihnen bei Ihrer täglichen Arbeit mit CSS helfen können. Es geht um das Testen und Validieren Ihrer Stylesheets, die Behebung von Problemen, die in bestimmten Versionen des Internet Explorer auftreten können, und darum, wie Sie auch ältere Browser mit den Fähigkeiten von CSS3 und HTML5 ausstatten können.

In Anhang A, *Ressourcen*, finden Sie eine Sammlung von Links, Websites und Büchern, die weitere Informationen über CSS enthalten.

Typografische Konventionen

In diesem Buch werden die folgenden typografischen Konventionen verwendet:

Kursivschrift

Kennzeichnet neue Begriffe, Menübefehle, Menüs, URLs, E-Mail-Adressen, Dateinamen und -endungen, Pfadnamen und Verzeichnisse.

Nichtproportionalschrift

Kennzeichnet Befehle, Optionen, Attribute, Funktionen, Typen, Klassen, Eigenschaften, Werte, Objekte, Events, Event-Handler, XML- und HTML-Tags, den Inhalt von Dateien und die Ausgabe von Befehlen.

Nichtproportionalschrift fett

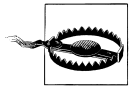
Wird innerhalb von Listings zur Hervorhebung genutzt.

Nichtproportionalschrift kursiv

Steht für Text, der durch Benutzereingaben ersetzt werden soll.



Dieses Icon steht für einen Tipp, einen Vorschlag oder einen allgemeinen Hinweis.



Dieses Icon steht für eine Warnung.

Codebeispiele zu diesem Buch

Die in diesem Buch abgedruckten Codebeispiele stehen Ihnen online zur Verfügung. Sie finden den Code auf unserer Verlagswebsite unter der folgenden URL:

http://examples.oreilly.de/german_examples/cssckbk3ger

Benutzung der Codebeispiele

Dieses Buch soll Ihnen bei Ihrer Arbeit helfen. Es ist prinzipiell erlaubt, den Code dieses Buchs in Ihren Webseiten und Ihrem Design weiterzuverwenden. Hierfür ist es nicht notwendig, uns um Erlaubnis zu fragen, es sei denn, es handelt sich um eine größere Menge Code. So ist es beim Schreiben eines Programms, das einige Codeschnipsel dieses Buchs verwendet, nicht nötig, sich mit uns in Verbindung zu setzen; beim Verkauf oder Vertrieb einer CD-ROM mit Beispielen aus O'Reilly-Büchern dagegen schon. Das Beantworten einer Frage durch Zitieren von Beispielcode erfordert keine Erlaubnis. Verwenden Sie einen erheblichen Teil der Beispielcodes aus diesem Buch in Ihrer Dokumentation, ist jedoch unsere Erlaubnis nötig.

Eine Quellenangabe ist zwar erwünscht, aber nicht unbedingt notwendig. Hierzu gehört in der Regel die Erwähnung von Titel, Autor, Verlag und ISBN, zum Beispiel: »CSS Kochbuch, 3. Auflage, von Christopher Schmitt und Jørgen W. Lang. Copyright 2012 O'Reilly Verlag, ISBN 978-3-89721-324-1«.

Sollten Sie nicht sicher sein, ob die Nutzung der Codebeispiele außerhalb der hier erteilten Genehmigung liegt, nehmen Sie bitte unter der Adresse permissions@oreilly.com Kontakt mit uns auf.

Danksagungen von Christopher Schmitt

Vielen Dank an David Siegel und Lynda Weinman für ihre Inspiration und Unterstützung seit dem Beginn des Webdesigns.

Ohne die Unterstützung und Freundschaft von Molly Holzschlag hätte ich nie ein Buch für eine Branche geschrieben, die ich so sehr mag.

Eine Menge Wertschätzung und Respekt gilt meinen Mit-Webdesignern Douglas Bowman, Tantek Çelik, Dan Cederholm, Mike Davidson, Ethan Marcotte, Eric A. Meyer, Mark Newhouse, Dave Shea und Jeffrey Zeldman dafür, dass sie das Webdesign mithilfe von CSS so weit vorangebracht haben.

Mein besonderer Dank gilt den technischen Gutachtern Erik J. Barzeski, Liza Daly und John Allsopp sowie der Lektorin dieser Ausgabe, Nancy Reinhardt, für ihre Zeit, ihre Sachkenntnis und ihre Geduld.

Danke auch an meinen Freund Porter Glendinning, der nicht nur die Gabe hat, W3C-Spezifikationen zu lesen und ihre Implikationen drei oder vier Schritte vor den meisten Webentwicklern zu erkennen, sondern diese Gedanken auch noch so formulieren kann, dass vermutlich sogar meine Großmutter sie verstehen könnte. Deine Übersetzungsdienste schätze ich sehr.

Tatiana Diaz hat hervorragende Arbeit geleistet, als es darum ging, meine Fragen zu beantworten und mich durch dieses Projekt zu führen. Das Schreiben dieses Buchs war für mich bisher bei Weitem die größte Herausforderung und Bestätigung. Und um ehrlich zu sein: Bei einem anderen Verlag hätte es nicht anders sein dürfen.

Danke an meine Freunde, die mich als den Web-Geek kennen, der ich nun einmal bin: Katrina Ferguson, Kelly and Nathan Hensley, Trueman Muhrer, Eric Ellis, Jessica Lorenzi, Ansley Simmons, Mark Trammell und Ryan Yordon.

Danke auch an meine Familie für ihre Liebe und Bestätigung. Eure Unterstützung in guten wie in schlechten Zeiten war für mich wie ein Fels in der Brandung. Ich freue mich wie immer schon jetzt auf unser nächstes Treffen.

Danksagungen von Jørgen W. Lang

Selbst ein scheinbar schlichtes Projekt wie die Überarbeitung eines Buchs lässt sich kaum ohne die Hilfe anderer Menschen realisieren. Hierzu gehört besonders meine Lektorin Inken Kiupel, die trotz aller Hindernisse und Unwägbarkeiten das Steuer fest in der Hand hielt. Sie sorgte mit Geduld, Ausdauer und Beharrlichkeit dafür, dass das Schiff auch im Sturm auf Kurs blieb und in unsicheren Gewässern nicht auf Grund lief. Danke dafür!

Ein weiteres Dankeschön gebührt meinen beiden technischen Gutachtern Jens Grochtdreis und Jens O. Meiert, die das Buch mit kritischem Sachverstand auf Fehler und Schwächen abgeklopft haben. Es hat sich gelohnt!

Bedanken möchte ich mich auch bei Angelika Metzler von der Webagentur *pointz.de* aus Karlsruhe, die mir besonders bei Fragen zum Thema Barrierefreiheit zur Seite stand.

Danke übrigens auch an alle freien und festen Verlagsmitarbeiter, die eher im Hintergrund tätig sind, sei dies nun als Grafiker, in der Herstellung, beim Korrektorat, Satz oder an irgendeiner anderen Stelle. Ohne euch geht's nicht!

Meine tiefste Dankbarkeit gilt allerdings meiner Verlobten Armelle für »the all« und meinem Sohn Mathis, der den Beginn der Arbeiten an diesem Buch noch im Bauch seiner Mutter miterlebte und der mich in den letzten Monaten tatkräftig beim Schreiben unterstützt hat. *Merci à vous pour être ma famille!*

Einführung

Durch den Einsatz von Cascading Style Sheets (CSS) können Sie Ihre Webseiten auf einfache Weise gestalten, ohne die gleichen Anweisungen für jedes Element (oder auch jedes HTML-Dokument) einzeln anzugeben. Tatsächlich ist CSS deutlich einfacher, als es auf den ersten Blick wirkt. Das wollen wir Ihnen in diesem Kapitel zeigen. Die folgenden Rezepte vermitteln Ihnen die Grundlagen für einen schnellen Einstieg in die Arbeit mit CSS. Schreiben Sie einfach ein paar Zeilen HTML-Code und fügen Sie ein wenig CSS hinzu – schon sind die Ergebnisse sichtbar.

In diesem Kapitel beschäftigen wir uns damit, was Sie grundsätzlich für Ihre Arbeit mit CSS brauchen: mit dem Einbinden von Stylesheets in Ihre Webseiten, dem Aufbau einer CSS-Regel und der Auswahl bestimmter Elemente. Außerdem erklären wir Ihnen, was es mit den verschiedenen Darstellungsrollen (Block, Inline, Liste, Tabelle usw.) auf sich hat, was unter der Vererbung zu verstehen ist und woher die Cascading Style Sheets ihren Namen haben.

Sie werden außerdem lernen, wie das Boxmodell funktioniert, wie Sie bestimmten Regeln Vorrang vor anderen Anweisungen geben und natürlich auch wie Sie Stylesheets in Ihre HTML-Seiten einbinden können.

Und jetzt ab in die Küche!

1.1 Eine Webseite mit Stildefinitionen versehen

Problem

Sie wollen wissen, wie Sie die in diesem Buch vorgestellten Rezepte in einer Webseite nutzen können.

Lösung 1

Verwenden Sie ein externes Stylesheet. Hierfür müssen Sie eine zusätzliche Textdatei erstellen, die Sie mit der Dateiendung `.css` versehen. Der Inhalt ist denkbar einfach:

```
/* externes Stylesheet */  
p { font-family: sans-serif; }
```

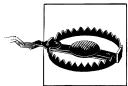
Das ist tatsächlich schon alles. Anstatt die Regeln direkt in der HTML-Datei zu definieren, wie in der zweiten Lösung gezeigt, stehen die Stildefinitionen in einer eigenen Datei. Damit der Browser Ihre Regeln auch »sieht«, müssen Sie im HTML-Dokument einen Verweis auf die Stylesheet-Datei anlegen. Dies ist die bei Weitem häufigste Methode, Stylesheets in eine Webseite einzubinden.

Für dieses Beispiel gehen wir davon aus, dass Ihr Stylesheet im selben Verzeichnis liegt wie die HTML-Datei und unter dem Namen `externes_stylesheet.css` gespeichert wurde.

Um das Stylesheet einzubinden, erweitern Sie den `head`-Teil des HTML-Dokuments um ein passendes `link`-Tag, wie hier gezeigt:

```
<!DOCTYPE html>  
<html>  
<head>  
  <title>CSS Kochbuch</title>  
  
  <link href="externes_stylesheet. css" rel="stylesheet" type="text/css" />  
  
</head>  
<body>  
  <p>Dieser Absatz wird in einer serifenlosen Schrift dargestellt.</p>  
</body>  
</html>
```

Dieses Vorgehen zahlt sich aus, sobald Sie mehr als ein HTML-Dokument mit den gleichen Stilen versehen müssen. Anstatt die CSS-Regeln in jeder Webseite zu wiederholen, reicht es aus, das externe Stylesheet per `<link>`-Tag einzubinden. Sie müssen also nicht jede einzelne Seite warten, sondern können die Änderungen in einer zentralen CSS-Datei vornehmen.



Beachten Sie, dass das HTML-Tag `<style>...</style>` in der externen CSS-Datei nicht verwendet wird. Auch HTML-Kommentare haben in einer solchen Datei nichts zu suchen.

Eine weitere Möglichkeit, externe Stylesheets einzubinden, besteht in der Verwendung einer `@import`-Direktive:

```
<!DOCTYPE html>  
<html>  
<head>  
  <title>CSS Kochbuch</title>
```

```

<style>
  @import url("externes_stylesheet.css");
</style>

</head>
<body>
  <p>Dieser Absatz wird in einer serifenlosen Schrift dargestellt.</p>
</body>
</html>

```

Da es sich bei der @import-Anweisung um CSS-Code handelt, muss diese Anweisung *im HTML-Dokument* von style-Tags umgeben werden. Ein weiterer Vorteil liegt darin, dass Sie auf diese Weise noch andere Stylesheets einbinden. So könnte die Datei *externes_stylesheet.css* weitere @import-Anweisungen enthalten, die zusätzliche CSS-Dateien laden.

Hierbei sollten Sie sich darüber klar sein, dass jede zusätzlich eingebundene CSS-Datei separat vom Server angefordert werden muss, was die Ladezeit der Webseite erhöhen kann.



Während der Entwicklung einer Website kann es hilfreich sein, die CSS-Regeln für verschiedene Aspekte der Website in eigenen Dateien zu sichern. Im Produktionsbetrieb können Sie die Ladezeit verringern, indem Sie die Regeln in einer gemeinsamen Datei speichern. (Noch schneller geht es, wenn Sie die Datei serverseitig beispielsweise mit Gzip komprimieren.)

Lösung 2

Verwenden Sie ein internes Stylesheet (siehe Abbildung 1-1). Hierfür versehen Sie den head-Teil Ihres HTML-Dokuments mit <style>...</style>-Tags. Zwischen den Tags können Sie nun Ihr Stylesheet definieren. Hier ein einfaches Beispiel:

```

<!DOCTYPE html>
<html>
<head>
  <title>CSS Kochbuch</title>
  <style type="text/css">
    p { font-family: sans-serif; }
  </style>
</head>
<body>
  <p>Dieser Absatz wird in einer serifenlosen Schrift dargestellt.</p>
</body>
</html>

```



Das Verwenden des type-Attributs im öffnenden style-Tag ist nicht verpflichtend. Wenn Sie Ihr Stylesheet mit einem Validierungsdienst wie <http://jigsaw.w3.org/css-validator/> überprüfen, erhalten Sie aber möglicherweise eine entsprechende Warnung, wenn das Attribut nicht angegeben wurde.

Das Ergebnis sehen Sie in Abbildung 1-1.



Abbildung 1-1: Mithilfe eines internen Stylesheets wird der Absatz in einer serifenlosen Schrift dargestellt

Lösung 3

Verwenden Sie Inline-Stile, die mithilfe des Attributs `style` direkt in den Markup-Code eingebettet werden. Unsere Beispielseite könnte etwa so aussehen:

```
<!DOCTYPE html>
<html>
<head>
  <title>CSS Kochbuch</title>
</head>
<body>
  <p style="font-family: sans-serif;">Dieser Absatz wird in einer
    serifenlosen Schrift dargestellt.</p>
</body>
</html>
```

Anstatt die Stile in einem internen oder externen Stylesheet zu definieren, stehen sie in diesem Fall direkt im HTML-Code. Das kann in seltenen Ausnahmefällen recht praktisch sein. Meistens überwiegen die Nachteile den Nutzen jedoch deutlich.

Diskussion

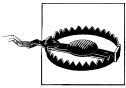
Die drei verschiedenen Arten von Stylesheets sind:

Extern

Mithilfe von `link`-Tags bzw. `@import`-Anweisungen können Sie externe CSS-Dateien in Ihre Webseiten einbinden. Diese dürfen ausschließlich CSS-Code enthalten. Mit externen Stylesheets können Sie Stile für mehrere Seiten auf einmal definieren, ohne die Regeln neu formulieren zu müssen. Das heißt, eine Änderung im externen Stylesheet wirkt sich automatisch auf alle Seiten aus, die die CSS-Datei einbinden.

Im Prinzip können Sie übrigens so viele `link`-Tags oder `@import`-Anweisungen verwenden, wie Sie wollen. Allerdings sollten Sie sich darüber klar sein, dass der Browser jedes eingebundene Dokument einzeln vom Webserver anfordern muss.¹

Der Autor Steve Souders empfiehlt in seinem Buch *High Performance Websites* (deutsche Ausgabe, O'Reilly, 2007), die `link`-Methode zu verwenden, um die Dateien parallel anstatt nacheinander zu laden.



Beachten Sie, dass per `url()` definierte Pfadangaben im Stylesheet, z.B. für Hintergrundbilder, immer vom Stylesheet aus angegeben werden. Liegen Ihre CSS-Dateien beispielsweise in einem Unterverzeichnis, müssen Sie dies bei der Angabe der Dateipfade berücksichtigen.

Intern

Wie in Lösung 2 gezeigt, werden interne Stylesheets direkt im `head`-Teil eines HTML-Dokuments definiert. Hier können Sie Stile für alle Elemente der betreffenden Seite angeben. Das kann sinnvoll sein, wenn andere Seiten diese Angaben nicht zu sehen bekommen sollen oder wenn Sie die Auswirkungen Ihrer Definitionen zuerst im kleinen Rahmen testen wollen. Sobald Sie die Regeln in mehr als einem HTML-Dokument wiederverwenden möchten, empfiehlt sich jedoch die Benutzung eines externen Stylesheets.

Zwischen den `<style>...</style>`-Tags dürfen keine weiteren HTML-Elemente stehen. Dieser Bereich ist ausschließlich für CSS-Code reserviert.

Inline

Inline-Stile werden als `style`-Attribut innerhalb von HTML-Tags definiert. Ihre Gültigkeit beschränkt sich auf den Inhalt des Tags, in dem sie definiert wurden. Die Verwendung von Inline-Stile wird in der Regel vermieden, da sie die Trennung von Inhalt und Darstellung untergraben. Sollten Sie davon ausgehen, dass Sie die Stildefinitionen ändern oder wiederverwenden werden, empfiehlt sich der Gebrauch von `id`-Attributen zusammen mit internen oder externen Stylesheets.

Siehe auch

Den Abschnitt zum Thema Stylesheets in der HTML 4.01-Spezifikation unter <http://www.edition-w3c.de/TR/1999/REC-html401-19991224/present/styles.html>; den Artikel *Anbindung von CSS an HTML* unter <http://www.thestyleworks.de/basics/applycss.shtml>; die Website von Steve Souders mit einem Vergleich von `@import` und `<link>` unter <http://www.stevesouders.com/blog/2009/04/09/dont-use-import/>; einen Artikel zur Verwendung des `type`-Attributs unter <http://www.maxdesign.com.au/2011/03/02/css-validator/>.

¹ Also gut. Nicht ganz. Im Internet Explorer 7 und 8 können Sie nur maximal 31 CSS-Dateien laden. Wenn das nicht ausreicht, sollten Sie Ihre Website-Strategie noch einmal genau überdenken.

1.2 Den Aufbau einer CSS-Regel verstehen

Problem

Sie wollen wissen, wie eine CSS-Regel aufgebaut ist.

Lösung

CSS-Regeln bestehen aus zwei Teilen: einem *Selektor* und einem *Deklarationsblock*. Der Selektor ist, wie sein Name schon sagt, dafür zuständig, das oder die Elemente auszuwählen, auf die die Regel angewandt werden soll. Auf den Selektor folgt, umgeben von geschweiften Klammern, der Deklarationsblock.

Innerhalb dieses Blocks befinden sich eine oder mehrere *Deklarationen*, mit denen festgelegt wird, wie bestimmte HTML-Elemente dargestellt werden sollen. Eine Deklaration hat ebenfalls zwei Teile: eine *Eigenschaft*, die sich auf ein bestimmtes Darstellungsmerkmal bezieht (z.B. Hintergrundfarbe, Schriftart und -größe usw.), sowie den dazugehörigen *Wert*, der bestimmt, wie das Merkmal dargestellt werden soll. Eigenschaft und Wert werden durch einen Doppelpunkt getrennt, die Deklaration wird mit einem Semikolon beendet. Die allgemeine Form einer CSS-Regel sieht so aus:

```
Selektor {  
    Eigenschaft1: Wert;  
}
```

Eine tatsächliche Anwendung könnte beispielsweise so aussehen:

```
li {  
    list-style-type: square;  
}
```

Diese Regel sorgt dafür, dass jedes Mal, wenn das Element `li` im Dokument auftaucht, als Aufzählungszeichen (Bullet) ein Quadrat anstelle des standardmäßigen Kreises angezeigt wird.

Diskussion

Selektoren

Mit *Selektoren* legen Sie fest, für welche Elemente eines Dokuments die Stildefinitionen gelten sollen. So können Sie beispielsweise Regeln definieren, die für alle Links gelten oder nur für Bilder, die ein ganz bestimmtes `class`-Attribut besitzen und sich z.B. im Hauptteil einer Seite befinden.

In den folgenden Rezepten werden wir die einzelnen Selektoren detailliert vorstellen und zeigen, wie sie benutzt werden können, um bestimmte Teile eines Dokuments auszuwählen.

Deklarationen

Deklarationen sind die Anweisungen, die festlegen, wie die ausgewählten Elemente gestaltet werden sollen. Eine CSS-Regel kann beliebig viele Deklarationsblöcke enthalten, die durch Semikola voneinander getrennt werden, wie hier gezeigt:

```
Selektor {  
    Eigenschaft1: Wert;  
    Eigenschaft2: Wert;  
    Eigenschaft3: Wert;  
}
```



Gewöhnen Sie sich an, Deklarationen stets mit einem Semikolon abzuschließen, auch wenn es nach der letzten Deklaration eigentlich nicht nötig ist. Sobald Sie Ihre Regeln erweitern oder verändern, kann das enorm bei der Vermeidung von Fehlern helfen.

Eigenschaften

Eigenschaften und *Werte* bestimmen, wie die verschiedenen Merkmale eines Elements dargestellt werden. Zu den Eigenschaften gehören Dinge wie Schrift- und Hintergrundfarben, Rahmen sowie die Breite und Höhe eines Elements (z.B. eines Bilds, eines Links, eines Listeneintrags usw.). Auch Abstände der Elemente zueinander und ihre Position in der Darstellung lassen sich mit CSS-Eigenschaften steuern.

Die Namen der Eigenschaften entsprechen in den meisten Fällen dem Darstellungsmerkmal, das Sie steuern. So legt die Eigenschaft `color` beispielsweise fest, dass die Vordergrundfarbe (meistens die Schriftfarbe) eines Elements verändert werden soll, während `width` die Breite eines Elements definiert.

Kurzschrifteigenschaften

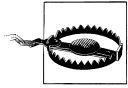
In manchen Fällen bedarf es einer ganzen Reihe von Eigenschaften, um beispielsweise den Rahmen um ein Element, die Schriftgestaltung oder die Abstände um ein Element herum zu definieren.

Für diese Fälle haben sich die Autoren der CSS-Spezifikationen die sogenannten *Kurzschrifteigenschaften* ausgedacht. Hiermit können Sie die Werte bestimmter Eigenschaften in einer gemeinsamen Deklaration angeben. Angenommen, Sie haben eine Regel wie diese:

```
h1 {  
    font-style:    italic;  
    font-variant:  small-caps;  
    font-weight:   bold;  
    font-size:     3em;  
    line-height:   1.2em;  
    font-family:   Times, "Times New Roman", serif;  
}
```

Dann könnten Sie diese Angaben auch folgendermaßen zusammenfassen:

```
h1 {  
  font: italic small-caps bold 3em/1.2 Times, "Times New Roman", serif;  
}
```



Kurzschrifteigenschaften haben eine Besonderheit: Werte, die nicht ausdrücklich angegeben wurden, werden auf ihren Standardwert zurückgesetzt. Die Angabe `background: #0000ff` ist also nicht das Gleiche wie `background-color: #0000ff`.

Kurzschrifteigenschaften kommen in diesem Buch immer wieder vor. Ihre Verwendung wird bei Bedarf in den einzelnen Rezepten genauer erklärt. Näheres zur Kurzschrifteigenschaft `font` und ihren Eigenheiten finden Sie in Kapitel 3, *Typografie für das Web*.

Werte

Auf ähnliche Weise wie `color` legt die Eigenschaft `background-color` die Hintergrundfarbe eines Elements fest. Sie bestimmt aber nicht, welche Farbe das sein soll – das ist Aufgabe des Werts.

Zu jeder Eigenschaft passen nur ganz bestimmte Werte, die sich nach dem Verwendungszweck der Eigenschaft richten. So ist es wenig sinnvoll, als Hintergrundfarbe (`background-color`) einen Wert von 3 Pixeln (3px) zu vergeben – hier ist stattdessen eine Farbangabe (z.B. `ffffff`) angebracht.

Manchmal kann ein Wert auf verschiedene Weise angegeben werden, etwa in Pixeln oder als Prozentwert, als RGB-Farbe (`rgb(255,255,255)`), als Hexadezimalwert (`ffffff`) oder als Schlüsselwort (`white`).

Außerdem hat jede Eigenschaft einen Standardwert. Dadurch zeigen die meisten Browser Webseiten auch dann mit einem weißen Hintergrund an, wenn Sie dies nicht ausdrücklich definiert haben.

Siehe auch

Die folgenden Rezepte zur Verwendung der CSS-Selektoren; die vollständige Eigenschaftentabelle des W3C unter <http://www.w3.org/TR/CSS21/propidx.html>; eine Tabelle der neuen CSS3-Eigenschaften unter <http://meiert.com/en/indices/css-properties/>; eine deutschsprachige Liste der CSS-Eigenschaften unter <http://www.css-wiki.com>; eine deutschsprachige Liste der verschiedenen CSS-Selektoren unter <http://css-selektoren.mozork.de/>; das *Values and Numbers*-Modul der CSS3-Spezifikation unter <http://www.w3.org/TR/css3-values/>.

1.3 Verschiedene Selektoren für die Zuweisung von Stilen verwenden

Problem

Sie wollen festlegen, für welche Teile eines HTML-Dokuments Ihre Stildefinitionen gültig sein sollen.

Lösung

Bevor Sie festlegen, wie bestimmte Teile eines Dokuments gestaltet werden, müssen Sie diese im Stylesheet auswählen. Genau hierfür gibt es die Selektoren. Sie unterscheiden sich je nach Zweck. So gibt es Selektoren, die nur auf bestimmte Elemente (Absätze, Überschriften usw.) zutreffen. Andere finden »alle Elemente« oder Elemente, deren `class`-Attribut einen bestimmten Wert hat. Außerdem können Sie Selektoren kombinieren, um bestimmte Zusammenhänge auszudrücken, wie »Links, die in einem Listenelement enthalten sind« oder »nur Absätze, die Kindelemente des `body`-Elements sind«.

Sie finden in diesem Kapitel eine Reihe von Rezepten, die die einzelnen Selektoren detailliert erklären und ihre Besonderheiten aufzeigen. Hier erhalten Sie zunächst einen kleinen Überblick. In den Kommentaren (dem Text zwischen den Zeichen `/*` und `*/`) wird kurz erklärt, was die Selektoren jeweils auswählen. Das Ergebnis sehen Sie in Abbildung 1-2.

```
/* "alle Elemente" */
* {
    font-family: Verdana, Arial, sans-serif;
}
/* Überschriften erster Ordnung */
h1 {
    font-size: 120%;
}

/* Das Element mit dem id-Attribut "navigation" */
#navigation {
    border: 1px solid black;
    padding: 40px;
}

/* Elemente mit dem class-Attribut "warnung" */
.warnung {
    color: red;
}

/* Links, die in einem Listenelement enthalten sind */
li a {
    text-decoration: none;
}
```

```

/* Absätze, die direkt im body-Element enthalten sind */
body > p {
    font-size: 90%;
}

```

Wenn Ihnen das jetzt ein wenig konfus vorkommt, lesen Sie bitte einfach weiter. Bevor wir im Detail auf die einzelnen Selektoren eingehen, sollten Sie aber noch ein paar grundsätzliche Dinge wissen.



Abbildung 1-2: Webseite mit CSS-Stildefinitionen

Diskussion

CSS bietet viele – manchmal ziemlich raffinierte – Möglichkeiten, um festzulegen, auf welche Seitenelemente die Stile angewandt werden sollen.

Um besser zu verstehen, wie die einzelnen Teile einer Webseite ausgewählt werden, müssen Sie wissen, dass durch die Markierung der Inhalte eines Dokuments mit HTML-Code eine Baumstruktur entsteht. Es ergibt sich eine feste Ordnung, auch wenn das auf den ersten Blick nicht so aussieht (siehe Abbildung 1-3).

Zwar bleibt diese Struktur dem Besucher einer Seite meistens verborgen, dennoch ist sie ein wesentlicher Teil des Darstellungsprozesses, den der Browser durchläuft.

Wenn der Browser eine Seite vom Server anfordert und mit ihrer Darstellung beginnt, sorgt die Software dafür, dass die verschiedenen Elemente der Seite in einer baumartigen Hierarchie, dem sogenannten *Document Object Model* (DOM), organisiert werden, wie in Abbildung 1-4 gezeigt.

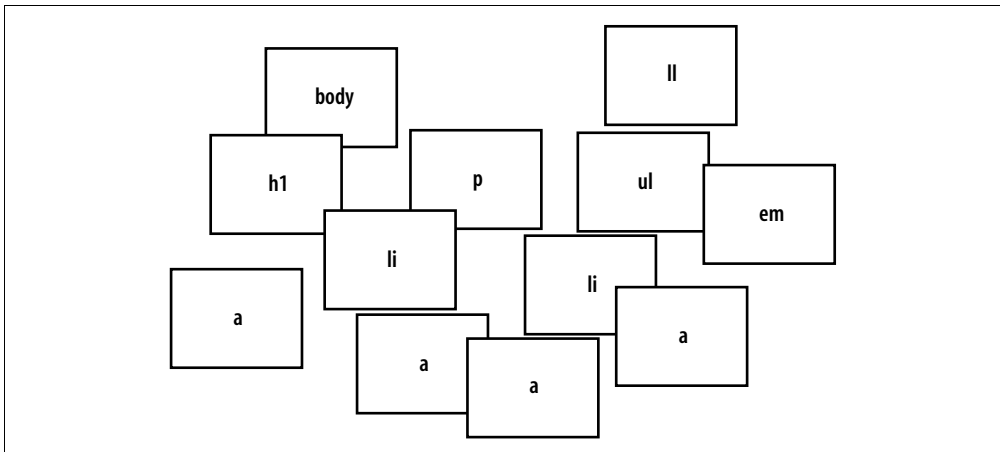


Abbildung 1-3: Die in der Lösung verwendeten Elemente

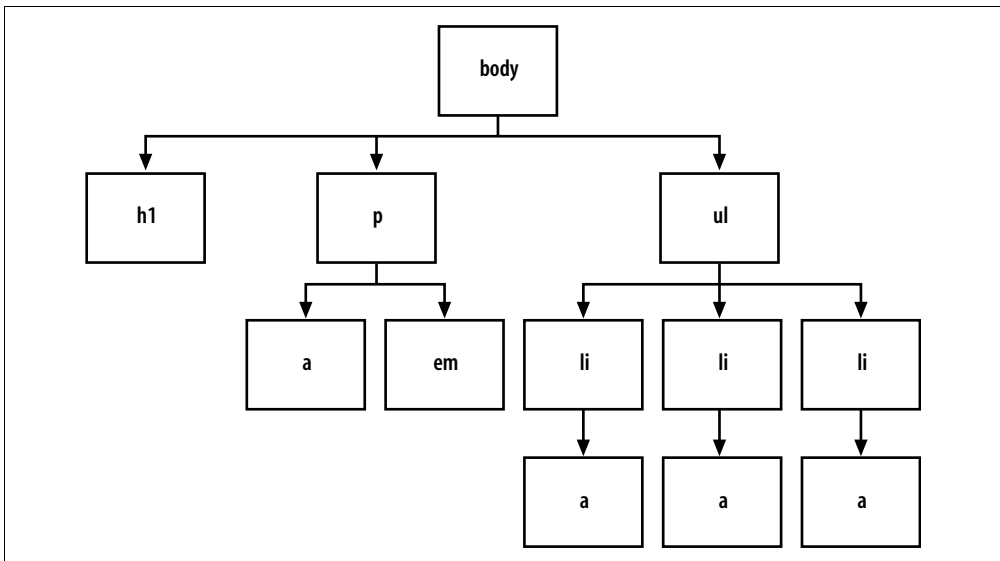


Abbildung 1-4: Eine vereinfachte Darstellung des DOM

Auf das DOM setzt CSS auf, um einzelne oder mehrere Elemente auszuwählen und die Beziehungen der einzelnen Elemente zueinander zu benennen. Die Struktur beginnt normalerweise mit dem `html`-Element, von dem wie bei einem Familienstammbaum alle anderen Elemente abstammen. Daher werden Sie in CSS auch oftmals Begriffe wie »Vorfahrelement« oder »Kindelement« finden. Die Beziehungen der einzelnen Elemente zueinander sehen Sie in Tabelle 1-1.

Tabelle 1-1: Beziehungen der Elemente in der Struktur eines HTML-Elements

Bezeichnung	Beschreibung
Vorfahrenelemente	Alle Elemente, die innerhalb der Struktur oberhalb von anderen Elementen stehen. Das body-Element gilt als gemeinsamer Vorfahr der mit CSS auswählbaren Elemente.
Elternelemente	Direkte Vorfahren eines anderen Elements, stehen direkt oberhalb des betreffenden Elements.
Nachkommelemente	Alle Elemente, die in der Baumstruktur unterhalb ihres Vorfahrenelements stehen.
Kindelemente	Direkte Nachkommen eines anderen Elements. Sie stehen in der Struktur direkt unterhalb ihres Elternelements.
Geschwisterelemente	Elemente, die in der Baumstruktur auf der gleichen Ebene stehen.
Benachbarte Geschwisterelemente	Elemente, die auf der gleichen Ebene direkt nebeneinanderstehen.

Auf Grundlage dieser Beziehungen definiert CSS die verschiedenen Selektoren, mit denen die Elemente, basierend auf ihrer Position im Dokumentenbaum oder ihrer Beziehung zueinander, ausgewählt werden können.

Wie in den folgenden Rezepten erklärt wird, gibt es auch noch eine Reihe weiterer Selektoren, mit denen Sie Elemente anhand bestimmter Eigenarten der Elemente (Position im Absatz, Attribute usw.) auswählen können.

Siehe auch

Die folgenden Rezepte zu den einzelnen Selektoren; den Abschnitt der CSS 2.1-Spezifikation zu Selektoren unter <http://www.w3.org/TR/CSS21/selector.html>; das CSS3-Selectors-Modul unter <http://www.w3.org/TR/css3-selectors/>; die Tabelle der Quirksmode-Website unter <http://www.quirksmode.org/css/contents.html>, die zeigt, welche Browser welche Selektoren unterstützen; eine deutschsprachige Liste der möglichen CSS-Selektoren unter <http://css-selektoren.mozork.de/>.

1.4 Selektoren gruppieren

Problem

Sie möchten eine Regel für mehrere Elemente auf einmal angeben.

Lösung

Geben Sie die Selektoren in einer durch Kommata getrennten Liste an und definieren Sie dann die gewünschte Regel, wie hier gezeigt:

```
h1, h2, h3, h4, .wichtig {  
    font-family: Helvetica, Arial, sans-serif;  
}
```


Mit dieser Regel wird festgelegt, dass alle Überschriften erster bis vierter Ordnung sowie alle Elemente, die mit der Klasse `wichtig` gekennzeichnet wurden, in einer serifenlosen Schrift dargestellt werden sollen.

Diskussion

Oftmals müssen für verschiedene Elemente die gleichen Einstellungen vorgenommen werden. Damit Sie nicht das Gleiche für jedes Element erneut schreiben müssen, gibt es in CSS die Möglichkeit, Selektoren zu einer Gruppe zusammenzufassen.

Hierbei spielt es keine Rolle, welche Art von Selektoren Sie angeben. Wichtig ist nur, dass die einzelnen Selektoren durch Kommata voneinander getrennt werden. Der Übersicht halber können Sie die einzelnen Selektoren zusätzlich auf einer eigenen Zeile definieren:

```
body > header > h1 a,  
#hauptteil a,  
ul li a {  
    color: green;  
}
```

Beachten Sie, dass am Ende der Liste kein Komma stehen darf. In dem Fall wäre die Regel ungültig und würde ignoriert.

Siehe auch

Das Rezept 1.13 zur Gruppierung von Klassen; die CSS2-Spezifikation zur Gruppierung von Selektoren unter <http://www.w3.org/TR/CSS2/selector.html#grouping>; das *Selectors*-Modul der CSS3-Spezifikation unter <http://www.w3.org/TR/selectors/#grouping>.

1.5 Alle Elemente auf einmal auswählen

Problem

Ihre CSS-Regeln sollen für alle Elemente einer Seite gelten.

Lösung

Verwenden Sie den universellen Selektor (`*`), zum Beispiel:

```
* {  
    margin: 0;  
    padding: 0;  
}
```

Hier sehen Sie eine häufige Verwendung des universellen Selektors. Mit dieser Regel werden die Innen- und Außenabstände aller Elemente auf den Wert 0 zurückgesetzt.

Diskussion

Bei der Gestaltung von Webseiten kommt es gelegentlich vor, dass Sie Regeln definieren müssen, die unabhängig davon funktionieren sollen, um welche Elemente es sich tatsächlich handelt.

Universelle Selektoren werden durch einen Asterisk (*) gekennzeichnet und beziehen sich auf alle Elemente eines Dokuments (siehe Abbildung 1-5).

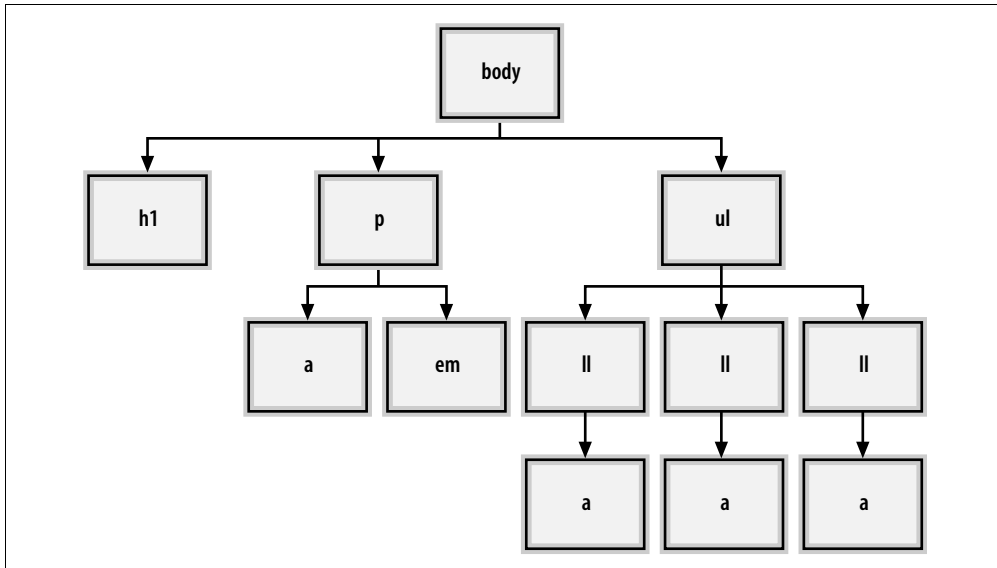


Abbildung 1-5: Die Stildefinitionen des universellen Selektors beziehen sich auf alle Elemente

Sie können den universellen Selektor aber auch mit anderen Selektoren kombinieren, um die Auswahl auf bestimmte Elemente einzuschränken. Das kann zum Beispiel der Fall sein, wenn der Hauptteil Ihrer Seite in einer bestimmten Schriftfarbe erscheinen soll.

```
div#hauptteil * { color: blue; }
```

Diese Regel würde für alle Elemente, die Nachfahren des `div`-Elements mit der ID `hauptteil` sind, eine blaue Schriftfarbe festlegen.

Siehe auch

Die übrigen Rezepte zu den einzelnen Selektoren; der Abschnitt der CSS 2.1-Spezifikation zu Selektoren unter <http://www.w3.org/TR/CSS21/selector.html>; das CSS3-Selectors-Modul unter <http://www.w3.org/TR/css3-selectors/>.

1.6 Bestimmte Elemente nach ihrem Namen auswählen

Problem

Sie wollen CSS-Regeln nur auf bestimmte HTML-Elemente anwenden, etwa nur auf Überschriften erster Ordnung oder nur auf alle Absätze.

Lösung

Verwenden Sie die entsprechenden Elementselektoren, z.B.:

```
h1 {  
    font-size: 120%;  
}  
p {  
    color: blue;  
}
```

Diskussion

Als *Elementselektoren* (auch *Typselektoren* genannt) bezeichnet man CSS-Selektoren, die ein bestimmtes Element oder ein HTML-Tag benennen, auf das die Stildefinitionen angewandt werden sollen. Die erste Regel legt fest, dass Text in h1-Elementen um 20% größer als die im Browser eingestellte Standardschriftgröße angezeigt werden soll. Für Absätze (p-Elemente) wird Blau als Schriftfarbe definiert (siehe Abbildung 1-6).

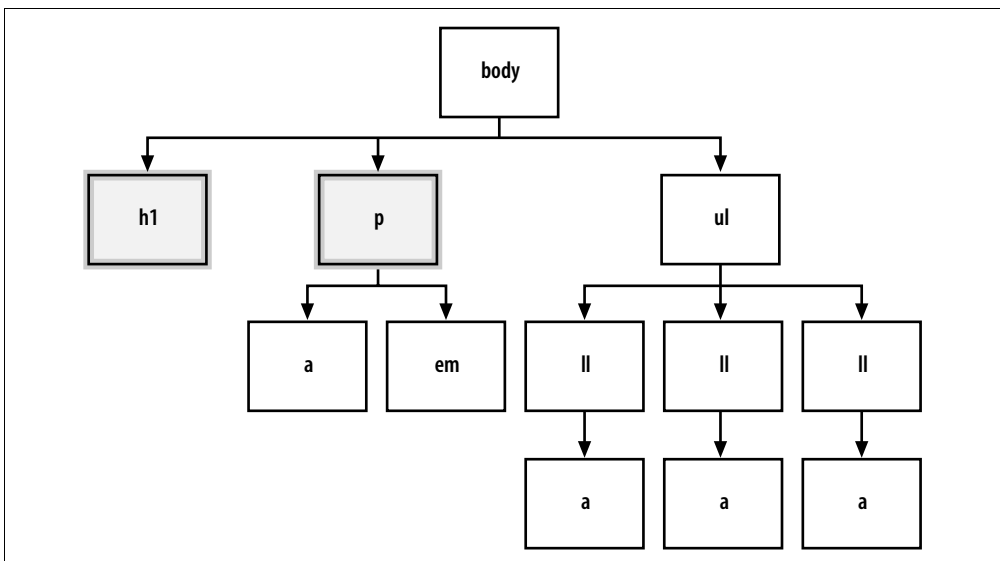


Abbildung 1-6: HTML-Elemente, auf die die CSS-Regeln angewandt werden



Beachten Sie, dass einige Elemente die Eigenschaftswerte ihrer Elternelemente *erben*. So wird nicht nur der Text im Absatz blau eingefärbt, sondern auch das im Absatz enthaltene `em`-Element.

Siehe auch

Die übrigen Rezepte zu den einzelnen Selektoren; den Abschnitt der CSS 2.1-Spezifikation zu Selektoren unter <http://www.w3.org/TR/CSS21/selector.html>; das *CSS3-Selectors*-Modul unter <http://www.w3.org/TR/css3-selectors/>.

1.7 Nachfahrenelemente auswählen

Problem

Sie wollen CSS-Regeln auf HTML-Elemente anwenden, die in der Baumstruktur von anderen Elementen »abstammen«, beispielsweise »alle in einer ungeordneten Liste enthaltenen Links«.

Lösung

Kombinieren Sie die Selektoren, die die gewünschten Elemente beschreiben (zuerst das Vorfahrenelement, dann das Nachfahrenelement), und trennen Sie beide durch ein Leerzeichen, wie hier gezeigt:

```
ul a {  
    text-decoration: none;  
}
```

HTML:

```
<ul>  
<li><a href="#">Ein Link</a></li>  
<li><a href="#">Noch ein Link</a></li>  
<li><a href="#">Ein weiterer Link</a>  
  <ul>  
    <li><a href="#">Ein Link in einer Unterliste</a></li>  
    <li><a href="#">Noch ein Link in einer Unterliste</a></li>  
    <li><a href="#">Ein weiterer Link in einer Unterliste</a>  
  </ul></li>  
</ul>
```

Die oben gezeigte CSS-Regel bezieht sich auf alle Links (a), die sich innerhalb einer ungeordneten Liste (ul) befinden, unabhängig davon, wie tief diese ineinander verschachtelt sind.

Diskussion

Wenn Sie sich das HTML-Dokument (siehe Abbildung 1-6) wie einen Stammbaum vorstellen, sehen Sie, dass sämtliche a-Elemente von einem ul-Element »abstammen«. Anhand von *Nachkommenselektoren* (auch Kontextselektoren genannt) können Sie diese Elemente auswählen. Hierbei müssen die Links nicht direkt auf ul-Elemente folgen. Die Nachkommen müssen also keine Kindelemente sein; die Regel gilt auch für Enkel, Urenkel usw.

Typischerweise bestehen die Nachkommenselektoren aus zwei Elementen. Das erste gibt an, von welchem Element die Nachfahren abstammen sollen, das zweite steht für das Element, auf das die Regel angewandt werden soll. Die Elemente werden durch Leerzeichen voneinander getrennt.

Mit der in der Lösung gezeigten Regel werden also alle a-Elemente ausgewählt, die Nachkommen eines ul-Elements sind. Das sind alle Links innerhalb der Liste, auch die in der Unterliste, wie in Abbildung 1-7 gezeigt.

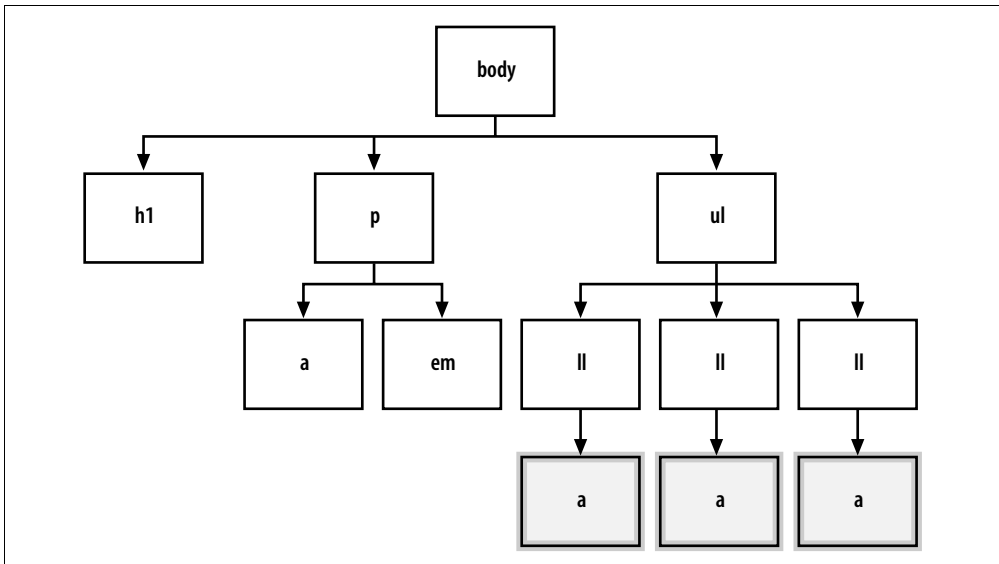


Abbildung 1-7: Alle Links innerhalb der Liste werden vom Nachkommenselektor ausgewählt

Nachkommenselektoren können aber auch aus mehr als zwei Elementen bestehen. So gilt die Regel

```
ul li a {  
    text-decoration: none;  
}
```

nur für Links (a), die sich innerhalb einer verschachtelten Liste befinden. Links auf der obersten Listenebene sind von dieser Regel nicht betroffen.

Nachkommenselektoren besitzen eine höhere *Spezifität* als einfache Elementselektoren oder der universelle Selektor. Das heißt, sie können unter Umständen allgemeiner formulierte Regeln überschreiben.

Siehe auch

Das Rezept 1.20 zur Spezifität von CSS-Regeln; die übrigen Rezepte zur Verwendung von CSS-Selektoren in diesem Kapitel.

1.8 Kindelemente auswählen

Problem

Ihre CSS-Regeln sollen für Elemente gelten, die Kindelemente (direkte Nachfahren) eines anderen Elements sind.

Lösung

Verwenden Sie den Selektor für Kindelemente (**>**). Wie beim Selektor für Nachkommen werden auch hier zwei (oder mehr) Elementselektoren miteinander kombiniert. Um ihre Beziehung zueinander zu kennzeichnen, werden beide durch eine öffnende spitze Klammer voneinander getrennt:

CSS:

```
p > strong {  
    text-decoration: underline;  
}
```

HTML:

```
<div>  
    In diesem Teil des Satzes passiert gar nichts, weil  
    <strong>strong</strong> kein direkter Nachkomme des p-Elements ist.  
  
    <p>Dieses <strong>strong-Element</strong> ist dagegen ein Kindelement des p-Elements  
    und erhält daher die in der CSS-Regel für diesen Fall festgelegten Stildefinitionen.</p>  
</div>
```

Das Ergebnis sehen Sie in Abbildung 1-8

Diskussion

Im Gegensatz zum Nachkommenselektor wählt ein Kindselektor ein Element nur dann aus, wenn es ein *direkter* Nachfahre seines Elternelements ist. Ein Kindselektor wird durch eine schließende spitze Klammer dargestellt, die oftmals zwischen zwei Elementselektoren steht.

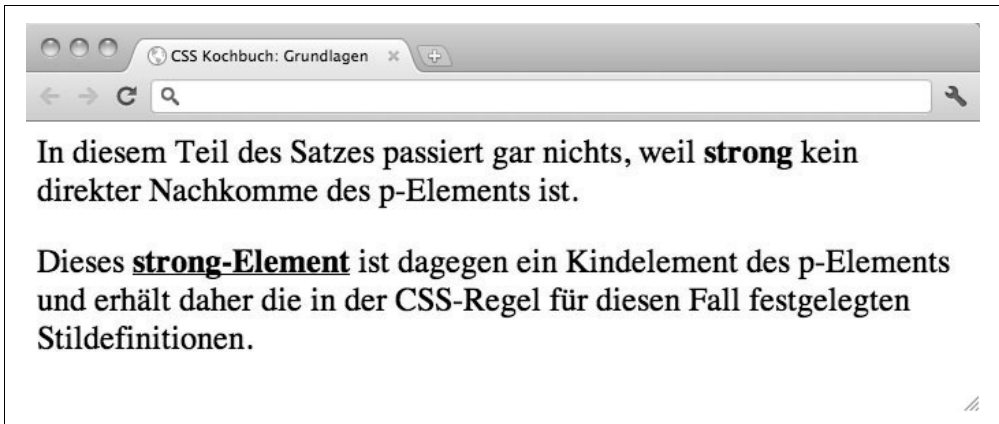


Abbildung 1-8: Die Auswirkungen der Regel für den Kindselektor

Das Diagramm in Abbildung 1-9 zeigt, auf welche Elemente sich die CSS-Regel auswirkt. Das erste strong-Element in den Abbildungen 1-8 und 1-9 wird nicht unterstrichen, weil es sich nicht innerhalb des p-Elements befindet. Damit der Stil angewandt wird, muss eine direkte Eltern-Kind-Beziehung zwischen den Elementen vorliegen. Dies ist ein einfacher, aber sehr wesentlicher Unterschied zwischen dem einfachen Nachkommen- und dem Kindselektor.

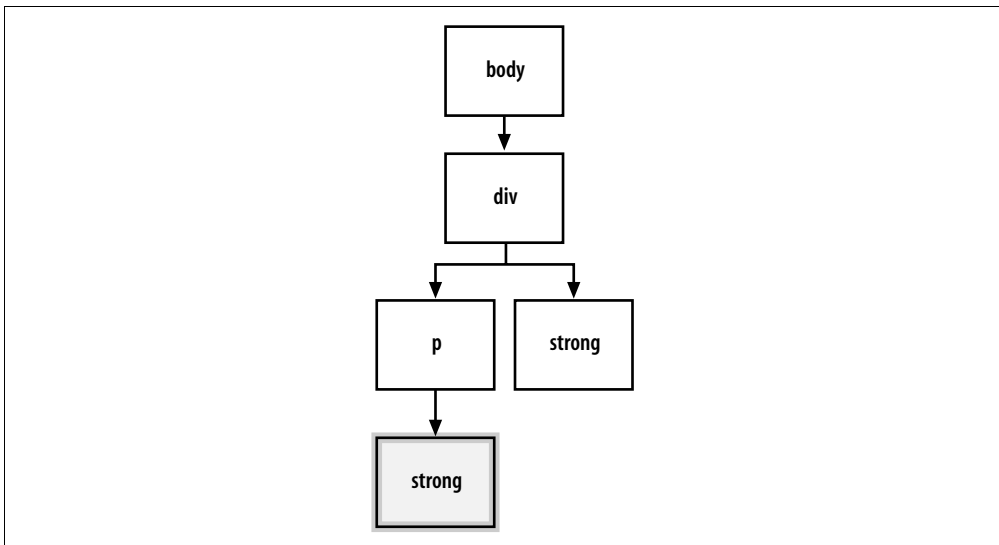


Abbildung 1-9: Das durch den Kindselektor ausgewählte Element innerhalb der Seitenstruktur

Siehe auch

Das Rezept 1.14 zu Pseudoklassen und -elementen in diesem Kapitel; die CSS 2.1-Spezifikation zum Thema Kindselektoren unter <http://www.w3.org/TR/CSS2/selector.html#child-selectors>.

1.9 Benachbarte Geschwisterelemente auswählen

Problem

Sie wollen CSS-Regeln definieren, die nur für ein Element gelten sollen, das ein direkter Nachbar eines anderen Elements ist.

Lösung

Um benachbarte Geschwisterelemente auszuwählen, werden diese mit einem Pluszeichen (+) verbunden, wie hier gezeigt:

CSS:

```
li + li {  
    font-size: 200%;  
}
```

HTML:

```
<ul>  
  <li>Ein Listeneintrag</li>  
  <li>Ein zweiter Listeneintrag</li>  
  <li>Ein dritter Listeneintrag</li>  
</ul>
```

Durch die obige Regel werden der zweite und der dritte Listeneintrag doppelt so groß dargestellt wie der erste.

Diskussion

Zwei Elemente gelten als benachbarte Geschwisterelemente, wenn sie sich im HTML-Quelltext direkt »nebeneinander« befinden.

Die Auswirkungen der Regeln des Selektors für benachbarte Geschwisterelemente sehen Sie in Abbildung 1-10. Hier werden die Regeln nur auf das zweite und dritte Listenelement angewandt, weil dies die einzigen Listenelemente sind, die sich in der Hierarchie direkt neben einem passenden Geschwisterelement (li) befinden. (Um nur das erste li-Element auszuwählen können Sie die Pseudoklasse first-child verwenden, z.B. li:first-child, siehe Rezept 1.14.)



Abbildung 1-10: Der Selektor für benachbarte Geschwister Elemente wirkt sich nur auf das zweite und dritte Listenelement aus, weil diese in der Hierarchie direkt nebeneinander liegen

Das Diagramm in Abbildung 1-11 zeigt, welche Elemente durch die CSS-Regel für benachbarte Geschwister Elemente betroffen sind. Der Selektor wird mittlerweile von allen aktuellen Browsern (IE7 mit geringen Einschränkungen²) unterstützt.

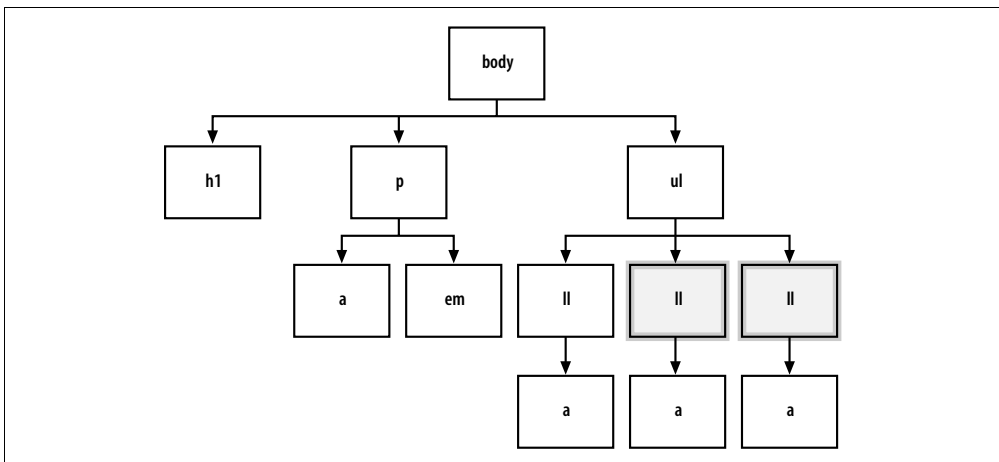


Abbildung 1-11: Die in Abbildung 1-10 ausgewählten Elemente im Diagramm

Der Selektor für benachbarte Geschwister Elemente funktioniert übrigens nicht nur für Elemente des gleichen Typs. Sie können selbstverständlich auch Dinge schreiben wie `p + ul` + `table`:

Der erste Selektor passt beispielsweise auf folgenden HTML-Code:

```

<p>Wann gibt es endlich einen <em>CSS-Kochkurs</em>?</p>
<ul>
  <li>...</li>

```

² IE7 hat die Eigenart, auch HTML-Kommentare als benachbarte Geschwister Elemente zu behandeln. Sie können das Problem vermeiden, indem Sie die Kommentare innerhalb der Geschwister Elemente platzieren.

```

    <li>...</li>
</ul>

<table>
...
</table>

```

Da die Elemente im Selektor auf der gleichen Hierarchieebene liegen, gelten sie als Nachbarn – trotz der verschachtelten Listenelemente.

Der Selektor `span + a + strong` wählt dagegen ein `strong`-Element aus, das im Fließtext neben einem Link steht, der wiederum der Nachbar eines `span`-Elements ist, wie hier gezeigt:

```

<p>Dies ist ein Beispielsatz. <span>Dies ist ein Beispielsatz.</span> Dies ist <a href=
"#">ein Beispielsatz</a>. Dies ist ein Beispielsatz. Dies ist ein <strong>Beispielsatz
</strong>.</p>

```

Die Regel funktioniert, obwohl sich zwischen den Nachbarelementen etwas Text (ein anonymes Inline-Element) befindet. Die Häuser der Nachbarn stehen also nicht Wand an Wand, sondern sind möglicherweise durch Gärten voneinander getrennt.

Siehe auch

Die Quirksmode-Seite zu benachbarten Geschwisterelementen unter http://www.quirks-mode.org/css/selector_adjacent.html; die CSS 2.1-Spezifikation unter <http://www.w3.org/TR/CSS2/selector.html#adjacent-selectors>.

1.10 Regeln für allgemeine Geschwisterelemente definieren

Problem

Sie wollen CSS-Regeln für ein Geschwisterelement definieren, das kein direkter Nachbar ist (siehe voriges Rezept).

Lösung

Verwenden Sie den CSS3-Selektor für allgemeine Geschwisterelemente (`~`), wie hier gezeigt:

CSS:

```

ul ~ div {
    border: 1px solid black;
}

```

HTML:

```

<div>
  <p>Ein Absatz</p>
  <ul>

```

```

    <li>Eine</li>
    <li>Liste</li>
  </ul>
  <p>Noch ein Absatz</p>

  <div><!-- Dieses div-Element wird ausgewählt. -->
    <p>Ein weiterer Absatz.</p>
  <div>Dieses div-Element wird nicht ausgewählt.</div>
</div>

</div>

```

Damit das `div`-Element mit einem schwarzen Rahmen versehen wird, muss eines der vorangehenden Geschwisterelemente eine ungeordnete Liste sein.

Diskussion

Mit dem in CSS3 neu hinzugekommenen Selektor für allgemeine Geschwisterelemente können Sie ein Element auswählen, wenn irgendeines seiner Geschwister bestimmte Bedingungen erfüllt.

Manchmal ist es nicht möglich, Klassen- oder ID-Selektoren zu verwenden, um bestimmte Elemente zu kennzeichnen und im Stylesheet auszuwählen (beispielsweise wenn Sie keinen Zugriff auf die HTML-Dateien oder Templates haben). In diesem Fall müssen Sie auf andere Selektoren ausweichen. Der Selektor für allgemeine Geschwisterelemente füllt hier eine wichtige Lücke und ermöglicht die Auswahl von Elementen, die vorher nur über recht komplexe Regeln oder zusätzlichen HTML-Code zugänglich waren.

Dieser Selektor funktioniert in allen aktuellen Browsern einschließlich Internet Explorer 7.

Siehe auch

Das *CSS3-Selectors*-Modul unter <http://www.w3.org/TR/css3-selectors/#general-sibling-combinators>; das Rezept 1.12 zur Verwendung von Klassen- bzw. ID-Selektoren.

1.11 Elemente nach ihren Attributen auswählen

Problem

Sie wollen CSS-Stile abhängig vom Attribut oder Attributwert (z.B. `href` bei Links oder `type` bei Formularelementen) eines Elements definieren.

Lösung

Verwenden Sie einen Attributselektor, zum Beispiel:

```

button[type="submit"] {
  color: red;
}

```

Hiermit wird beispielsweise festgelegt, dass Submit-Buttons rot beschriftet werden sollen. Die verschiedenen Möglichkeiten, diesen Selektor zu nutzen, finden Sie in der Diskussion.

Diskussion

Attributselektoren bieten eine Fülle von Möglichkeiten, ein Element anzusprechen, das ein passendes Attribut enthält. Im Folgenden sehen Sie Beispiele für die verschiedenen Optionen:

[Attribut] – Sucht basierend auf dem Namen des Attributs nach Treffern.

```
a[title] {  
    text-decoration: none;  
}
```

Immer wenn das HTML-Element `a` das Attribut `title` enthält, wird der Link nicht unterstrichen.

[Attribut=Wert] – Sucht basierend auf dem Attributwert nach Treffern.

```
a[href="http://www.csskochbuch.com"] {  
    text-decoration: none;  
}
```

Immer wenn ein Link auf `http://www.csskochbuch.com` im HTML-Code erscheint, wird dieser ohne Unterstreichung dargestellt.

[Attribut~=Wert] – Mithilfe dieses Selektors kann ein beliebiges Element anhand eines Teils der durch Leerzeichen getrennten Attributwerte ausgewählt werden.

```
a[title~="digital"] {  
    text-decoration: none;  
}
```

Immer wenn das Attribut `title` eines `a`-Elements die Zeichenkette `"digital"` enthält, wird der Link ohne Unterstreichung dargestellt. (Dies entspricht übrigens der Funktionsweise von Klassenselektoren. Sie können ein Element, das im HTML-Code per `class='wichtig'` gekennzeichnet wurde, also einerseits mit `.wichtig`, aber auch mit `[class~='wichtig']` auswählen.)

[Attribut|=Wert] – Findet Treffer, wenn der Attributwert mit »Wert« beginnt und darauf ein Bindestrich folgt.

```
*[lang|= "de"] {  
    color: blue;  
}
```

Dieser Selektor findet alle HTML-Elemente, die ein `lang`-Attribut besitzen. Der Wert des Attributs muss mit `de` beginnen, worauf direkt ein Bindestrich folgt. Das obige Beispiel sorgt dafür, dass alle Elemente, deren `lang`-Attribut beispielsweise den Wert `de-DE` haben, blau dargestellt werden.

Neue Attributselektoren in CSS3

Die CSS3-Spezifikation definiert eine Reihe neuer Attributselektoren, mit denen Elemente ausgewählt werden können, wenn ihre Attributwerte beispielsweise mit einer bestimmten Zeichenkette beginnen oder enden:

[Attribut^=Wert] – Findet Treffer, wenn der Attributwert mit »Wert« beginnt.

```
a[href^="http://"] {  
    text-decoration: none;  
}
```

Hier werden Links, deren href-Attribut mit der Zeichenkette http:// beginnt, nicht unterstrichen dargestellt. Auf diese Weise lassen sich beispielsweise externe und interne Links unterschiedlich darstellen.

[Attribut\$=Wert] – Findet Treffer, wenn der Attributwert mit »Wert« endet.

```
a[href$=".gif"] {  
    color: green;  
}
```

In diesem Fall werden Links, deren href-Attribut mit der Zeichenkette .gif endet, grün dargestellt. Ähnlich wie im vorigen Beispiel können Sie auf diese Weise Links auf verschiedene Dateien unterschiedlich darstellen.

[Attribut=Wert] – Findet Treffer, wenn der Attributwert die Zeichenkette »Wert« enthält.*

```
a[href*=".com"] {  
    text-decoration: none;  
}
```

Dieses Beispiel sorgt dafür, dass Links, deren href-Attribut die Zeichenkette .com enthält, nicht unterstrichen dargestellt werden.

Attributselektoren gruppieren

Die CSS-Spezifikationen (2.1 und 3) erlauben die Gruppierung von Attributselektoren. Mit folgendem Selektor können Sie beispielsweise ein a-Element auswählen, wenn dessen href-Attribut mit http beginnt und das title-Attribut das Wort »Knüll« enthält:

```
a[href^='http:'][title*="Knüll"] {  
    background-color: #efefef;  
    border: 1px solid green;  
}
```

Auf diese Weise können beispielsweise Links, die auf das SommerMusikFest verweisen, wie `<a href="http://www.sommermusikfest.de" title="Das SommerMusikFest auf dem Knüllköpfchen">SMF!!</a>`, speziell hervorgehoben werden. Hierbei spielt die Reihenfolge der Attributselektoren keine Rolle.



Es ist sogar möglich, mehrere Selektoren auf das gleiche Attribut anzuwenden, wie beispielsweise `[title*="Knüll"][title*="köpfchen"]`. Dieser Selektor findet beliebige Elemente, deren title-Attribute den Substring »Knüll« und den Substring »köpfchen« enthalten.

Siehe auch

Eine Tabelle der von den verschiedenen Browsern unterstützten Selektoren unter <http://www.quirksmode.org/css/contents.html> und <http://kimblim.dk/css-tests/selectors/>; eine Liste der vom Internet Explorer unterstützten CSS-Selektoren unter [http://msdn.microsoft.com/de-de/library/cc351024\(v=vs.85\).aspx](http://msdn.microsoft.com/de-de/library/cc351024(v=vs.85).aspx); das *Selectors*-Modul der CSS3-Spezifikation zur Gruppierung von Attributselektoren unter <http://www.w3.org/TR/selectors/#attribute-representation>.

1.12 Elemente aufgrund von Klassen- und ID-Attributen auswählen

Problem

Bestimmte Teile eines HTML-Dokuments wurden zur besseren Strukturierung mit class- oder id-Attributen gekennzeichnet. Diese sollen, unabhängig von den verwendeten HTML-Elementen, mit eigenen Stilen, beispielsweise abgerundeten Ecken oder einer bestimmten Breite, versehen werden.

Lösung

Verwenden Sie je nach Zweck Klassen- oder ID-Selektoren, z.B.:

```
.abgerundete_ecken {  
    border-radius: 15px;  
}  
  
#hauptteil {  
    width: 900px;  
}
```

Der obere Selektor sorgt dafür, dass die Deklaration auf alle Elemente angewandt wird, deren class-Attribut den Wert `.runde_ecken` hat. Der zweite Selektor bezieht sich nur auf das eine Element, dessen id-Attribut den Wert `hauptteil` besitzt.

Diskussion

Klassen- und ID-Selektoren schaffen eine Möglichkeit, bestimmte Elemente, Gruppen von Elementen oder auch bestimmte Textteile mit eigenen Stilen zu versehen. Oftmals kommen hierfür die HTML-Elemente `div` und `span` zur Anwendung, die mit einem passenden class- oder id-Attribut versehen werden. Natürlich ist es auch möglich, andere HTML-Elemente so zu kennzeichnen.

Klassen werden mithilfe eines class-Attributs im HTML-Quelltext definiert und dürfen im Dokument beliebig oft vorkommen:

```
<div class='abgerundete_ecken'>...</div>
```

IDs werden anhand eines id-Attributs vergeben. Sie dürfen pro Dokument nur einmal vorkommen:

```
<ul id='navigation'>...</ul>
```

Daher werden IDs in CSS hauptsächlich dazu eingesetzt, bestimmte Bereiche der Seite (Kopfteil, Navigation, Seitenleiste usw.) mit eigenen Stilen zu versehen. (Im HTML-Code werden id-Attribute außerdem häufig für die Definition von Sprungmarken oder für die Arbeit mit JavaScript, z.B. in jQuery, verwendet.)

Der Name einer Klasse oder ID sollte immer mit einem Buchstaben beginnen.³ Danach ist eine beliebige Kombination aus Buchstaben (A–Z, a–z), Zahlen (0–9), Bindestrichen (-) und Unterstrichen (__) erlaubt.

Klassenselektoren

Wenn Sie die gleichen CSS-Regeln häufig auf verschiedene Elemente anwenden wollen, empfiehlt sich die Verwendung eines *Klassenselektors*. Das könnte zum Beispiel der Fall sein, wenn bestimmte Elemente auf der rechten Seite von Text umfließen werden sollen und andere auf der linken. In diesem Fall könnten die folgenden Regeln zur Anwendung kommen:

```
.rechts_umfliessen {  
    float: left;  
}  
  
.links_umfliessen {  
    float: right;  
}
```

Wie Sie sehen, werden Klassenselektoren in CSS mit einem vorangestellten Punkt (.) gekennzeichnet. Im HTML-Code müssten Sie die gewünschten Elemente außerdem mit einem class-Attribut kennzeichnen:

```
<img class="links_umfliessen" href="bild.jpg" alt="Bild" />
```

oder auch:

```
<div class="rechts_umfliessen">...</div>
```

Wenn Sie dem Klassenselektor keine weiteren Selektoren voranstellen, gilt er für alle Elemente, die mit dieser Klasse markiert wurden. Daher können wir die Regeln im obigen Beispiel sowohl auf Bilder wie auch auf div-Elemente anwenden.

Soll die Stildefinition dagegen nur für bestimmte Elemente dieser Klasse gelten, schreiben Sie den Namen des Elements einfach vor den Punkt, wie hier:

```
div.runde_ecken {  
    border-radius: 15px;  
}
```

³ Offiziell sind auch Bindestriche und Unterstriche erlaubt. Diese sind allerdings für die Definition von herstellerspezifischen Erweiterungen reserviert. Um Konflikte zu vermeiden, sollten Sie den Namen einer Klasse oder ID weder mit - noch mit _ beginnen.

Wie die Verwendung des Klassenselektors in der Praxis aussieht, sehen Sie in Abbildung 1-12.



Abbildung 1-12: Die modifizierte CSS-Regel in der Webseite

ID-Selektoren

Anders verhält es sich, wenn Sie nur ganz bestimmte Elemente mit Stilen versehen wollen. Dies könnte der Navigationsbereich einer Seite sein, eine Seitenleiste oder auch die Seite selbst. In diesem Fall ist es sinnvoll, die entsprechenden HTML-Elemente mit id-Attributen zu versehen, auf die Sie dann mit dem entsprechenden Selektor in Ihrem Style-sheet zugreifen können:

```
div#navigation {
    border: 1px solid #000000;
    background-color: rgba(200,200,200,0.2);
}
```

Die oben stehende Regel erzeugt um das div-Element mit der ID navigation einen schwarzen Rahmen und versieht es mit einem hellgrauen, durchscheinenden Hintergrund. Der entsprechende HTML-Code könnte beispielsweise so aussehen:

```
<div id="navigation">
  <a href="#hauptteil">Navigation überspringen ...</a>
```



```

<ul>
<li><a href="/index.html">Startseite</a></li>
<li><a href="/produkte.html">Produkte</a></li>
<li><a href="/ueber_uns.html">Über uns</a></li>
<li><a href="/impressum.html">Impressum</a></li>
</ul>
</div>

```

In diesem Beispiel haben wir eine typische Liste aus Navigationslinks mit einem `div`-Element umgeben, das treffenderweise das `id`-Attribut `navigation` erhalten hat. Um innerhalb des Stylesheets darauf zugreifen zu können, verwenden Sie den Namen des Attributs, dem ein Doppelkreuz vorangestellt wird (`#navigation`).

Auch ID-Selektoren können Sie andere Selektoren voranstellen, wie im obigen Beispiel gezeigt. Dies ist bei Elementen, die sowieso nur einmal im Dokument vorkommen dürfen, allerdings nur begrenzt sinnvoll.⁴



Versuchen Sie, bei der Benennung von Klassen und IDs Namen zu vergeben, die die Funktion des oder der Elemente beschreiben und nicht ihr Aussehen. Sobald sich das Layout ändert, steht die `linke_seitenleiste` vielleicht auf der rechten Seite, und die `rote_schrift` ist plötzlich grün.

Auch Namen wie `stl23_a_grd` sind nicht besonders aussagekräftig. Sofern Sie nicht akribisch über den tieferen Sinn dieser Benennung Buch führen, haben Sie spätestens zwei Wochen nach dem Schreiben dieses Codes ein Problem.

Klassen- oder ID-Attribute?

Verwenden Sie `class`-Attribute, wenn die gleiche Stildefinition mehrere Male in einem Dokument benutzt werden soll. `id`-Attribute kommen zum Einsatz, wenn ein Stil nur auf ein einziges Element in einem Dokument angewandt werden soll.

Siehe auch

Die CSS 2.1-Spezifikation zum Thema ID-Selektoren unter <http://www.w3.org/TR/CSS21/selector.html#id-selectors>; die CSS 2.1-Spezifikation zum Thema Klassenselektoren unter <http://www.w3.org/TR/CSS21/selector.html#class-html>.

⁴ Dieser Sinn kann darin bestehen, Ihr Stylesheet leichter lesbar zu machen, oder darin, die Spezifität für einen bestimmten Selektor zu erhöhen. Kommt es zwischen verschiedenen Stildefinitionen zum Konflikt, hat `div#navigation` Vorrang vor `#navigation`.

1.13 Mehrere Klassen auf einmal angeben

Problem

Sie möchten mehrere Klassen auf einmal für das gleiche Element benutzen.

Lösung

Geben Sie im `class`-Attribut des Elements die gewünschten Klassennamen in einer durch Leerzeichen getrennten Liste an, z.B.:

```
<div class="hinweis wichtig">...</div>
```

Im Stylesheet definieren Sie die beiden Klassen wie gewohnt:

```
.hinweis {  
    font-weight: bold;  
    color:      #68000;  
}  
.wichtig {  
    border: 2px dotted red;  
}
```

In der Browserdarstellung erscheint das `div`-Element nun in dunkelroter fetter Schrift und wird zudem von einem gepunkteten roten Rahmen umgeben.

Diskussion

Gelegentlich ist es sinnvoll, Stildefinitionen auf mehrere Regeln zu verteilen, beispielsweise wenn die Definitionen auch unabhängig benutzt werden sollen. Die im HTML-Code gezeigte Schreibweise gibt Ihnen diese Möglichkeit.

Siehe auch

Die HTML-Spezifikationen zur Verwendung des `class`-Attributs unter <http://www.w3.org/TR/html401/struct/global.html#h-7.5.2> bzw. <http://www.w3.org/TR/html5/elements.html#classes>.

1.14 Regeln für unbenannte Dokumentteile definieren

Problem

Sie wollen Stile für Teile des HTML-Dokuments definieren, die ohne zusätzliches Markup nicht zugänglich wären oder die sich durch Interaktion mit Benutzern ändern können. Wie in der Lösung gezeigt wird, kann das beispielsweise die erste Zeile eines Absatzes sein oder auch ein Link, über dem sich gerade der Mauszeiger befindet.

Lösung

Verwenden Sie *Pseudoelemente* und *Pseudoklassen*. Angenommen, Sie möchten die erste Zeile eines Absatzes mit einem eigenen Stil versehen. Die Länge der Zeile kann sich abhängig von der Breite des Browserfensters (oder des umgebenden Elements) ändern. In diesem Fall können Sie das Pseudoelement `:first-line` verwenden:

```
p:first-line {  
    font-size: 120%;  
}
```

Unabhängig von ihrer Länge erhält die erste Zeile eines Absatzes hier eine eigene Schriftgröße.

Häufig sollen Elemente hervorgehoben werden, wenn sich der Mauszeiger darüber befindet. Genau für diesen Fall gibt es die dynamische Pseudoklasse `:hover`:

```
a:hover {  
    color: green;  
    background-color: red;  
}
```

Durch diese Regel werden Links, über denen sich der Mauszeiger befindet, in grüner Schrift vor rotem Hintergrund dargestellt.

Diskussion

Manche Dinge eines HTML-Dokuments sind durch den Dokumentenbaum nur schwer oder gar nicht zugänglich und lassen sich durch andere Selektoren nicht ausdrücken. Das kann der erste Buchstabe oder die erste Zeile eines Abschnitts sein, aber auch die verschiedenen Zustände, die ein Link einnehmen kann, bevor, während und nachdem ein Benutzer ihn anklickt. Außerdem ist es nicht ohne Weiteres möglich, beispielsweise das erste Kindelement eines anderen Elements mit Stilen zu versehen. Für alle diese Fälle gibt es Pseudoklassen und -elemente.

CSS 2.1-Pseudoklassen

Pseudoklassen ermöglichen die Auswahl von Elementen, wenn diese zum Beispiel einen bestimmten Zustand haben, der sich in Interaktion mit dem Benutzer ändern kann. Dies kann ein Link sein, über dem sich gerade der Mauszeiger befindet oder der bereits besucht wurde, aber auch ein Formularelement, das aktuell den Fokus hat. Ausnahmen sind die Pseudoklassen `:first-child` und `:lang`. Erstere wählt ein Element aus, wenn dieses das erste Kindelement eines anderen Elements ist, die zweite passt auf Elemente, deren `lang`-Attribut einen bestimmten Wert hat.

Pseudoklassen werden dadurch gekennzeichnet, dass man ihrem Namen einen Doppelpunkt (:) voranstellt.

In Tabelle 1-2 sehen Sie eine Übersicht über die in der CSS 2.1-Spezifikation definierten Pseudoklassen.

Tabelle 1-2: In der CSS 2.1-Spezifikation definierte Pseudoklassen

Name der Pseudoklasse	Funktion
:link	Kennzeichnet einen noch nicht besuchten Link.
:visited	Kennzeichnet einen bereits besuchten Link.
:hover	Kennzeichnet ein Element, über dem sich gerade der Mauszeiger befindet.
:active	Kennzeichnet einen Link, der gerade angeklickt wird.
:focus	Kennzeichnet ein Element (meist ein Formularelement), das gerade den Fokus hat, z.B. während einer Eingabe in ein Textfeld.
:first-child	Das erste Kindelement eines anderen Elements.
:lang(Sprache)	Steht für ein Element, dessen lang-Attribut den Wert Sprache (z.B. de) hat.

CSS 2.1-Pseudoelemente

Während sich Pseudoklassen auf einen bestimmten *Zustand* eines Elements beziehen, verweisen Pseudoelemente auf einen bestimmten *Teil* eines Elements. Dies kann beispielsweise die erste Zeile eines Absatzes sein oder auch der erste Buchstabe einer Textzeile.

Tabelle 1-3 enthält eine Liste der in der CSS 2.1-Spezifikation definierten Pseudoelemente.

Tabelle 1-3: In der CSS 2.1-Spezifikation definierte Pseudoelemente

Name des Pseudoelements	Funktion
:first-line	Wählt die erste Textzeile eines Elements unabhängig von der Breite des Elements oder des Ansichtsbereichs aus.
:first-letter	Trifft auf den ersten Buchstaben eines Elements zu.
:before	Kann verwendet werden, um mittels der Eigenschaft <code>content</code> Inhalte am Anfang eines Elements einzufügen, z.B. Anführungszeichen vor einem Zitat.
:after	Kann verwendet werden, um mittels der Eigenschaft <code>content</code> Inhalte am Ende eines Element einzufügen, z.B. Anführungszeichen nach einem Zitat.

Die durch die Pseudoelemente `:before` und `:after` eingefügten Inhalte werden innerhalb des Elements platziert, für das sie definiert wurden. Aus diesem Grund funktionieren `:before` und `:after` auch nicht mit ersetzten Elementen wie beispielsweise Bildern oder Formularelementen.

Neue Pseudoklassen in CSS3

Das *CSS-Selectors*-Modul definiert eine ganze Reihe neuer Pseudoklassen. Durch ihren Einsatz kann in vielen Fällen auf die Verwendung spezieller `class`- und `id`-Attribute im HTML-Quelltext verzichtet werden. Ein paar Beispiele:

Das letzte Element einer Liste: Bei der Erstellung von Navigationsmenüs ist es oftmals nötig, dem letzten Listenelement eigene Stile zuzuweisen. Bisher musste das Element hierfür

mit einem eigenen class- oder id-Attribut versehen werden. Durch die Verwendung der Pseudoklasse `:last-child` ist das nicht mehr nötig, wie hier gezeigt:

```
ul#navigation li:last-child {  
    border-bottom: 1px solid black;  
}
```

Jedes zweite Element: Um beispielsweise den Zeilen einer Tabelle abwechselnd unterschiedliche Hintergrundfarben zu geben, war es bisher nötig, jede zweite Zeile mit einem class-Attribut zu versehen. Diese zusätzliche Kennzeichnung entfällt, wenn Sie die neue Pseudoklasse `:nth-child()` verwenden. Um jeder zweiten Tabellenzeile einen grauen Hintergrund zu verpassen, reicht es, wenn Sie schreiben:

```
tr:nth-child(even) {  
    background-color: rgb(200,200,200);  
}
```

Das Schlüsselwort `even` sorgt dafür, dass die Regel nur auf jedes zweite (geradezahlige) `tr`-Element angewandt wird. Neben Schlüsselwörtern sind für `:nth-child` noch deutlich komplexere Angaben möglich.

In Tabelle 1-4 finden Sie eine Übersicht über die neuen Pseudoklassen in CSS3.

Tabelle 1-4: In der CSS3-Spezifikation definierte Pseudoklassen

Name der Pseudoklasse	Funktion
<code>:root</code>	Trifft auf das Wurzelement des Dokuments zu. In HTML-Dokumenten ist dies das Element <code>html</code> .
<code>:last-child</code>	Kennzeichnet ein Element, das das letzte Kindelement eines anderen Elements ist, z.B. den letzten Eintrag in einer Liste.
<code>:nth-child(n)</code>	Wählt eines oder mehrere Kindelemente eines anderen Elements aus. Die Kindelemente werden von vorne nach hinten durchgezählt. Anhand eines numerischen Werts, eines Schlüsselworts oder einer Formel innerhalb der runden Klammern werden bestimmte Elemente ausgewählt.
<code>:nth-last-child(n)</code>	Funktioniert wie <code>:nth-child()</code> . Allerdings werden die Kindelemente hier von hinten nach vorne durchgezählt.
<code>:nth-of-type(n)</code>	Funktioniert wie <code>:nth-child()</code> , bezieht sich jedoch auf Kindelemente eines bestimmten Typs, z.B. <code>li</code> oder <code>td</code> .
<code>:nth-last-of-type(n)</code>	Funktioniert wie <code>:nth-of-type()</code> . Allerdings werden die Elemente hierbei von hinten nach vorne durchgezählt.
<code>:first-of-type</code>	Bezieht sich auf das erste Element eines bestimmten Typs.
<code>:last-of-type</code>	Bezieht sich auf das letzte Element eines bestimmten Typs.
<code>:only-child</code>	Trifft nur zu, wenn das Element das einzige Kindelement eines anderen Elements ist.
<code>:only-of-type</code>	Trifft nur zu, wenn es keine weiteren Kindelemente des gleichen Typs gibt.
<code>:empty</code>	Bezieht sich auf ein Element, das keinerlei Nachkommenelemente hat («anonymer» Text eingeschlossen).
<code>:target</code>	Kennzeichnet ein Element, das durch einen sogenannten Fragmentbezeichner (fragment identifier) als Ziel eines Links angegeben wurde.

Tabelle 1-4: In der CSS3-Spezifikation definierte Pseudoklassen (Fortsetzung)

Name der Pseudoklasse	Funktion
:enabled	Steht für ein Formularelement oder einen anderen Bestandteil einer Benutzerschnittstelle, das nicht mit dem Attribut <code>disabled</code> gekennzeichnet wurde.
:disabled	Steht für ein Formularelement oder einen anderen Bestandteil einer Benutzerschnittstelle, das direkt oder indirekt mit dem Attribut <code>disabled</code> gekennzeichnet wurde.
:checked	Steht für ein Formularelement (z.B. ein Ankreuzfeld), das aktiviert wurde.
:not()	Steht für eine Verneinung. Um beispielsweise allen Buttons einen bestimmten Stil zuzuweisen, die nicht deaktiviert sind, könnten Sie schreiben: <code>button:not([disabled])</code> .

Wie Sie sehen, ist die Zahl der Pseudoklassen in CSS3 deutlich angestiegen. Wenn Sie diese Selektoren richtig einsetzen, entfällt in vielen Fällen die Notwendigkeit, mit speziellen Klassen oder IDs zu arbeiten. Dadurch können Inhalt und Gestaltung noch besser voneinander getrennt werden, weil bei einer Layoutänderung die Anpassungen im HTML-Code weitgehend wegfallen.

Im Gegensatz zu den Pseudoklassen werden im *Selectors*-Modul von CSS3 keine neuen Pseudoelemente definiert. Allerdings hat sich ihre Schreibweise geringfügig geändert.

Damit sich Pseudoelemente leichter von Pseudoklassen unterscheiden lassen, sollen Erstere laut der CSS3-Spezifikation mit einem doppelten Doppelpunkt (::) gekennzeichnet werden. Dies ist allerdings nur für neu hinzugekommene Pseudoelemente wirklich verpflichtend. (Da ältere Browser diese Schreibweise nicht verstehen, verwenden wir in diesem Buch die Schreibweise mit einem Doppelpunkt.)

Internet Explorer und CSS3-Selektoren

Die hier gezeigten CSS3-Selektoren und Pseudoklassen werden im Internet Explorer erst ab Version 9 ohne Hilfsmittel unterstützt. Mithilfe des Polyfills *selectivizr* können Sie auch IE7 und 8 zum Mitmachen bewegen. Mehr zu diesem Werkzeug finden Sie in Kapitel 13, *Tipps und Tricks*.

Siehe auch

Die Rezepte zur Gestaltung von Links in Kapitel 6, *Links und Navigation*; zur ersten Absatzzeile Rezept 3.20 und zum ersten Buchstaben eines Texts Rezept 3.18; das Rezept 6.5 mit einem Anwendungsbeispiel für die CSS3-Pseudoklassen `:target` und `:not`; die CSS-Spezifikationen zum Thema Pseudoklassen und -elemente unter <http://www.w3.org/TR/CSS2/selector.html#pseudo-elements> (CSS 2.1) bzw. <http://www.w3.org/TR/css3-selectors/#pseudo-classes> und <http://www.w3.org/TR/css3-selectors/#pseudo-elements> (CSS3); die HTML5-Spezifikation zum Deaktivieren bestimmter Formularelemente mit dem Attribut `disabled` unter <http://www.w3.org/TR/html5/association-of-controls-and-forms.html#enabling-and-disabling-form-controls>; die Website der JavaScript-Bibliothek *selectivizr* unter <http://selectivizr.com/>.

1.15 Werte für die neue Pseudoklasse :nth-child() und ihre Verwandten richtig angeben

Problem

Sie möchten verstehen, wie man die Werte für Pseudoklassen wie :nth-child() oder :nth-of-type() richtig angibt.

Lösung

Verwenden Sie die runden Klammern, die dem Namen der Pseudoklasse nachgestellt werden, um mehr oder weniger komplexe Angaben darüber zu machen, welche Elemente ausgewählt oder übersprungen werden sollen, wie in Abbildung 1-13 gezeigt.

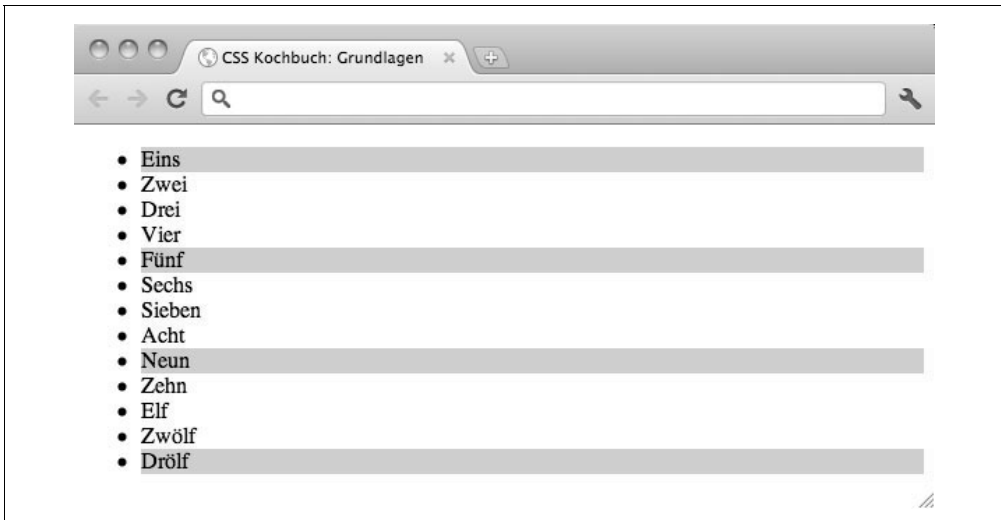


Abbildung 1-13: Die Auswirkungen des Selektors `li:nth-child(4n+1)` auf die Gestaltung einer ungeordneten Liste

Die einfachste Form eines Parameters für :nth-child() könnte so aussehen:

```
li:nth-child(5) {  
    background-color: rgb(230,230,0);  
}
```

Diese einfache Regel versieht das fünfte Listenelement mit einem hellgrünen Hintergrund. Wollen Sie dagegen allen Elementen nach dem fünften Element den gleichen Stil zuweisen, können Sie schreiben:

```
li:nth-child(n+6) {  
    background-color: rgb(230,230,0);  
}
```

Jetzt wird dem sechsten und allen folgenden Elementen der gleiche Stil zugewiesen. Diese Regel lässt sich übrigens auch umkehren:

```
li:nth-child(-n+5) {  
    background-color: rgb(230,230,0);  
}
```

Hier erhalten nur die ersten fünf Elemente der Liste den gleichen Stil. Um nur jedem vierten Element, beginnend mit dem ersten Element der Liste, die gleichen Definitionen zuzuweisen, können Sie beispielsweise die folgende Regel schreiben:

```
li:nth-child(4n+1) {  
    background-color: rgb(230,230,0);  
}
```

Nach dieser Regel erhalten das erste, das fünfte, das neunte sowie das dreizehnte Element und so weiter jeweils die gleiche Gestaltung.

Eines der häufigsten Anwendungsgebiete für `nth-child()` wird aber wohl das abwechselnde Gestalten von aufeinanderfolgenden Elementen gleichen Typs sein. Das können wie hier Listenelemente oder auch Tabellenzeilen und andere Dinge sein. Daher definiert das *Selectors*-Modul der CSS3-Spezifikation für diese Fälle die Schlüsselwörter `odd` und `even`. Um beispielsweise alle ungerade Listenelemente (1, 3, 5 usw.) auszuwählen, reicht es, wenn Sie schreiben:

```
li:nth-child(odd) {  
    background-color: rgb(230,230,0);  
}
```

Jetzt werden alle ungeraden Listenelemente mit dem hellgrünen Hintergrund versehen. Um stattdessen die Elemente 2, 4, 6 usw. zu gestalten, tauschen Sie `odd` einfach gegen das Schlüsselwort `even` aus.

Diskussion

Mit der neuen Pseudoklasse `:nth-child()` und ihren Verwandten `:nth-last-child()`, `:nth-of-type()` und `:nth-last-of-type()` wurden die CSS-Selektoren um einige sehr mächtige Werkzeuge ergänzt.

Diese Pseudoklassen erweitern und verfeinern die einfachen Element- bzw. Typselektoren um eine Reihe »magischer« Fähigkeiten. Um zu verstehen, was diese Pseudoklassen machen, sollten Sie sich noch einmal mit der Pseudoklasse `:first-child` beschäftigen. Sie wählt ein Element aus, wenn es das erste Kindelement eines anderen Elements ist, beispielsweise das erste Listenelement oder die erste Tabellenzeile.

Mit der Pseudoklasse `:nth-child()` ist es nun möglich, nicht nur das erste, sondern ein oder mehrere beliebige Elemente anhand ihrer Position anzusprechen. Mit den neuen Pseudoklassen können Sie erstmals Dinge wie »jedes zweite Kindelement« oder »die ersten fünf Elemente eines bestimmten Typs« auf einfache Weise ausdrücken, ohne hierfür auf die Verwendung von `class`-Attributen im HTML-Code zurückgreifen zu müssen.

Allerdings hat diese Vielseitigkeit auch ihren Preis: Um genau anzugeben, welche Elemente einer Liste, Tabellenzeilen o.Ä. gestaltet werden sollen, müssen Sie verstehen, wie die Werte für `:nth-child()` und seine Verwandten richtig formuliert werden.

`:nth-child()` oder `:nth-of-type()`?

Während bei `:nth-child()` die aufeinanderfolgenden Elemente vom gleichen Typ sein müssen, können sich bei `:nth-of-type()` auch Elemente eines anderen Typs dazwischen befinden.

Angenommen, Sie verwenden die folgende Regel, um nur den zweiten Absatz im Hauptteil einer Seite mit besonderen Stilen zu versehen:

```
p:nth-child(2) { color: red; }
```

Das funktioniert, solange der HTML-Code aussieht wie hier:

```
<div id="hauptteil">
  <p>Ein Absatz</p>
  <p>Dieser Absatz hat Stil!</p>
  <p>Noch ein Absatz</p>
  <p>et cetera ad nauseam</p>
</div>
```

Sobald aber vor dem zweiten Absatz weitere Elemente eingefügt werden, funktioniert die Regel nicht mehr:

```
<div id="hauptteil">
  <p>Ein Absatz</p>
  <ul>
    <li>Wichtig</li>
    <li>Sehr wichtig!</li>
  </ul>
  <p>Dieser Absatz hat Stil!</p>
  <p>Noch ein Absatz</p>
  <p>et cetera ad nauseam</p>
</div>
```

Plötzlich greift die Regel nicht mehr, weil der gewünschte Absatz nicht mehr das zweite, sondern nun das dritte Kindelement von `div#hauptteil` ist.

Mit der Pseudoklasse `:nth-of-type()` können Sie diese Situation in den Griff bekommen, wenn Sie schreiben:

```
p:nth-of-type(2) { color: red; }
```

Nutzen Sie die Pseudoklasse `:nth-of-type()`, werden nur noch die Kindelemente gezählt, die Absätze sind.

Neben den oben bereits erwähnten Schlüsselwörtern `odd` und `even` für ungerade bzw. gerade Elemente können Sie mehr oder weniger komplexe Angaben zur Position des gewünschten Elements machen.

Die einfachste Version besteht in der Angabe einer einfachen Zahl. So steht `td:nth-child(4)` beispielsweise für das vierte `td`-Element.

Das ist allerdings erst der Anfang. Sie können auch ein *Intervall* definieren. Wenn Sie etwa nur jedes vierte Element mit einem bestimmten Stil versehen wollen, können Sie `td:nth-child(4n)` schreiben. Hiermit erhalten nur die Tabellenzellen 4, 8, 12 usw. die gewünschte Gestaltung.

Um festzulegen, für welches Element die Stildefinition als Erstes gelten soll, können Sie nach dem Intervall (jeder zweite, dritte usw.) zusätzlich einen Abstand angeben, sodass z.B. die Elemente 1, 5, 9 usw. einen eigenen Stil erhalten. Hierfür ergänzen Sie das Intervall um ein Pluszeichen (+), gefolgt vom Index des Elements, das als Erstes gestaltet werden soll. Für dieses Szenario sähe der Selektor dann so aus: `td:nth-child(4n+1)`.



Sie können den Abstand übrigens auch mit einem negativen Vorzeichen versehen. $4n+1$ und $4n-3$ oder auch $4n-7$ haben demnach den gleichen Effekt.

Dieses Rezept basiert auf dem Artikel *Useful :nth-child Recipes* von Chris Coyler.

Siehe auch

Die CSS3-Spezifikation zu den Pseudoklassen `:nth-child()`, `:nth-of-type()` usw. unter <http://www.w3.org/TR/css3-selectors/>; den Originalartikel von Chris Coyler unter <http://css-tricks.com/9657-useful-nth-child-recipes/>.

1.16 Das Boxmodell begreifen

Problem

Sie wollen besser verstehen, wie das CSS-Boxmodell funktioniert, und möchten wissen, auf welche Weise Innenabstände (`padding`), Außenabstände (`margin`) und Rahmen (`border`) den Inhalt umgeben.

Lösung

Jedes Blockelement (z.B. `p` oder `div`) erzeugt eine sogenannte Elementbox, die sich aus mehreren Schichten zusammensetzt. In Abbildung 1-14 sehen Sie eine grafische Darstellung des Boxmodells.

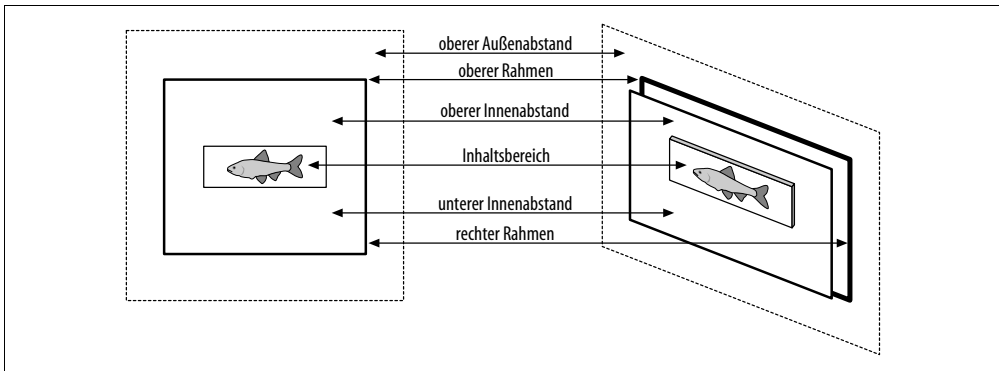


Abbildung 1-14: Das Boxmodell für ein Blockelement

Die einzelnen Teile des Boxmodells sind:

Inhaltsbereich (content)

Der eigentliche Inhalt wie etwa Text, Bilder, eingebundene Multimedia-Dateien oder auch andere Elemente. Der Inhaltsbereich befindet sich in der Mitte der Box und wird von den übrigen Schichten umgeben. Dieser Bereich wird auch als *Content-Box* bezeichnet.

Innenabstand (padding)

Umgibt den Inhaltsbereich. Er wird begrenzt von der Innenkante. Der von der Außenkante des Innenabstands umgebene Bereich (inklusive des Inhalts) wird als *Padding-Box* bezeichnet.

Rahmen (border)

Die nächste Schicht umgibt den Innenabstand und definiert den Rahmen der Box. Den von der Außenkante des Rahmens umgebenen Bereich (inklusive Padding-Box und Content-Box) nennt man auch *Border-Box*.

Außenabstand (margin)

Transparenter Bereich, der, beginnend an der Außenkante eines möglichen Rahmens, die übrigen Schichten umgibt. Die vier äußeren Ecken des mit margin definierten Bereich begrenzen die *Margin-Box*, die alle anderen Boxen (Border-Box, Padding-Box und Content-Box) enthält.

Der Außenabstand beginnt bündig mit der Außenkante des Rahmens. Ein Rahmen (border) mit dem Wert 0 ist seinerseits bündig mit der Außenkante des Innenabstands (padding). Hat der Innenabstand (padding) ebenfalls den Wert 0, ist er offensichtlich bündig mit der Außenkante des Inhaltsbereichs.

Diskussion

Jedes Element einer HTML-Seite erzeugt seine eigene Box. Je nach Art des Elements können sich die Boxen unterscheiden. So wird beispielsweise nach der Box eines Blockelements (`div`, `p` usw.) automatisch ein Zeilenumbruch eingefügt, wodurch Blockboxen prinzipiell untereinander angeordnet werden. Die Boxen der Inline-Elemente (`a`, `span`, `em`, `strong` usw.) werden dagegen zeilenweise dem Textfluss folgend neben- bzw. untereinander platziert.



Um eine möglichst browserübergreifende Darstellung des Boxmodells zu erreichen, sollte Ihr Dokument auf jeden Fall mit einer korrekten DOCTYPE-Deklaration versehen werden (siehe Rezept 1.22).

Das *Basic User Interface*-Modul der CSS3-Spezifikation definiert eine neue Eigenschaft namens `box-sizing`, mit der gesteuert werden kann, ob sich die Werte für `width` und `height` auf den Inhaltsbereich (Content-Box) oder den von den Außenkanten des Rahmens (border box) beziehen. Näheres hierzu finden Sie in Rezept 13.8.

Siehe auch

Das CSS-2.1-Boxmodell unter <http://www.w3.org/TR/CSS21/box.html>; das Brain Jar-Boxmodell unter <http://www.brainjar.com/css/positioning/default.asp>; das interaktive CSS-Boxmodell unter http://www.redmelon.net/tstme/box_model/; das Rezept 13.8 zur Verwendung der Eigenschaft `box-sizing`.

1.17 Darstellungsrollen verstehen

Problem

Sie wollen ein Element (z.B. ein Listenelement) anders darstellen, als ursprünglich vorgesehen (siehe Abbildung 1-15).

Lösung

Weisen Sie dem Element mit der CSS-Eigenschaft `display` die gewünschte Darstellungsrolle zu, z.B.:

```
#navigation li {  
    display: inline-block;  
}
```

Hier sehen Sie ein typisches Beispiel für die Änderung der Darstellungsrolle eines Elements: Die Elemente einer ungeordneten Liste werden standardmäßig untereinander angeordnet. Durch die Zuweisung von `inline-block` stehen sie nun nebeneinander.



Abbildung 1-15: Listenelemente in der Standarddarstellung und mit geänderter Darstellungsrolle

Diskussion

Damit Sie bei der Gestaltung Ihrer Webseiten nicht jedes Detail selbst definieren müssen, haben sich die Entwickler von CSS die *Darstellungsrollen* ausgedacht.

Sie legen fest, wie die verschiedenen HTML-Elemente grundsätzlich dargestellt werden, und gehen Hand in Hand mit dem in Rezept 1.16 vorgestellten Boxmodell. Grundsätzlich unterscheidet man zwei große Gruppen: *Blockelemente* und *Inline-Elemente*.

Blockelemente

Blockelemente werden prinzipiell untereinander angeordnet, und zwar in der Reihenfolge, in der sie im Quellcode des Dokuments vorkommen. Dieses Verhalten lässt sich durch verschiedene CSS-Eigenschaften (float, position usw.) verändern und anpassen.

Für Blockelemente stehen sämtliche im Boxmodell definierten Eigenschaften für die Gestaltung zur Verfügung. Typische Beispiele für Blockelemente sind p, div, h1, ul und form.

Inline-Elemente

Die zweite große Gruppe von Elementen sind die Inline-Elemente. Im Gegensatz zu Blockelementen werden Inline-Elemente zeilenweise angeordnet. Üblicherweise folgen sie dem automatischen Zeilenumbruch, der vom Browser je nach Breite des umgebenden Elements bzw. Ansichtsbereichs erzeugt wird.

Aus diesem Grund sind Eigenschaften wie `margin` und `border` für Inline-Elemente auch nur eingeschränkt anwendbar bzw. verhalten sich anders. Es werden nur die Werte für `margin-left` bzw. `margin-right` berücksichtigt. Innenabstände können für alle Seiten definiert werden. Sie haben zwar keinen Einfluss auf die Zeilenhöhe, aber auf die Platzierung eines möglichen Rahmens. Die Werte für `padding-left` und `padding-right` werden dagegen berücksichtigt.

Inline-Elemente sind beispielsweise `span`, `em` und `strong`.

Anonyme Inline-Elemente

Häufig kommt es vor, dass Textteile direkt von einem Blockelement umgeben werden. Das gängigste Beispiel ist Text innerhalb eines Absatzes, der nicht von einem Inline-Tag umgeben wird:

```
<p>Ich bin anonym. <span>Ich bin ein echtes Inline-  
Element.</span> Ich bin nicht ich selbst.</p>
```

Die Textteile *Ich bin anonym.* und *Ich bin nicht ich selbst.* sind anonyme Inline-Elemente. Obwohl sie nicht gesondert im HTML-Code markiert wurden, erben sie die Werte von ihrem Elternelement (dem Absatz), soweit dies möglich ist. (Mehr zur Vererbung erfahren Sie im folgenden Rezept.)

Inline-Block-Elemente

Einen Sonderfall stellen Inline-Block-Elemente dar. Sie werden wie Blockelemente dargestellt, folgen dabei aber dem Textfluss. Das heißt einerseits, dass Sie alle Gestaltungsmöglichkeiten für Blockelemente (z.B. Außenabstände) nutzen können. Dadurch haben sie allerdings auch Einfluss auf die Zeilenhöhe, die automatisch an das höchste Element angepasst wird.

Ersetzte Elemente

Einige Elemente stammen nicht aus dem HTML-Dokument selbst. Stattdessen definieren Sie im Quelltext einen Platzhalter. Die am häufigsten vorkommenden Beispiele für ersetzte Elemente stellen per `img`-Tag eingebundene Bilder dar. Hierbei wird das Element `` durch das Bild ersetzt, das unter der im `src`-Attribut angegebenen Adresse zu finden ist, und in die Darstellung der Seite eingefügt.

Weitere Beispiele für ersetzte Elemente sind `object`, die neuen HTML5-Elemente `video` und `audio`, aber auch Formularelemente wie `button`, `textarea`, `input` und `select`.

Ersetzte Elemente stellen hierbei keine neue Kategorie von Elementen dar, sondern sie definieren eine Sonderform bestehender Elemente. So sind `img`, `input`, `button` usw. ersetzte Inline-Elemente, während zum Beispiel `video` als ersetztes Inline-Block-Element definiert ist.

Die Eigenschaft display

Anhand der CSS-Eigenschaft `display` können Sie selbst festlegen, welche Darstellungsrolle ein Element haben soll. Um zum Beispiel ein Bild ausdrücklich als Blockelement zu definieren, können Sie schreiben:

```
img { display: block; }
```

Umgekehrt können Sie Blockelemente aber auch als Inline-Elemente definieren, indem Sie schreiben:

```
h1 {display: inline; }
```

Mit der obigen Regel werden Überschriften nun als Teil des Fließtexts betrachtet und entsprechend dargestellt.

Zusätzlich zu Block- und Inline-Elementen gibt es noch eine Reihe spezieller Darstellungsrollen. Eine Übersicht über die aktuell gültigen Werte (CSS 2.1) für die Eigenschaft `display` finden Sie in Tabelle 1-5.

Tabelle 1-5: Werte für die Eigenschaft display

Wert für die Eigenschaft display	Beschreibung
block	Blockelement
inline	Inline-Element
inline-block	Inline-Block-Element
list-item	Listeneintrag
table	Tabelle
inline-table	Tabelle im Textfluss
table-row-group	Gruppe von Tabellenzeilen (tr-Elementen)
table-header-group	Gruppe von Tabellenüberschriften (th-Elementen)
table-footer-group	Gruppe von Tabellenfußzeilen (tfoot)
table-row	Tabellenzeile (tr)
table-column-group	Gruppe von Tabellenspalten (colgroup)
table-column	Tabellenspalte (col)
table-cell	Tabellenzelle (td)
table-caption	Tabellenbeschriftung (caption)
inherit	Wert für display wird geerbt
none	Element wird nicht dargestellt

display: none

Neben den bereits erwähnten Werten für die Eigenschaft `display` ist der Wert `none` ebenfalls von besonderer Bedeutung. Wenn Sie die Deklaration `display: none` für ein Element angeben, nehmen Sie dieses komplett aus der Darstellung heraus, sein Platz wird gegebenenfalls für andere Elemente freigegeben.

Diese Deklaration wird häufig benutzt, um bestimmte Teile eines Dokuments zu verbergen. Dadurch ist es möglich, bestimmte Elemente nur bei Bedarf – etwa beim Überfahren eines Links mit dem Mauszeiger – anzuzeigen.



Die Anweisung `display: none` wirkt sich auch auf die Ausgabe einer Webseite mit Screenreader-Programmen aus.

Siehe auch

Die CSS-Spezifikationen des W3C zum Thema Darstellungsrollen unter <http://www.w3.org/TR/CSS2/visuren.html#display-prop>.

1.18 Die Vererbung nutzen

Problem

Sie wollen bestimmte Eigenschaftswerte nicht für jedes Kindelement neu definieren.

Lösung

Verlassen Sie sich auf die *Vererbung*. CSS ist in der Lage, die Werte für bestimmte Eigenschaften automatisch an die folgenden Kindelemente weiterzugeben.

Diskussion

Ein Kindelement erbt die Werte seines Elternelements, sofern dies in der CSS-Spezifikation nicht anders definiert ist. Das Kindelement erhält in diesem Fall automatisch die gleichen Eigenschaftswerte wie sein Elternelement. Legen Sie beispielsweise für einen Absatz, der ein `em`-Element enthält, eine bestimmte Schriftfamilie (`font-family`) fest, wird auch das enthaltene `em`-Element in dieser Schrift dargestellt:

```
p {  
    font-family: Helvetica, Arial, sans-serif;  
}
```

Hier der dazugehörige HTML-Code:

```
<p>Dieser Text enthält ein <em>em-Element</em>, das den gleichen Wert für font-  
family verwendet wie sein Elternelement, der Absatz.</p>
```

Das Ergebnis sehen Sie in Abbildung 1-16.

Für jede CSS-Eigenschaft ist ein Standardverhalten definiert, das festlegt, ob ihre Werte vererbt werden oder nicht. In den meisten Fällen entspricht das genau dem, was Sie wollen. Manchmal aber auch nicht.

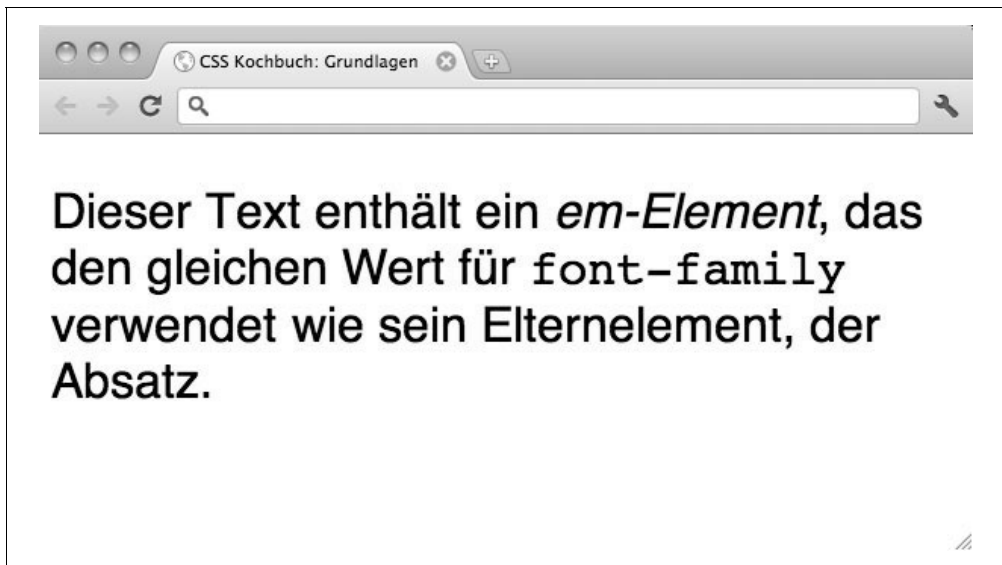


Abbildung 1-16: Der Wert für *font-family* wird vom *p*-Element an das enthaltene *em*-Element vererbt

Wenn Sie verhindern wollen, dass ein normalerweise vererbter Wert automatisch an Nachkommenelemente weitergereicht wird, müssen Sie den Wert ausdrücklich überschreiben. Sehen wir uns hierzu noch einmal das obige Beispiel an. Ohne weitere Angaben wird der Wert für *font-family* an das im Absatz enthaltene *em*-Element vererbt. Wollen Sie stattdessen eine andere Schriftfamilie verwenden, müssen Sie dies ausdrücklich angeben, zum Beispiel so:

```
p em { Helvetica, Arial, sans-serif; }
```

Hiermit werden alle *em*-Elemente innerhalb von Absätzen explizit in einer serifenlosen Schrift dargestellt.

Der Wert *inherit*

Es geht aber auch andersherum. Gelegentlich soll ein Element den Wert von einem Vorfahren erben, obwohl das eigentlich nicht vorgesehen ist. Für diesen Fall können Sie für die meisten Eigenschaften das Schlüsselwort *inherit* angeben.

Der Browser sucht daraufhin nach dem nächsten Vorfahren, der einen (zugewiesenen oder berechneten) Wert für die Eigenschaft besitzt, und verwendet dessen Wert, z.B.:

```
#erbgemeinschaft { background-color: inherit; }
```

Hiermit erhält das Element mit der ID *erbgemeinschaft* explizit die gleiche Hintergrundfarbe wie sein nächster Vorfahr, für den ein Wert angegeben oder berechnet wurde.

Siehe auch

Die CSS-2.1-Spezifikation zum Thema Vererbung unter <http://www.w3.org/TR/CSS21/cascade.html#inheritance>; das Rezept 1.16 zum Boxmodell; den Sitepoint-Artikel zur Vererbung unter <http://reference.sitepoint.com/css/inheritance>; den Sitepoint-Artikel zum Eigenschaftswert inherit unter <http://reference.sitepoint.com/css/inheritvalue>.

1.19 Die Reihenfolge von Stylesheets: die Kaskade

Problem

Sie wollen wissen, wie Konflikte zwischen CSS-Regeln gelöst werden, wenn zwei Regeln unterschiedliche Eigenschaftswerte für das gleiche Element definieren.

Lösung

In diesen Fällen greift ein Mechanismus, der als *Kaskade* bezeichnet wird (von der die Cascading Style Sheets übrigens ihren Namen haben). Je nachdem, aus welcher Quelle die Stildefinitionen stammen (Standardstile des Browsers, Stylesheet einer Website usw.), erhalten Sie eine bestimmte Priorität.

Diskussion

Wurde nur eine CSS-Regel für ein bestimmtes Element definiert, ist die Lage einfach. Der Browser wendet die Regel auf das Element an, und die Sache ist beendet.

Gelegentlich kommt es aber vor, dass sich verschiedene CSS-Regeln widersprechen. Das kann passieren, wenn in einem internen und in einem externen Stylesheet für das gleiche Element unterschiedliche Werte festgelegt werden oder im gleichen Stylesheet das gleiche Element mit verschiedenen Selektoren angesprochen wird. Um zu bestimmen, welche Regel letztendlich angewandt wird, werden mehrere Schritte durchlaufen, durch die geklärt wird, welche Stildefinition »gewinnt«.

Zunächst wird geklärt, aus welcher Quelle die Stildefinitionen stammen. Dies sind:

Browser-Stylesheet

Webbrowser verwenden eigene Grundformatierungen, um Webseiten darzustellen, wenn keine anderen Stylesheets angegeben wurden. Das W3C definiert ein »Standard-Stylesheet« mit den typischen Formatierungen von HTML-Elementen (beispielsweise werden das h1, div und p als Blockelemente angezeigt usw.). Die genaue Umsetzung unterscheidet sich jedoch oft von Browser zu Browser. Browser-Stylesheets sorgen dafür, dass HTML-Seiten nach einem gewissen Mindeststandard dargestellt werden, wenn keine weiteren Stile definiert wurden.

In den meisten Fällen wollen Sie die Regeln aus dem Browser-Stylesheet jedoch mit eigenen Anweisungen überschreiben.

Browser-Stylesheets haben die geringste Priorität. Sie können sowohl von Benutzer- als auch von Autoren-Stylesheets überschrieben werden.

Benutzer-Stylesheet

Anhand des Benutzer-Stylesheets kann der Benutzer eines Webbrowsers eigene Regeln für die Darstellung aufstellen. In einigen Browsern können Sie über die Einstellungen des Programms selbst angeben, wo auf Ihrem Rechner das Benutzer-Stylesheet liegt.

Das Benutzer-Stylesheet ist in vielen Browsern nicht gerade einfach zu erreichen. Sie müssen als Webautor allerdings wissen, dass der Benutzer hier bestimmte Regeln definieren kann, die sich auf die Darstellung auswirken.

Benutzer-Stylesheets haben Vorrang vor den browsereigenen Stilen. Sie können allerdings ohne Weiteres durch Autorenstile überschrieben werden.⁵

Autoren-Stylesheet

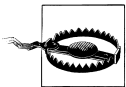
Dies sind die Stylesheets, mit denen Sie in den meisten Fällen arbeiten werden. Sie werden vom Autor der Webseite (also von Ihnen) definiert und anhand verschiedener Methoden (link-Tag, @import, intern usw.) in das Webdokument eingebunden. Im Gegensatz zu Browser- und Benutzer-Stylesheets liegen Autoren-Stylesheets auf einem Webserver.

Autoren-Stylesheets haben Priorität vor den Stilen des Benutzers und des Browsers. Auf diese Weise ist es überhaupt möglich, dass Sie die Gestaltung einer Webseite steuern können.

Gibt es widersprüchliche CSS-Regeln, versucht der Browser zunächst, den Konflikt anhand der Herkunft zu lösen. Sollte das nicht ausreichen, etwa weil beide Regeln aus einem Autoren-Stylesheet stammen, greift der nächste Schritt. In diesem wird geklärt, um welche Art von Stylesheet es sich handelt (extern, intern, Inline). Hierfür gelten folgende Grundsätze:

Im head-Teil des HTML-Dokuments definierte Regeln (interne Stylesheets) haben Vorrang vor externen Stylesheets. Hierbei spielt es keine Rolle, ob das externe Stylesheet per link-Tag oder @import eingebunden wurde.

Inline-Stile haben Vorrang vor internen und externen Stylesheets.



In den meisten Fällen ist der Nutzen von Inline-Stilen kleiner als der dafür zu zahlende Preis. Per style-Attribut definierte Stile blasen den HTML-Quelltext unnötig auf und sind so unflexibel wie die glücklicherweise abgeschafften font-Tags. Nach Möglichkeit sollten Sie Inline-Stile ganz vermeiden.

⁵ Einzige Ausnahme: Sie versehen die Regeln im Benutzer-Stylesheet mit der Markierung !important. Das sollten Sie allerdings nur tun, wenn Sie genau wissen, was Sie tun.

Spezifität

Kann trotz der Herkunft einer Regel nicht geklärt werden, welche Stile Vorrang haben, wird versucht, den Konflikt anhand der Spezifität zu lösen. Je spezifischer ein Selektor ist, desto mehr Gewicht hat er. Dieses Konzept wird im folgenden Rezept erklärt.

Manchmal kommt es vor, dass selbst anhand der Spezifität nicht geklärt werden kann, welche Regel Vorrang vor einer anderen hat. In diesem Fall greift das letzte und einfachste Konzept: Die zuletzt definierte Regel gewinnt.

!important

Wenn Sie eine CSS-Regel mit der `!important`-Deklaration versehen, erhält sie besondere Priorität. Nähere Informationen hierzu finden Sie in Rezept 1.21.

Siehe auch

Die folgenden Rezepte zu den Themen Spezifität und `!important`; David Barons Artikel zur Kaskade unter <http://dbaron.org/css/user/cascade.en.html>; die CSS 2.1-Spezifikation zur Kaskade unter <http://www.w3.org/TR/CSS21/cascade.html#cascade>; Teil 6 des CSS-Workshops auf css4you unter <http://www.css4you.de/wcss/css06.html>.

1.20 Darstellungskonflikte anhand der Spezifität lösen

Problem

Sie wollen wissen, was passiert, wenn trotz der Kaskade nicht geklärt werden kann, welche Regel vor einer anderen Vorrang hat.

Lösung

In diesem Fall werden die Regeln aufgrund ihrer *Spezifität* beurteilt. Nehmen Sie beispielsweise die folgenden CSS-Regeln:

```
#hauptteil p.gross {
    font-family: Impact, Haettenschweiler, "Arial Narrow Bold", sans-serif;
}

p.gross {
    font-family: Futura, "Century Gothic", AppleGothic, sans-serif;
}

p {
    font-family: "Gill Sans", Trebuchet, Calibri, sans-serif;
}
```

Jetzt wenden wir die Regeln auf diesen HTML-Code an:

```
<div id="hauptteil">
  <h1>Spezifität</h1>
```

```
<p class="gross">Lorem ipsum dolor sit amet...</p>
<p>Vestibulum vestibulum placerat ante...</p>
<p>Aenean sollicitudin tristique nulla id sollicitudin...</p>

</div>
```

Hier versuchen drei Regeln, die Absätze mit Stildefinitionen zu versehen. Die erste Regel bezieht sich einen Absatz, der mit der Klasse `gross` gekennzeichnet wurde und ein Nachfahrenelement von `#hauptteil` ist.

Die zweite Regel legt für den gleichen Absatz (mit der Klasse `gross`) eine andere Schrift fest. Die dritte Regel definiert schließlich eine allgemein Schrift für alle Absätze.

Welche Regel gewinnt? Für den ersten Absatz kommen im Prinzip alle drei Regeln infrage. Genau für diese Fälle haben sich die Erfinder von CSS das Konzept der Spezifität ausgedacht. Je präziser (spezifischer) eine Regel formuliert ist, desto mehr Vorrang erhält sie vor den anderen. Das Ergebnis sehen Sie in Abbildung 1-17.



Abbildung 1-17: Die CSS-Regel mit der höchsten Spezifität bestimmt die Darstellung

Diskussion

Je höher die Spezifität einer CSS-Regel, desto größer ist die Chance, dass sie sich gegen eine andere Regel durchsetzt. Um die Spezifität einer Regel zu bestimmen, werden die verschiedenen Selektoren gezählt. Außerdem wird untersucht, ob die Regel als Inline-Stil (mithilfe eines `style`-Attributs) definiert wurde.

Hierbei gilt:

- Inline-Stile haben eine höhere Spezifität als ID-Selektoren (z.B. #hauptteil).
- ID-Selektoren haben eine höhere Spezifität als Klassenselektoren (z.B. .gross).
- Klassenselektoren haben eine höhere Spezifität als Elementselektoren.
- Elementselektoren und Pseudoklassen/-elemente haben die gleiche Spezifität.
- Der universelle Selektor (*) besitzt eine Spezifität von 0,0,0,0.
- Vererbte Werte haben keine eigene Spezifität.

Die folgende Tabelle gibt Aufschluss über die Spezifität der Regeln in unserer Lösung:

Tabelle 1-6: Beispiel zur Bestimmung der Spezifität von Selektoren

Selektor-Beispiel	Inline-Stil	Anzahl ID-Selektoren	Anzahl Klassenselektoren	Anzahl Elementselektoren/Pseudoelemente/-klassen
p	0	0	0	1
p.gross	0	0	1	1
#hauptteil p.gross	0	1	1	1

Nach dieser Tabelle gilt:

- Die Spezifität des Selektors p beträgt 0,0,0,1.
- Die Spezifität des Selektors p.gross beträgt aufgrund des Klassenselektors 0,0,1,1.
- Die Spezifität des Selektors #kopfteil p.gross beträgt 0,1,1,1 wegen des ID-Selektors und des Klassenselektors.

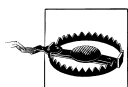
Das bedeutet, in unserem Beispiel hat der erste Selektor (#hauptteil p.gross) die höchste Spezifität und erhält daher in einem Konflikt den Vorrang vor den anderen – der erste Absatz wird in der Schrift »Impact« dargestellt.

Die folgenden beiden Absätze tragen kein class-Attribut und werden daher in der Schrift »Gill Sans« dargestellt.

Herkunft und Sortierreihenfolge der Stylesheets helfen dem Browser, festzustellen, welche Regeln vor anderen Vorrang bekommen. Reichen diese Regeln nicht aus, greift die Spezifität der Regel selbst.

Je höher die Spezifität einer CSS-Regel, desto größer die Wahrscheinlichkeit, dass sie gegenüber einer anderen Regel den Vorrang erhält.

Im Web gibt es eine Reihe von Websites, auf denen Sie die Spezifität eines Selektors berechnen lassen können, z.B. <http://www.suzyit.com/tools/specificity.php>.



Manchmal kann es passieren, dass zwei Regeln die gleiche Spezifität aufweisen. In diesem Fall gewinnt die Regel, die zuletzt im Stylesheet definiert wurde.

Siehe auch

Eric Meyers Artikel zum Thema Spezifität unter <http://meyerweb.com/eric/css/link-specificity.html>; Molly Holzschlags Artikel über die Spezifität in CSS2 und CSS 2.1 unter <http://www.molly.com/2005/10/06/css2-and-css21-specificity-clarified/>; den Artikel *CSS: Specificity Wars* von Andy Clarke unter http://www.stuffandnonsense.co.uk/archives/css_specificity_wars.html.

1.21 Regeln mit besonderer Priorität versehen

Problem

Sie wollen, dass eine CSS-Regel prinzipiell Vorrang vor anderen Definitionen erhält.

Lösung

Verwenden Sie die `!important`-Deklaration, um einer Regel ausdrücklich Priorität vor allen anderen zu verschaffen. Angenommen, in einem internen Stylesheet wird die folgende Regel definiert, die alle Links im Hauptteil der Seite in blauer Schrift darstellt:

```
div#hauptteil p a:link { color: blue; }
```

Unter Umständen haben Sie auf das HTML-Dokument oder das Template (z.B. bei einem Content-Management-System) keinen Zugriff, sondern nur auf das ebenfalls eingebundene externe Stylesheet. In diesem Fall können Sie Ihre eigene Linkfarbe definieren, indem Sie folgende Regel aufstellen:

```
a:link { color: green !important; }
```

Trotz einer deutlich niedrigeren Spezifität und der Definition in einem externen Stylesheet werden nun alle Links im Dokument in grüner Schrift angezeigt.

Diskussion

Die `!important`-Deklaration sorgt dafür, dass so markierte Regeln anderen Definitionen vorgezogen werden. Dadurch können Sie sich bei Bedarf über Regeln, die an anderer Stelle aufgestellt wurden, hinwegsetzen. Verwenden Sie diese Methode vorsichtig. Sie kann schnell zu Verwirrung führen – besonders wenn Sie mehrere `!important`-Deklarationen benutzen, die sich widersprechen.

Als `!important` gekennzeichnete Regeln in einem Autoren-Stylesheet können nur von `!important`-Deklarationen im Stylesheet des Benutzers überschrieben werden.

Siehe auch

Das Rezept 1.19 zur Reihenfolge der Stylesheets; das Rezept 1.20 zum Konzept der Spezifität; den Abschnitt »Kaskadierung« in Kapitel 3 von *CSS – Das umfassende Handbuch* von Eric A. Meyer (O'Reilly 2007); den Abschnitt zur `!important`-Deklaration in der CSS 2.1-Spezifikation unter <http://www.w3.org/TR/CSS21/cascade.html#important-rules>.

1.22 Dokumenttypen und ihre Auswirkungen auf die Browserdarstellung

Problem

Ihre Webseite soll standardkonform dargestellt werden und validierbar sein.

Lösung

Fügen Sie zu Beginn Ihres HTML-Dokuments eine gültige DOCTYPE-Deklaration ein.

Hier sehen Sie ein Beispiel für ein minimales Dokument gemäß der Dokumenttyp-Definition von HTML5:

```
<!DOCTYPE html>
<html>
<head>
<title>DOCTYPE-Deklarationen</title>
</head>

<body>

<p>Da die DOCTYPE-
Deklarationen rückwärtskompatibel sind, reicht es aus, die stark vereinfachte HTML5-
Form zu verwenden.</p>

</body>
</html>
```

Diskussion

Die Dokumenttyp- bzw. DOCTYPE-Deklaration steht am Anfang jedes Webdokuments noch vor dem `html`-Element. Sie erfüllt zwei Zwecke:

- Sie teilt Validatorprogrammen mit, welche HTML-Version verwendet wird.
- Sie sorgt dafür, dass der Browser versucht, das Dokument so standardkonform wie möglich darzustellen.

Ohne DOCTYPE-Deklaration können die Seiten nicht validiert werden, und manche Browser schalten in den sogenannten *Quirks-Modus*⁶. Der Internet Explorer 7 verwendet diesen Modus, um auch Seiten darstellen zu können, die fehlerhaften Markup-Code enthalten.⁷ Obwohl eine Seite vollkommen korrekten HTML- und CSS-Code enthält, kann es dadurch beim Fehlen einer DOCTYPE-Deklaration zu einer fehlerhaften Darstellung kommen.

⁶ Als *quirk* bezeichnet man im Englischen eine ungewöhnliche Eigenart, einen Spleen, eine Macke o.Ä.

⁷ Das kann im Internet Explorer 7 beispielsweise erreicht werden, indem Sie vor der eigentlichen DOCTYPE-Deklaration einen HTML-Kommentar einfügen.

Das World Wide Web Consortium (W3C) bietet webbasierte Validatorprogramme an, mit denen HTML- und CSS-Code auf seine Gültigkeit geprüft werden kann. Enthält ein HTML-Dokument keine DOCTYPE-Deklaration, gibt das Programm eine entsprechende Warnung aus.

HTML5

Im Laufe der vergangenen Jahre wurden viele verschiedene Versionen von HTML und XHTML entwickelt, die mehr oder weniger miteinander kompatibel sind. Verschiedene Entwicklungszweige und neue Versionen führten zu einem ganzen Zoo an möglichen Dokumenttypen.

Damit ein Browser ein Dokument standardkonform darstellt, reicht es aber schon aus, überhaupt einen gültigen Dokumenttyp anzugeben. Zudem ist die HTML5-DOCTYPE-Deklaration *rückwärtskompatibel*, d.h., sie funktioniert sogar im Internet Explorer 6.

Aus diesem Grund verwenden wir in diesem Buch für alle Beispiele den vereinfachten HTML5-Dokumenttyp.

Siehe auch

Die vorläufige HTML5-Spezifikation zum Thema DOCTYPE-Deklarationen unter <http://www.w3.org/TR/html5/syntax.html#the-doctype>; die HTML 4.01-Spezifikation zum Thema DTD unter <http://www.w3.org/TR/html401/intro/sgmltut.html#h-3.3>; die W3C-Validatorprogramme unter <http://validator.w3.org>; den Artikel zu DOCTYPEs auf der »A List Apart«-Website unter <http://www.alistapart.com/stories/doctype/>; einen Artikel zum Quirks-Modus unter <http://www.quirksmode.org/css/quirksmode.html>; die Informationen von Mozilla zum Quirks-Modus unter https://developer.mozilla.org/index.php?title=en/Mozilla%27s_Quirks_Mode; eine Erläuterung der Änderungen in HTML5 von Mark Pilgrim unter <http://diveintohtml5.info/semantics.html>; die Empfehlung auf der Microsoft-Website, die HTML5-DOCTYPE-Deklaration zu verwenden: [http://msdn.microsoft.com/en-us/library/cc288325\(VS.85\).aspx](http://msdn.microsoft.com/en-us/library/cc288325(VS.85).aspx); den Artikel *Quirks mode in IE 6 and IE 7* unter <http://www.satzansatz.de/cssd/quirksmode.html>.

1.23 Stylesheets für verschiedene Ausgabemedien definieren

Problem

Sie wollen für verschiedene Ausgabemedien (Druck, Bildschirmausgabe, PDAs, Vorlesegeräte usw.) eigene Stylesheets definieren.

Lösung

Erstellen Sie mehrere externe Stylesheets für die verschiedenen Medientypen und benennen Sie diese nach ihrem Verwendungszweck, z.B. *druck.css*, *bildschirm.css* und *projek-*

tion.css⁸.⁸ Danach benutzen Sie das Element `link` und erweitern dies um den Medientyp, um das entsprechende Stylesheet zu laden:

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

Die oben stehenden Zeilen binden drei Stylesheets in das Dokument ein: eines für die Druckausgabe der Seite (siehe Kapitel 9, *Drucken*), eines für die Bildschirmausgabe (der häufigste Fall) und eines für die Ausgabe mit Projektionsmedien, z.B. Beamern.

Mithilfe von `@media`-Abschnitten können Sie die Regeln für unterschiedliche Medientypen auch in einem gemeinsamen Stylesheet festlegen:

```
/* Stylesheet für die Druckausgabe */
@media print {
  body {
    font: 10pt Times, Georgia, serif;
    background-image: none;
  }
}

/* Stile für die Bildschirmausgabe */
@media screen {
  body {
    font: 1.2em Verdana, Arial, sans-serif;
  }
}

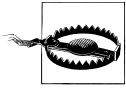
/* Ausgabe mit Projektionsmedien */
@media projection {
  body { font-size: 14px; }
}

/* Stildefinitionen für mehrere Medientypen */
@media screen, print, projection {
  body { line-height: 120% }
}
```

Diskussion

Medienspezifische Stile sind nur für Geräte sichtbar, die den entsprechenden Medientyp auch unterstützen. Stylesheets für andere Medientypen haben keinen Einfluss auf die Darstellung in anderen Medien oder die Webseite selbst. In Tabelle 1-7 finden Sie eine Liste der aktuell gültigen Medientypen aus der Spezifikation für CSS 2.1.

⁸ Das ist zwar nicht zwingend notwendig, aber es hilft bei der Arbeit. Im Prinzip können Sie Ihre Dateien auch *schnullerbacke.css*, *pippi-langstrumpf.css* oder *c3456.css* nennen. Allerdings bezweifeln wir ernsthaft, dass Sie in einem Jahr noch wissen, was damit gemeint war. Tun Sie dem Webworker, der Ihren Code irgendwann pflegen muss, einen Gefallen und verwenden Sie Namen, die auch später noch verständlich sind.



Das Konzept der Medientypen hat sich bis auf wenige Ausnahmen nicht durchgesetzt. In der Praxis kommen so gut wie ausschließlich die Medientypen `all`, `screen` und `print` zum Einsatz. Opera verwendet ab Version 9 den Medientyp `projection` für den Vollbildmodus.

Tabelle 1-7: Gängige CSS 2.1-Medientypen

Medientyp	Ausgabemedium
<code>all</code>	Für alle Ausgabegeräte gültig
<code>print</code>	Drucker und Druckvorschau
<code>projection</code>	Ausgabe auf Projektoren, Beamern usw.
<code>screen</code>	Farbmonitore

Siehe auch

Kapitel 9, *Drucken*, für die Definition spezieller Stile für die Druckausgabe; die deutsche Übersetzung der CSS-2.1-Spezifikation zu Medientypen unter <http://www.edition-w3.de/TR/1998/REC-CSS2-19980512/kap07.html>; den Artikel *ALA's New Print Styles* unter <http://www.alistapart.com/articles/alaprintstyles>; *Pocket-Sized Design: Taking Your Website to the Small Screen* unter <http://www.alistapart.com/articles/pocket>.

1.24 Den Inhalt eines Stylesheets richtig strukturieren

Problem

Sie wollen wissen, wie Sie den Inhalt eines Stylesheets so strukturieren können, dass dieser möglichst leicht zu warten ist.

Lösung

Eine Strukturierung von CSS-Regeln kann erreicht werden, indem die verschiedenen visuellen Elemente einer Webseite zu Gruppen zusammengefasst werden. In der folgenden Liste finden Sie einen Vorschlag für eine mögliche Gruppierungsreihenfolge der Elemente in einem Stylesheet:

- HTML-Elemente (`h1`–`h6`, `p`, `a`, Listen, Links, Bilder)
- Typografie (Schriftarten, -größen, -stile, Zeilenhöhen usw.)
- Seitenlayout (Kopfzeilen, Inhalt, Seitennavigation, globale Navigation, Subnavigation, Seitenleiste, Fußzeilen)
- Formulartags (`form`, `fieldset`, `label`, `legend`)
- Inhalte (Eintrag, Ereignis, Nachrichten)

Diskussion

Was für eine Person gut ist, muss für eine andere nicht unbedingt hilfreich sein. Daher sollten Sie diese Aufstellung als Empfehlung betrachten, die auf einer Kombination aus persönlicher Erfahrung und »Best Practices« basiert. Sie sollte für kleine bis mittlere Websites recht gut funktionieren.

Um herauszufinden, welche Art der Strukturierung für Sie am besten geeignet ist, besuchen Sie Ihre bevorzugten Websites und sehen sich deren Stylesheets an.

CSS-Kommentare

Kommentare können dabei helfen, Ihren Code zu erläutern und zu organisieren, damit er auch später noch verständlich ist. Mithilfe von Kommentaren können auch andere Menschen, die den Code nicht selbst erstellt haben, nachvollziehen, was gemeint war. Dinge, die zwischen `/*` und `*/` stehen, werden vom Browser ignoriert.

Wenn Sie Ihren Code in verschiedene Abschnitte unterteilen, können Kommentare dabei helfen, die einzelnen Abschnitte (Kopfzeile, Fußzeile, Navigation, Unternavigation usw.) besser zu unterscheiden. Kommentare können außerdem das Testen Ihrer Seiten stark erleichtern. Sollten Sie sich bei einer bestimmten Regel nicht sicher sein, ob diese korrekt ist oder welche Auswirkungen sie hat, umgeben Sie die Regel einfach mit Kommentarzeichen, um sie zu deaktivieren:

```
/*
a {
  text-decoration: none;
}
*/
```

Dadurch hat diese keinerlei Einfluss mehr auf die Darstellung der Webseite. Sofern es keine anderen Regeln für `a`-Tags gibt, werden Links nun erneut unterstrichen dargestellt, bis Sie die Kommentarzeichen wieder entfernen.

Wichtig ist nur, Ihre CSS-Regeln nicht einfach durcheinanderzuwürfeln. Überlegen Sie sich eine sinnvolle Reihenfolge und halten Sie sich daran. Verwenden Sie aussagekräftige Kommentare und rücken Sie Ihre Regeln anhand von Leerzeichen und Tabulatoren ein.

Siehe auch

Den Artikel *CSS Organization Tip 1: Flags* von Doug Bowman für eine Methode, bestimmte Regeln in Ihren CSS-Dateien zu finden, unter <http://www.stopdesign.com/log/2005/05/03/css-tip-flags.html>; den Artikel *CSS-Code organisieren* von Vladimir Simovic unter <http://www.perun.net/2010/03/30/css-code-organisieren/>; die CSS 2.1-Spezifikation zum Thema Kommentare unter <http://www.w3.org/TR/CSS21/syndata.html#comments>.

Grundrezepte

Einführung

In diesem Kapitel finden Sie allgemeine Rezepte, die Sie bei Ihrer Arbeit mit CSS immer wieder gebrauchen können. Anstatt im Verlauf des Buchs fünfmal zu erklären, wie Sie beispielsweise ein Element mit einem Rahmen versehen können (Tipp: mit der Eigenschaft `border`), zeigen wir Ihnen die wichtigsten Grundlagen in diesem Kapitel.

In Kapitel 1 haben wir uns mit dem Aufbau von CSS-Regeln, Selektoren und dem Boxmodell beschäftigt – hier finden Sie die ersten Anwendungsbeispiele. Es geht um die Definition von Abständen, Positionierung, die Definition von Hintergründen und noch einiges mehr.

2.1 Längeneinheiten verstehen

Problem

Sie wollen wissen, welche Längeneinheiten Sie in CSS verwenden können.

Lösung

Die CSS-Spezifikationen beschreiben eine Vielzahl von Möglichkeiten, Breiten, Höhen und Abstände zu definieren. Hierfür stehen drei Arten von Maßeinheiten zur Verfügung:

- absolute Werte (z.B. Zoll, Zentimeter, Punkt)
- relative Werte (z.B. em-Werte, px)
- Prozentwerte

Diskussion

Absolute Längeneinheiten

Prinzipiell unterscheidet man zwischen absoluten und relativen Längeneinheiten. Zu den absoluten Längeneinheiten gehören:

- Inches (in, Zoll)
- Zentimeter (cm)
- Millimeter (mm)
- Punkte (pt)
- Pica (pc)

Relative Längeneinheiten

Relative Längeneinheiten bestimmen die Länge einer Eigenschaft basierend auf dem Wert einer anderen Eigenschaft, die eine bestimmte Länge definiert. Hierzu gehören die folgenden Längeneinheiten:

- em-Einheiten (em)
- x-Höhe (ex)
- Pixel (px)

Pixel: relativ oder absolut?

Tatsächlich werden Pixel in der CSS-Spezifikation als »absolute« Maßeinheiten definiert, weil sie sich auf die physikalischen Bildpunkte des Ausgabemediums beziehen.

Das Problem dabei ist nur, dass die CSS-Maßeinheit *px* nicht unbedingt einem tatsächlichen Hardwarepixel entsprechen muss. Je nachdem, ob Sie sich eine Webseite auf einem alten 12-Zoll-Monitor oder auf einem iPhone mit Retina-Display ansehen, ist die Pixeldichte unterschiedlich hoch.

Aus diesem Grund bezeichnen wir die Maßeinheit *px* in diesem Buch als relative Maßeinheit.

em-Einheiten gehören zu den beliebtesten Maßeinheiten in CSS. Sie haben allerdings eine Besonderheit: Sie beziehen sich immer auf die Schriftgröße des aktuellen Elements. Angenommen, Sie haben die folgende Regel:

```
h1 {  
  font-size: 1.5em;  
  width:    10em;  
}
```

In diesem Fall wird die Überschrift nicht etwa 160 Pixel breit (10 multipliziert mit der Standardschriftgröße 16px) darstellt, sondern 240 Pixel breit. Der Grund liegt darin, dass Sie zuvor die Schriftgröße um das Eineinhalbfache erhöht haben.

Das gilt auch bei verschachtelten Elementen. Hat das Elternelement eine Schriftgröße von 0.8em und Sie definieren für ein Nachfahrenelement ebenfalls `font-size: 0.8em`, werden die Texte nicht etwa gleich groß dargestellt. Tatsächlich ist das Kindelement im Verhältnis zur Standardschriftgröße nur noch 0.64em groß.

Ähnliche Probleme können bei der Verwendung von Prozentwerten auftreten.



Alle aktuellen Browser verwenden eine Standardschriftgröße von 16 Pixeln bzw. 1em. Diese kann allerdings vom Benutzer in den Browsereinstellungen verändert werden.

rem-Einheiten in CSS3

Zusätzlich zu den hier genannten Maßeinheiten definiert das *Values and Units*-Modul der CSS3-Spezifikation eine Reihe weiterer möglicher Längeneinheiten, unter anderem die Einheit `rem`.

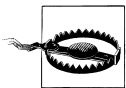
Angaben in `rem` verhalten sich im Prinzip wie `em`-Einheiten. Der Unterschied besteht darin, dass sich die Größe von `rem`-Einheiten immer von der Schriftgröße des Wurzelements herleitet – daher auch der Name `rem`, abgeleitet von »root em«.

Das bedeutet, sofern der Benutzer die Standardschriftgröße nicht verändert hat, entspricht `1rem` immer 16 Pixeln. Angaben in `rem` haben also überall die gleiche Bedeutung. `3.5rem` entsprechen immer 56 Pixeln – unabhängig davon, welche Größen für einzelne Elemente definiert wurden oder wo diese stehen.

Folgende Browser können mit `rem`-Einheiten umgehen: Internet Explorer 9+, Firefox 5+, Safari 5+, Chrome 13+, Opera 11.6+, iOS Safari 4+ und Android Browser 2.1+.

Andere relative Längeneinheiten

Die *x-Höhe* (`ex`) bezieht sich auf die Höhe des kleingeschriebenen Buchstabens `x` des verwendeten Zeichensatzes. Diese Einheit kommt jedoch nur sehr selten zum Einsatz. Deutlich häufiger verwendet man Pixel und Prozentwerte.



Bei der Angabe einiger Maßeinheiten sind auch Nachkommastellen erlaubt, z.B. `0.1em`. Beachten Sie hierbei, dass statt des Kommas der englische Dezimalpunkt (`.`) verwendet werden muss.

Prozentwerte

Prozentwerte (z.B. `120%`) beziehen sich immer auf einen anderen Wert, z.B. eine Längenangabe. Welcher Wert dies im Einzelnen ist, hängt von der Eigenschaft ab, für die er definiert wurde.

Siehe auch

Die CSS2-Spezifikation zu Längeneinheiten unter <http://www.w3.org/TR/REC-CSS2/syn-data.html#length-units>; das *Values and Units*-Modul der CSS3-Spezifikation zu rem-Einheiten unter <http://www.w3.org/TR/css3-values/#relative>.

2.2 Höhen und Breiten definieren

Problem

Sie wollen für bestimmte Elemente (siehe Diskussion) eine Höhe und eine Breite festlegen.

Lösung

Verwenden Sie die Eigenschaften `height` und `width`, um dem Element eine ausdrückliche Breite und/oder Höhe zuzuweisen. Der folgende Code sorgt dafür, dass das Element mit der ID `#seitenleiste` die gesamte Höhe und 20% der Breite des umgebenden Elements einnimmt:

```
#seitenleiste {  
    height: 100%;  
    width: 20%;  
}
```

Die Höhe von 100% bezieht sich auf die Höhe des umgebenden Elements, z.B. eines `div`-Containers. Wurde für diesen keine ausdrückliche Höhe definiert, passt sie sich dem Inhalt an.

Diskussion

Höhe und Breite können nur für Block- und ersetzte Inline-Elemente (z.B. Bilder, Formularelemente) definiert werden. Hierbei steht die Eigenschaft `height` für die Höhe und `width` für die Breite des Elements. Als Maßeinheiten können alle in CSS gültigen Längeneinheiten sowie Prozentwerte benutzt werden. Negative Werte sind nicht möglich.

Siehe auch

Das Rezept 2.1 zum Thema Längeneinheiten; die CSS-Spezifikation zum Thema Breite und Höhe unter <http://www.w3.org/TR/CSS2/visudet.html#the-width-property> bzw. <http://www.w3.org/TR/CSS2/visudet.html#the-height-property>.

2.3 Farben definieren

Problem

Sie wollen wissen, auf welche Weise Sie Farbwerte (für Eigenschaften wie `color` und `background-color`) angeben können.

Lösung

CSS kennt eine Vielzahl von Möglichkeiten, Farbwerte zu definieren. Hierbei ist die Eigenschaft `color` hauptsächlich für die Textfarbe¹ zuständig, während `background-color` die Hintergrundfarbe verschiedener Elemente definiert.

Hier einige der gängigsten Möglichkeiten am Beispiel der Eigenschaft `background-color`:

```
body {  
    background-color: white;  
}
```

oder:

```
body {  
    background-color: #ffffff;  
}
```

oder auch:

```
body {  
    background-color: rgb(255,255,255);  
}
```

bzw.:

```
body {  
    background-color: rgb(100%,100%,100%);  
}
```

oder auch (CSS3):

```
body {  
    background-color: rgba(255,255,255,0.6);  
}
```

bzw. (CSS3):

```
body {  
    background-color: hsla(0,100%,100%,0.6);  
}
```

¹ Tatsächlich kann sich die CSS-Eigenschaft `color` auch auf andere Dinge auswirken, beispielsweise auf die Farbe eines möglichen Rahmens. Zum Testen können Sie einfach ein Element mit einem Wert für `color` versehen. Danach definieren Sie per `border-style` einen Rahmen für das Element, ohne ausdrücklich eine Farbe anzugeben: `<div style="color: red; border: solid;">TEST</div>`. In diesem Fall wird das `div`-Element mit einem roten Rahmen umgeben, der seine Farbe von `color` erhält.

Die ersten drei Beispiele sind gleichbedeutend: Sie legen fest, dass der Ansichtsbereich mit einem weißen Hintergrund versehen wird. Im ersten Beispiel benutzen wir das Schlüsselwort `white`, die zweite Variante verwendet einen hexadezimalen Wert, während die Farbe in der dritten Option als RGB-Wert angegeben wurde.

Das vierte Beispiel benutzt die in CSS3 neu hinzugekommene `rgba()`-Schreibweise, das fünfte verwendet die ebenfalls in CSS3 eingeführte `hsla()`-Notation.

Diskussion

Farben sind ein Grundbaustein bei der Gestaltung von Webseiten. Hiermit können bestimmte Layoutbereiche visuell voneinander getrennt und bestimmte Teile hervorgehoben werden. CSS bietet eine Reihe von Möglichkeiten, Farben zu definieren.

Benannte Farben

Die CSS 2.1-Spezifikation definiert 17 Farbnamen, basierend auf dem ursprünglichen Windows-VGA-Standard.² In CSS3 wurde diese Liste auf 140 Namen erweitert, die auch jetzt schon von vielen Browsern verstanden wird.

Hexadezimale Werte

Häufig kommen auch hexadezimale Werte zum Einsatz (z.B. `#a4d3ee`). Diese Schreibweise ermöglicht die Definition von über 16 Millionen ($256 \times 256 \times 256$) Einzelfarben. Dem Wert wird hierbei ein Doppelkreuz (`#`) vorangestellt. Darauf folgen drei Paare hexadezimaler Zahlen, die jeweils eine Teilfarbe (Rot, Grün, Blau) definieren.



Bestehen die Paare aus gleichen Zahlen oder Buchstaben, zum Beispiel `#bb33ff` oder `#000000`, können Sie den Wert »halbieren«. Es reicht dann aus, wenn Sie `#b3f` oder `#000` schreiben.

rgb- und rgba-Werte

Außerdem können die Farben auch als RGB-Werte angegeben werden. Hierbei wird dem Wert das Kürzel `rgb` vorangestellt. Die Werte stehen durch Kommata getrennt innerhalb runder Klammern. Sie können entweder als ganzzahlige Werte zwischen 0 und 255 (`rgb(150,0,20)`) oder als Prozentwerte (`rgb(20%,50%,4%)`) angegeben werden.

Diese Schreibweise wurde in CSS3 um die Möglichkeit erweitert, einen Alphakanal (das `a` in `rgba`) festzulegen. Dieser definiert, wie durchscheinend die gewünschte Farbe sein soll. Dadurch ist es möglich, Transparenzen verschiedener Stärke auch ohne die Verwendung von PNG-Grafiken umzusetzen (siehe auch Rezept 2.15). Die Transparenz wird als Wert

² Hierzu gehört das Schlüsselwort `gray`, das die meisten Browser gnädigerweise auch in der englischen Schreibweise `grey` verstehen. Man lernt eben nie aus.

zwischen 1.0 (keine Transparenz) und 0.0 (vollkommen durchsichtig) angegeben, z.B.: `rgba(50%,50%,50%,0.5)` – 50% Grau, halb transparent.



Anders als bei der einfachen `rgb()`-Schreibweise können für `rgba()` keine hexadezimalen Werte verwendet werden.

Das funktioniert übrigens nicht nur für Hintergründe, sondern überall dort, wo Sie eine Farbe als Wert angeben können, z.B. `color`.

Da die `rgba`-Schreibweise vom Internet Explorer erst ab der Version 9 unterstützt wird, sollten Sie nach Möglichkeit einen passenden Fallback-Wert definieren, z.B.:

```
background: rgb(250,250,250); /* für ältere Browser */
background: rgba(255,255,255,0.6);
```

Hierbei ist die Reihenfolge wichtig. In Browsern, die die `rgba`-Notation unterstützen, wird die erste Definition durch die zweite überschrieben, und dadurch wird der zweite Wert benutzt. Versteht ein Browser die zweite Deklaration nicht, wird sie ignoriert. Stattdessen wird die erste Angabe verwendet, wie in Abbildung 2-1 gezeigt.

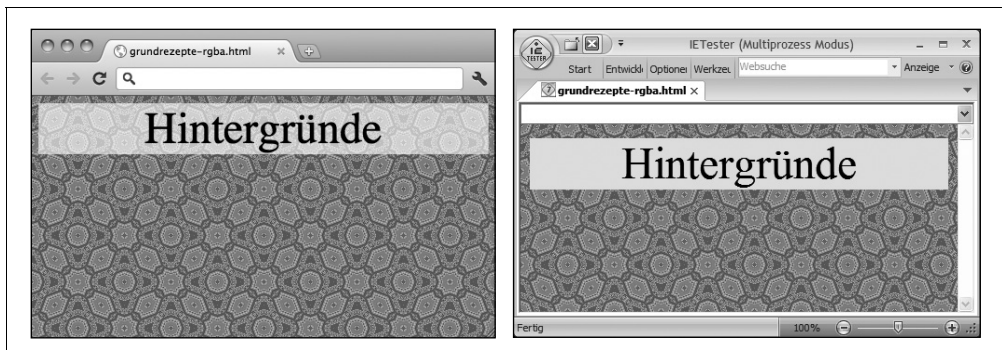
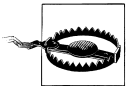


Abbildung 2-1: Moderne Browser verwenden die `rgba()`-Schreibweise, ältere Browser wie der IE 7 (hier rechts) greifen auf die einfachere `rgb()`-Notation zurück



Damit die Definition einer Fallback-Farbe auch im Internet Explorer 7 funktioniert, müssen Sie die Kurzschrifteigenschaft `background` verwenden. Ansonsten wird die Angabe vom IE7 ignoriert.

hsla()-Werte

Alternativ zur `rgba()`-Schreibweise steht in CSS3 außerdem die `hsla()`-Notation zur Verfügung. Anstelle von Teilfarben geben Sie hier die Werte für Farbtön (Hue), Sättigung (Saturation) und Helligkeit (Lightness) an. Hierbei werden die Farben auf einem Kreis angeordnet. Durch die Angabe eines Winkels kann die Grundfarbe bestimmt werden.

Dabei entspricht 0° der Farbe Rot, Grün liegt bei 120° und Blau bei 240°. Der Winkel wird als einfacher numerischer Wert zwischen 0 und 360 entsprechend der Gradzahl für die gewünschte Farbe angegeben.

Die Werte für Sättigung und Helligkeit werden dagegen in Prozentwerten definiert. Die Transparenz wird wie bei `rgba()` als Fließkommazahl zwischen 0 und 1 angegeben. Hier ein paar Beispiele:

```
hsla(240, 100%, 50%, 0.5); /* Blau halb transparent */  
hsla( 30, 100%, 50%, 0.1); /* fast durchsichtiges Orange
```

Befürworter dieser Schreibweise argumentieren, dass die Verwendung der `hsla()`-Schreibweise intuitiver sei als die `rgba()`-Notation. Sobald Sie die Position der Farbe im Farbkreis kennen, lässt sich mit ein wenig Übung an der Definition ablesen, welcher Farbton erzeugt wird. Andererseits werden viele Webworker bereits mit der `rgba()`-Form vertraut sein. Sie haben die Wahl. Am Ende zählt das Ergebnis.

`rgba()` und `hsla()` werden von allen aktuellen Browsern (IE ab Version 9, Opera ab Version 10) unterstützt.

Siehe auch

Die CSS 2.1-Spezifikation zu den background-Eigenschaften unter <http://www.w3.org/TR/CSS2/colors.html#background-properties>; das CSS3-Color-Modul unter <http://www.w3.org/TR/css3-color/#alphavalue>; den englischen Wikipedia-Eintrag zum Thema Webfarben unter http://en.wikipedia.org/wiki/Web_colors; einen Artikel auf css.info zum Thema `rgba` unter <http://www.css3.info/preview/rgba/>; die Website css-tricks.com zum »rgb-Bug« im Internet Explorer 7 unter <http://css-tricks.com/ie-background-rgb-bug/>.

2.4 Externe Ressourcen einbinden

Problem

Sie möchten externe Ressourcen wie zusätzliche Stylesheets, Webfonts oder Bilder einbinden.

Lösung

Geben Sie den Pfad zur externen Datei mithilfe der `url()`-Schreibweise an, wie hier gezeigt:

```
background-image: url("/bilder/hintergrund.jpg");
```

oder auch:

```
@import url("http://www.csskochbuch.com/css/externes-stylesheet.css");
```

Diskussion

Um externe Ressourcen (z.B. per `background-image`, `@import`, `@font-face` oder auch `content`) in Ihr Stylesheet einzubinden, müssen Sie dem Browser mitteilen, wo die Datei zu finden ist. Das geschieht in CSS mit der `url()`-Schreibweise. Innerhalb der runden Klammern können Sie den Verzeichnispfad oder den URI angeben, unter dem die Ressource sich befindet. Die Schreibweise ist die gleiche, die Sie auch bei `img`-Tags für das `src`-Attribut verwenden würden. Hierbei kann der URI mit einfachen oder doppelten Anführungszeichen umgeben werden.

Offiziell ist die Verwendung von Anführungszeichen nicht notwendig. Allerdings hilft ihr Einsatz bei der Vermeidung von Problemen mit komplexen Webadressen (z.B. wenn Sonderzeichen enthalten sind). In diesem Buch verwenden wir doppelte Anführungszeichen.

Beachten Sie, dass mit `url()` angegebene Pfade (z.B. `/css/externes-stylesheet.css`) relativ zum Speicherort des Stylesheets interpretiert werden. Angenommen, Sie haben folgende Verzeichnisstruktur:

```
/htdocs/index.html
/htdocs/css/externes-stylesheet.css
/htdocs/bilder/hintergrund.jpg
```

Hierbei bindet die Datei *index.html* das Stylesheet im Verzeichnis *css* ein. In *externes-stylesheet.css* wird wiederum ein Hintergrundbild für die Seite definiert. Damit dies auch wirklich geladen wird, müssen Sie im externen Stylesheet schreiben:

```
background-image: url("../bilder/hintergrund.jpg");
```

Diese Angabe besagt: »Gehe um eine Verzeichnisebene nach oben und steige dann in das Verzeichnis *bilder* ab. Suche dort nach dem Bild *hintergrund.jpg*.« Alternativ dazu können Sie auch einen Pfad relativ zum Wurzelverzeichnis des Webserverns angeben:

```
background-image: url("/bilder/hintergrund.jpg");
```

Hierbei werden die Pfade immer vom Wurzelverzeichnis des Webserverns (*htdocs*) aus interpretiert. Oder aber Sie verwenden einen vollständigen URI:

```
background-image: url("http://www.csskochbuch.com/bilder/hintergrund.jpg");
```

Folgendes wird nicht funktionieren:

```
background-image: url("bilder/hintergrund.jpg");
```

Hierfür müsste sich die CSS-Datei im gleichen Verzeichnis befinden wie die Datei *index*.

Siehe auch

Die CSS 2.1-Spezifikation zur `url()`-Schreibweise unter <http://www.w3.org/TR/CSS2/syndata.html#uri>; das Rezept 3.11 zu Webfonts.

2.5 Außenabstände definieren

Problem

Sie wollen den Abstand eines Elements von den Rändern des Darstellungsbereichs (oder des umgebenden Elements) selbst festlegen.

Lösung

Verwenden Sie die Kurzschrifteigenschaft `margin` oder eine der entsprechenden Einzeleigenschaften, z.B.:

```
p {  
  margin:    1em;  
  margin-left: 2em;  
}
```

Hierdurch erhält der Absatz rundherum einen Außenabstand von 1em. Dieser Wert wird für die linke Seite überschrieben (siehe Abbildung 2-2).



Abbildung 2-2: Ein Absatz mit explizit definierten Außenabständen

Diskussion

Die Außenabstände (`margin`) legen fest, wie groß der Leerraum zwischen dem Element und seinen Nachbarelementen sein soll. In diesem Bereich werden keine anderen Elemente platziert (es sei denn, sie wurden hier ausdrücklich positioniert). Es ist der Hintergrund des umgebenden Elements zu sehen.

Für die meisten HTML-Elemente definieren die Browser bestimmte Ausgangswerte. Diese sind aber nicht immer gleich. Daher werden Sie für Ihr Layout oftmals eigene Einstellungen verwenden wollen. Mit den `margin`-Eigenschaften können Sie selbst festlegen, wie viel Leerraum um ein Element freigelassen werden soll.

Für jede Seite eines Elements gibt es eine eigene `margin`-Eigenschaft:

- `margin-top`: oberer Außenabstand
- `margin-right`: rechter Außenabstand
- `margin-bottom`: unterer Außenabstand
- `margin-left`: linker Außenabstand

Sie können die einzelnen Außenabstände aber auch mit einer sogenannten *Kurzschrift-eigenschaft* gemeinsam definieren. Sie heißt schlicht `margin`:

```
margin: 0.5em 0.2em 0.5em 0.2em;
```

Hiermit werden die Außenabstände in der Reihenfolge oben, rechts, unten, links (sprich: im Uhrzeigersinn) festgelegt. Um sich die Reihe leichter merken zu können, hilft Ihnen vielleicht die englische Eselsbrücke `TRouBLE` (für `top`, `right`, `bottom` und `left`).

Als Maßeinheit kann eine beliebige Längeneinheit oder ein Prozentwert verwendet werden (z.B. `5px`, `2em` oder `10%`). Auch negative Werte können angegeben werden.

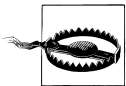
Automatische Außenabstände

Anstelle einer Maßeinheit wird für den linken und rechten Außenabstand häufig auch das Schlüsselwort `auto` verwendet, um ein Element horizontal zu zentrieren. Mehr zu diesem Thema finden Sie in Rezept 2.24 in diesem Kapitel.

Negative Außenabstände

Für die verschiedenen `margin`-Eigenschaften können Sie übrigens auch negative Werte angeben. Wenn Sie ein Element um 20 Pixel nach links verschieben möchten, könnten Sie beispielsweise schreiben:

```
p.ausgerueckt {  
  margin-left: -20px;  
}
```



Bei der Verwendung negativer Außenabstände kann es passieren, dass Elemente ganz oder teilweise aus dem Ansichtsbereich herausgeschoben werden.

Zusammengefasste vertikale Außenabstände

Laut CSS-Spezifikation werden obere und untere Außenabstände für Blockelemente zusammengefasst. Es wird hierbei nur der jeweils größere Wert für `margin-top` bzw. `mar-`

gin-bottom berücksichtigt. Wie das funktioniert, lässt sich am besten an einem Beispiel zeigen. Angenommen, Sie definieren für zwei aufeinanderfolgende div-Elemente diese CSS-Regeln:

```
div:first-child {  
    margin-bottom: 3em;  
}  
  
div + div {  
    margin-top: 2em;  
}
```

In diesem Fall ist der Wert für margin-bottom größer und wird daher vom Browser verwendet, um den Abstand zwischen den Elementen #oben und #unten festzulegen. Der Wert für margin-top des unteren Elements wird verworfen, wie in Abbildung 2-3 gezeigt.

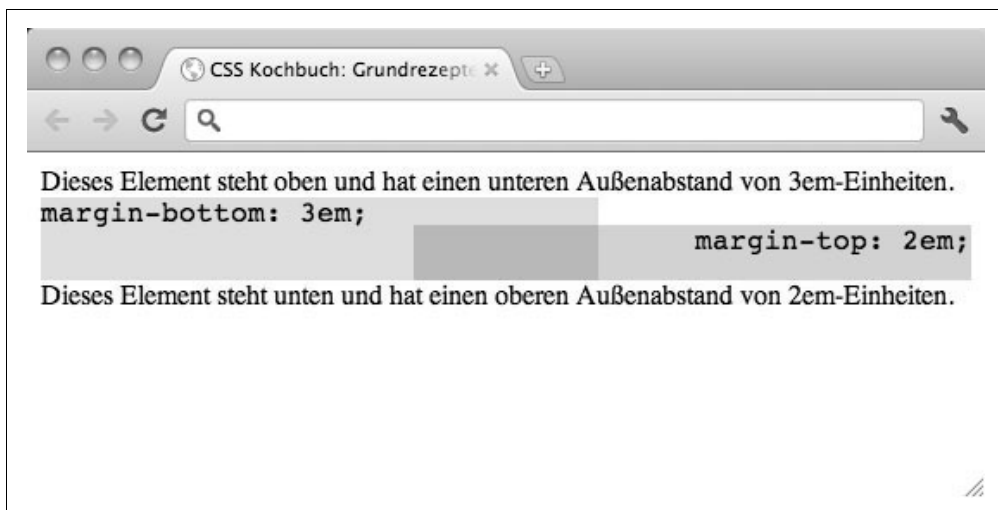


Abbildung 2-3: Zusammengefasste vertikale Außenabstände

Die oberen und unteren Außenabstände werden übrigens nicht nur zusammengefasst, wenn sich die Elemente wie im obigen Beispiel auf der gleichen Hierarchieebene befinden, sondern auch, wenn es sich um Vor- und Nachfahrenelemente handelt.

Für folgende Elemente werden die vertikalen Außenabstände nicht zusammengefasst:

- Inline-Elemente (weil diese per definitionem keine oberen und unteren Außenabstände besitzen).
- Elemente, für die ein Rahmen (border) oder ein Innenabstand (padding) definiert wurde.
- Elemente, die als Float (siehe Rezept 2.23) definiert wurden.
- Absolut positionierte Elemente.
- Inline-Block-Elemente.

- Elemente, für die die Eigenschaft `overflow` einen anderen Wert hat als `visible` (gilt nur für gemeinsame Außenabstände mit Kindelementen).
- Elemente, deren Wert für die Eigenschaft `clear` anders lautet als `none` (Standardwert). Obere Außenabstände solcher Elemente werden nicht mit dem Block des Elternelements zusammengefasst.

Seitliche Außenabstände werden niemals zusammengefasst. Wurde für eines der Elemente ein negativer Außenabstand definiert (siehe weiter oben), werden beide Werte addiert, um den tatsächlichen Abstand zu ermitteln.

Siehe auch

Die CSS 2.1-Spezifikation zum Thema Außenabstände unter <http://www.w3.org/TR/CSS2/box.html#margin-properties>; das Rezept 1.16 zum Boxmodell; das Rezept 13.16 zum Zurücksetzen der Browser-Standardstile; der Sitepoint-Artikel zu zusammengefassten Außenabständen unter <http://reference.sitepoint.com/css/collapsingmargins>; das Rezept 2.1 zu den in CSS möglichen Längeneinheiten.

2.6 Innenabstände definieren

Problem

Sie wollen festlegen, wie groß der Leerraum zwischen dem Inhalt eines Elements und dessen Rahmen sein soll.

Lösung

Verwenden Sie die Kurzschrifteigenschaft `padding` oder eine der Einzeleigenschaften für die verschiedenen Seiten, wie hier gezeigt:

```
p {  
    padding: 15px;  
}
```

Diese Regel sorgt dafür, dass der Text aller Absätze mit einem Innenabstand von 15 Pixeln umgeben wird. Um den Abstand nur für bestimmte Seiten einzustellen, gibt es die Einzeleigenschaften `padding-top`, `padding-right`, `padding-bottom` und `padding-left`. Es können alle in CSS gültigen Längeneinheiten sowie Prozentwerte verwendet werden. Im Gegensatz zu Außenabständen sind hier jedoch keine negativen Werte erlaubt.

Diskussion

Das CSS 2.1-Boxmodell definiert Innenabstände als Abstand zwischen dem Inhaltsbereich und der Außenkante eines eventuell definierten Rahmens (siehe unten) für das Element, wie in Abbildung 2-4 gezeigt.



Abbildung 2-4: Ein Absatz mit einem Innenabstand von 15 Pixeln

Die Angabe von Werten für padding bzw. die Einzeleigenschaften funktioniert analog zu margin. Im Gegensatz zu margin kann padding allerdings auch für Inline-Elemente verwendet werden.

Siehe auch

Die CSS 2.1-Spezifikation zum Thema Innenabstände unter <http://www.w3.org/TR/CSS2/box.html#padding-properties>; das Rezept 1.16 zum Boxmodell.

2.7 Elemente mit einem Rahmen versehen

Problem

Sie wollen einem Element einen Rahmen geben.

Lösung

Verwenden Sie die Eigenschaft border bzw. die verschiedenen border-Einzeleigenschaften, um Dicke, Gestaltung und Farbe des Rahmens zu steuern, wie hier gezeigt:

```
ul {  
    border: 1px solid black;  
}
```

Diese Regel umgibt ungeordnete Listen mit durchgehenden schwarzen Rahmen (siehe Abbildung 2-5) von 1 Pixel Breite. Wir verwenden hier die Kurzschrifteigenschaft border. Tatsächlich gibt es gerade für die Definition von Rahmen einen ganzen Zoo von Einzeleigenschaften, die in der folgenden Diskussion erläutert werden.

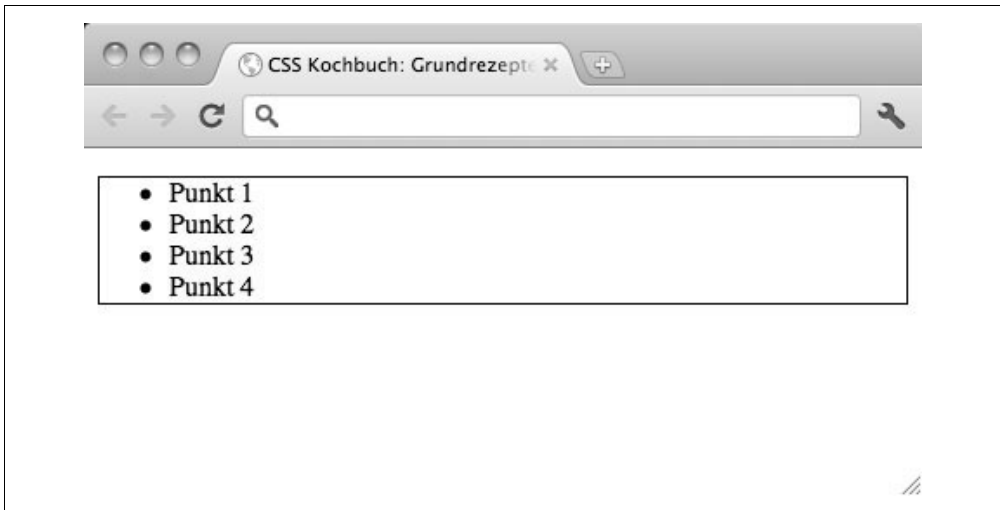


Abbildung 2-5: Die Liste erhält einen schwarzen Rahmen von 1 Pixel Breite

Diskussion

Neben der Kurzschrifteigenschaft `border` definieren die CSS 2.1-Spezifikation sowie das *Borders and Backgrounds*-Modul der CSS3-Spezifikation eine Reihe von Einzeleigenschaften, mit denen sich die Details bei der Darstellung von Rahmen sehr genau steuern lassen.

Einzeleigenschaften für Breite, Stil und Farbe

Was wir im obigen Beispiel auf einer Zeile definiert haben, lässt sich auch folgendermaßen ausdrücken:

```
ul {  
  border-width: 1px;  
  border-style: solid;  
  border-color: black;  
}
```

Jeder Aspekt der Rahmengestaltung wird hier mit einer eigenen Eigenschaft ausgedrückt. Dadurch werden die Anweisungen lesbarer, aber auch umfangreicher.

Damit ist das Ende der Möglichkeiten aber noch nicht erreicht. Sie können diese Angaben auch nur für einzelne Seitens des Rahmens machen:

```
ul {  
  border-left-width: 1px;  
  border-left-style: solid;  
  border-left-color: black;  
}
```

Diese Regel funktioniert genau wie das Beispiel davor – es gilt allerdings nur für die linke Rahmenseite. Entsprechende Eigenschaften gibt es natürlich auch für die anderen Rahmenseiten. Für die Rahmenbreite können Sie eine beliebige CSS-Längeneinheit oder einen Prozentwert angeben (negative Werte sind nicht erlaubt); der Rahmenstil wird durch ein Schlüsselwort angegeben. Die gängigsten Werte sind: `solid` (durchgehend), `dashed` (gestrichelt) und `dotted` (gepunktet). Die Rahmenfarbe wird durch eine beliebige CSS-Farbangabe (benannt, hexadezimal, `rgba()` usw.) bestimmt. (Wenn Sie keine explizite Farbe angeben, wird stattdessen der aktuelle Wert von `color` verwendet.)

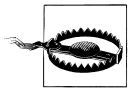


Die Darstellung des Stils `dotted` kann sich von Browser zu Browser unterscheiden. Während die Deklaration `border: 10px dotted red;` in Firefox tatsächlich Kreise mit einem Durchmesser von 10 Pixeln erzeugt, sind es in Chrome und Safari 10 Pixel dicke Rechtecke.

Kurzschrifteigenschaften für einzelne Rahmenseiten

Damit Sie sich bei der Definition von Rahmen nicht die Finger wund schreiben und trotzdem die Kontrolle über die einzelnen Rahmenseiten behalten, gibt es für jede Seite außerdem eine eigene Kurzschrifteigenschaft. Das vorige Codebeispiel für die linke Rahmenseite kann man demnach so zusammenfassen:

```
ul {  
  border-left: 1px solid black;  
}
```



Im Gegensatz zu anderen Kurzschrifteigenschaften wie `margin` und `padding` ist es bei `border` nicht möglich, Werte für die einzelnen Rahmenseiten zu definieren, sondern nur für alle vier Seiten auf einmal.

Oft kommt es vor, dass Sie einen Rahmen für ein Element definieren, bei dem eine Rahmenseite anders dargestellt werden soll als der Rest. In diesem Fall ist es nicht nötig, alle Rahmenseiten einzeln zu definieren. Stattdessen geben Sie zuerst die allgemeine Regel an und überschreiben diese danach mit der entsprechenden Einzeleigenschaft:

```
border: 1px solid black;  
border-left: none;
```

Die Definitionen für die übrigen Rahmenseiten bleiben hierbei erhalten.

Schlüsselwörter

Sie können die Rahmenbreite übrigens auch anhand der Schlüsselwörter `thin`, `medium` und `thick` angeben. Hierbei entspricht der Wert `thin` meist 1 Pixel Breite, `medium` steht für 3 Pixel, und `thick` führt zu einer Breite von 5 Pixeln. Laut der CSS 2.1-Spezifikation ist die Umsetzung letztendlich aber den Browserprogrammen überlassen.

Siehe auch

Die CSS 2.1-Spezifikation zum Thema Rahmen unter <http://www.w3.org/TR/CSS2/box.html#border-properties>; das CSS3-Modul *Backgrounds and Borders* unter <http://www.w3.org/TR/css3-background/#borders>; die CSS 2.1-Spezifikation zur Verwendung der Schlüsselwörter *thin*, *medium* und *thick* unter <http://www.w3.org/TR/CSS2/box.html#border-width-properties>; das Rezept 12.6 zur Erstellung von abgerundeten Ecken.

2.8 Hintergrundbilder verwenden

Problem

Sie möchten Elemente mit Hintergrundbildern versehen.

Lösung

Verwenden Sie die CSS-Eigenschaft `background-image`.

```
div {  
    background-image: url("bilder/hintergrund.jpg");  
    background-color: #ffffff;  
}
```

Die oben stehende Regel sorgt dafür, dass die Datei *hintergrund.jpg* als Hintergrundbild für `div`-Elemente verwendet wird.

Standardmäßig wird das Bild in der linken oberen Ecke des Elements platziert und von dort aus entlang der x-Achse wiederholt (horizontal gekachelt), zu sehen in Abbildung 2-6.

Für den Fall, dass das Hintergrundbild unter dem angegebenen Dateipfad nicht gefunden werden kann, geben wir in der zweiten Deklaration eine Hintergrundfarbe an.

Diskussion

Mit CSS können Sie Hintergrundbilder nicht nur für die gesamte Seite definieren, sondern auch für die meisten enthaltenen HTML-Elemente. Dadurch ist es möglich, jedem Element seinen eigenen Hintergrund zu geben. Hierbei gibt die Eigenschaft `background-image` an, welches Bild für den Hintergrund verwendet werden soll, wie hier gezeigt:

```
background-image: url("pfad/zur/datei.jpg");
```

oder auch hier:

```
background-image: url("http://www.csskochbuch.com/pfad/zur/datei.jpg");
```

Für die Definition des Dateipfads wird die `url()`-Schreibweise verwendet. Eine Beschreibung finden Sie in Rezept 2.4.

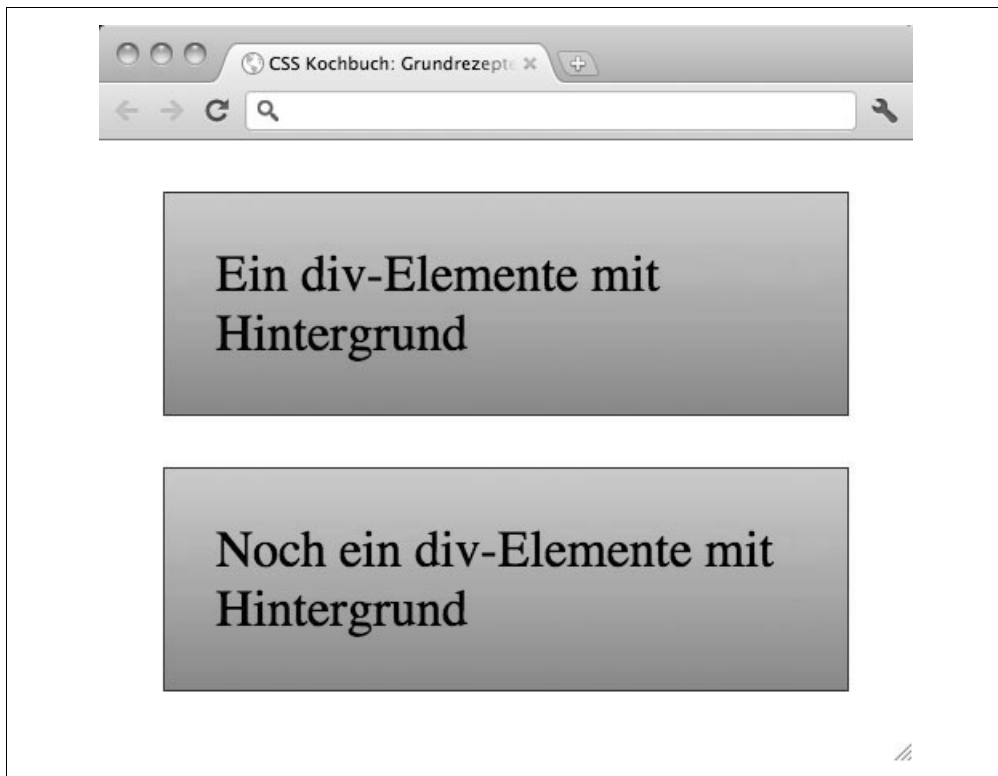


Abbildung 2-6: Für die beiden div-Elemente wurde ein Farbverlauf als Hintergrundbild definiert.

Dieses Rezept zeigt die einfachste Form, ein Hintergrundbild zu definieren. Dabei wird das Bild standardmäßig in der linken oberen Ecke des Elements (oder des Ansichtsberreichs) platziert und von dort aus horizontal und vertikal wiederholt, bis der gesamte Hintergrund ausgefüllt ist.

In den folgenden Rezepten zeigen wir Ihnen, wie Sie die einzelnen Aspekte der Darstellung von Hintergrundbildern an die eigenen Bedürfnisse anpassen können.

Siehe auch

Das Rezept 1.16 zum Boxmodell; das Rezept 2.1 zu den CSS-Längenangaben; Kapitel 9 über Farben und Hintergründe in *CSS – Das umfassende Handbuch* von Eric A. Meyer (O'Reilly 2007); die CSS 2.1-Spezifikation zum Thema Hintergründe unter <http://www.w3.org/TR/CSS2/colors.html#background>; die folgenden Rezepte zur Verwendung von Hintergrundbildern.

2.9 Die Wiederholung von Hintergrundbildern steuern

Problem

Sie wollen selbst steuern, auf welche Weise die von Ihnen definierten Hintergrundbilder wiederholt werden.

Lösung

Um ein Hintergrundbild horizontal, also entlang der *x*-Achse, zu wiederholen, können Sie die folgende CSS-Regel verwenden (siehe Abbildung 2-7):

```
body {  
    background-image: url("/bilder/hintergrund.jpg");  
    background-repeat: repeat-x;  
}
```

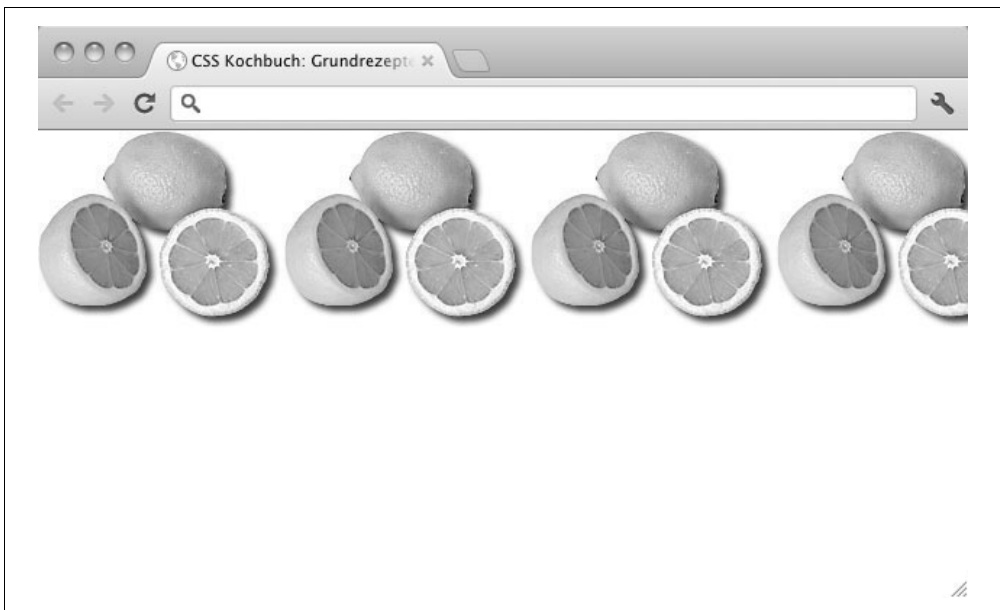


Abbildung 2-7: Das Hintergrundbild wird horizontal wiederholt

Diskussion

Mit der Eigenschaft `background-repeat` können Sie steuern, ob und wie das Bild wiederholt wird. Mögliche Werte sind: `repeat` (Standardwert, das Bild wird horizontal und vertikal wiederholt) und `repeat-x` (horizontale Wiederholung). Um das Hintergrundbild vertikal (entlang der *y*-Achse) zu wiederholen, verwenden Sie für die Eigenschaft `background-repeat` den Wert `repeat-y`. Soll das Hintergrundbild überhaupt nicht wiederholt werden, versehen Sie `background-repeat` mit dem Wert `none`.

Siehe auch

Das Rezept 2.10 zur Platzierung von Hintergrundbildern an einer bestimmten Position innerhalb der Webseite; die übrigen Rezepte zur Verwendung von Hintergrundbildern in diesem Kapitel.

2.10 Hintergrundbilder positionieren

Problem

Sie möchten ein Hintergrundbild an einer bestimmten Stelle auf der Webseite positionieren.

Lösung

Verwenden Sie die Eigenschaft `background-position`, um festzulegen, an welcher Stelle das Hintergrundbild platziert werden soll:

```
body {  
    background-image:    url("bilder/hintergrund.jpg");  
    background-repeat:   no-repeat;  
    background-position: 75px 50px  
}
```

Diese Regel sorgt beispielsweise dafür, dass das Hintergrundbild – von der linken oberen Ecke des Ansichtsbereichs aus gesehen – um 75 Pixel nach rechts und 50 Pixel nach unten verschoben dargestellt wird (siehe Abbildung 2-8).

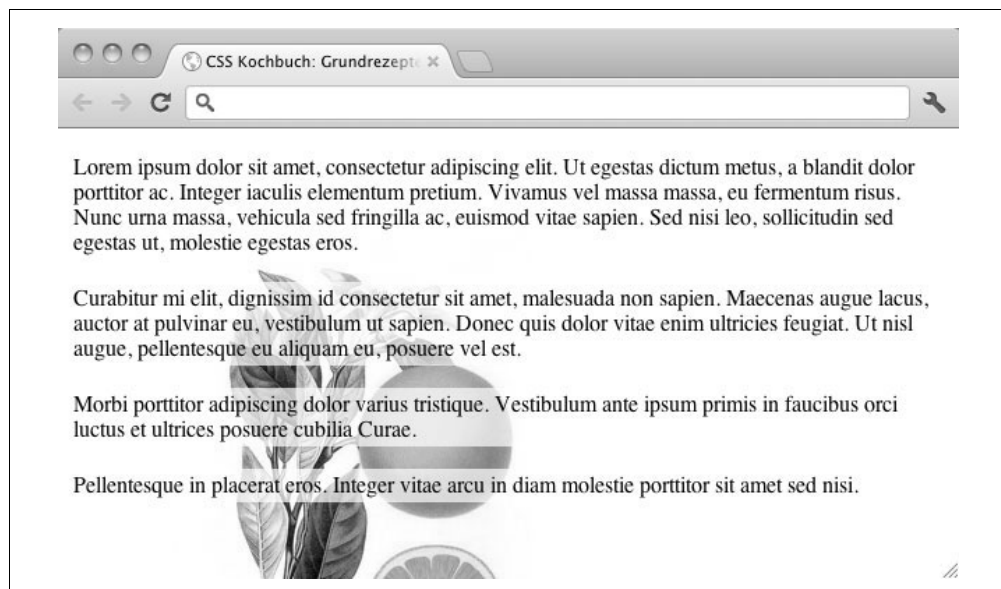


Abbildung 2-8: Das Hintergrundbild wird exakt 75 Pixel rechts und 50 Pixel unterhalb der linken oberen Ecke des Ansichtsbereichs platziert

Diskussion

Die Eigenschaft `background-position` legt fest, an welcher Stelle das Hintergrundbild platziert werden soll. Als Werte erhält `background-position` zwei durch ein Leerzeichen getrennte Werte. Der erste Wert des Paares gibt an, wie weit das Bild entlang der x-Achse verschoben werden soll, während der zweite Wert die Verschiebung entlang der y-Achse definiert. Als Referenzpunkt dient hierbei die linke obere Ecke des Bereichs, den Inhalt und eventuell definierte Innenabstände zusammen einnehmen (der sogenannten `padding-box`). Wird nur ein Wert angegeben, wird dieser für die horizontale Verschiebung verwendet, der Wert für die vertikale Verschiebung wird standardmäßig auf 50% gesetzt.

In der Lösung wurden als Maßeinheit Pixel verwendet, um die Platzierung des Hintergrundbilds festzulegen, Sie können aber auch Prozentwerte setzen. Erhält `background-position` den Wert 50%, platziert der Browser das Bild genau im Zentrum des Ansichtsbereichs, wie in Abbildung 2-9 zu sehen ist. Verwenden Sie die Werte 0% bzw. 100%, wird das Bild dagegen in der linken oberen bzw. unteren Ecke platziert.



Abbildung 2-9: Das Hintergrundbild wird im Browserfenster zentriert

Neben Pixeln und Prozentwerten können Sie auch die Schlüsselwörter `left`, `center` und `right` für die Position auf der x-Achse sowie `top`, `center` und `bottom` für die auf der y-Achse einsetzen. Durch die Kombination dieser Schlüsselwörter können Sie ein Hintergrundbild also an acht genau definierten Positionen im Ansichtsbereich platzieren (in den Ecken und in der Mitte dazwischen). Um beispielsweise den Wert 50% aus Abbildung 2-9 nachzustellen, können Sie auch schreiben:

```
body {
  background-image: url("bilder/hintergrund.jpg");
  background-repeat: no-repeat;
  background-position: center center;
}
```

Um das Hintergrundbild dagegen in der rechten unteren Ecke zu positionieren, verwenden Sie diese CSS-Regel:

```
body {
  background-image: url('bilder/hintergrund.jpg');
  background-repeat: no-repeat;
  background-position: bottom right;
}
```

Und noch etwas: Wurde das Hintergrundbild direkt für das body-Element definiert, befindet sich der Ausgangspunkt für die Werte von background-position in der linken oberen Ecke des Ansichtsbereichs. Das gilt auch, wenn für body Außenabstände definiert wurden.

Siehe auch

Das Rezept 2.11 zur Verankerung von Hintergrundbildern, die an ihrer Position bleiben, während der Rest der Seite gescrollt wird; die CSS 2.1-Spezifikation zur Eigenschaft background-position unter <http://www.w3.org/TR/CSS21/colors.html#propdef-background-position>; das Rezept 1.16 zum Boxmodell.

2.11 Das Hintergrundbild verankern

Problem

Das Hintergrundbild soll sichtbar bleiben, auch wenn der Seiteninhalt gescrollt wird, wie in Abbildung 2-10 gezeigt.

Lösung

Verwenden Sie die Eigenschaft background-attachment mit dem Wert fixed:

```
body {
  background-image: url("bilder/hintergrund.jpg");
  background-repeat: no-repeat;
  background-attachment: fixed;
}
```

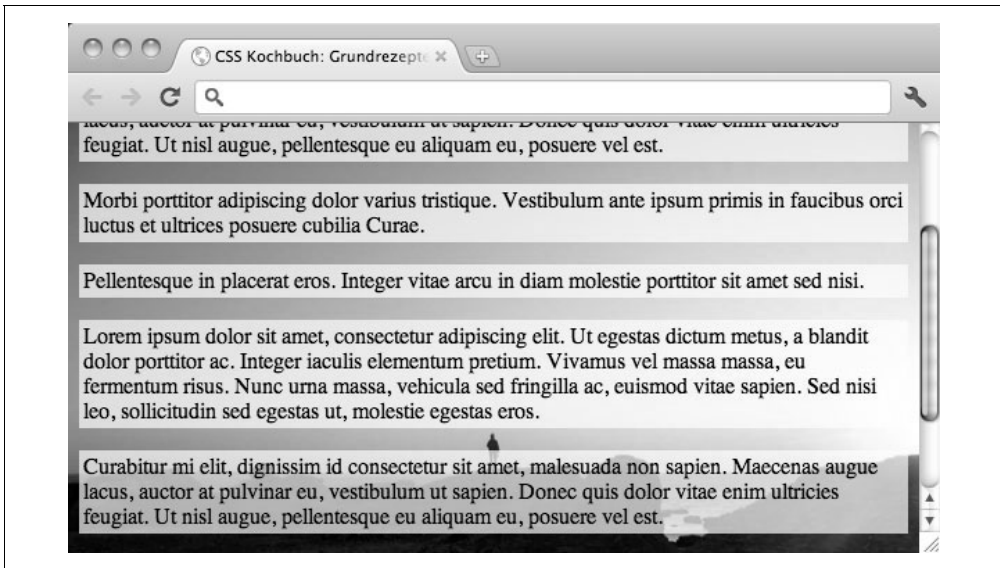


Abbildung 2-10: Auch beim Scrollen der Seite bleibt das Hintergrundbild im Ansichtsbereich verankert

Diskussion

Mithilfe dieser Technik können Sie das Hintergrundbild an einer bestimmten Position fest verankern. Selbst wenn der Besucher die Seite scrollt, bleibt das Bild dort im Browserfenster sichtbar, wo Sie es positioniert haben. Ein weiterer möglicher Wert für `background-attachment` ist der Standardwert `scroll`. Dies ist der Standardwert, d.h., selbst wenn Sie diesen Wert nicht ausdrücklich festlegen, wird das Bild mit dem übrigen Inhalt bewegt.

Siehe auch

Das Rezept 2.10 zur Positionierung von Hintergrundbildern; die CSS 2.1-Spezifikation zur Eigenschaft `background-attachment` unter <http://www.w3.org/TR/CSS21/colors.html#propdef-background-attachment>.

2.12 Mehrere Hintergrundbilder für ein Element definieren

Problem

Sie wollen ein Element mit mehr als einem Hintergrundbild versehen.

Lösung 1

Nach der CSS3-Spezifikation ist es möglich, mehrere Hintergrundbilder als Wert für die Eigenschaft `background-image` bzw. `background` zu definieren. Geben Sie die Werte für jedes Hintergrundbild durch Kommata getrennt an, wie hier:

```
body {
  background-image:    url("bilder/hintergrund_1.jpg"), url("bilder/hintergrund_2.jpg");
  background-position: top left, bottom right;
  background-repeat:   repeat-x, repeat-y;
}
```

Die oben stehende Regel versieht ein Element mit zwei Hintergrundbildern. Das erste (*hintergrund_1.jpg*) wird in der linken oberen Ecke platziert und horizontal wiederholt. Das zweite Bild (*hintergrund_2.jpg*) positionieren wir rechts unten und wiederholen es vertikal. Das Ergebnis sehen Sie in Abbildung 2-11.

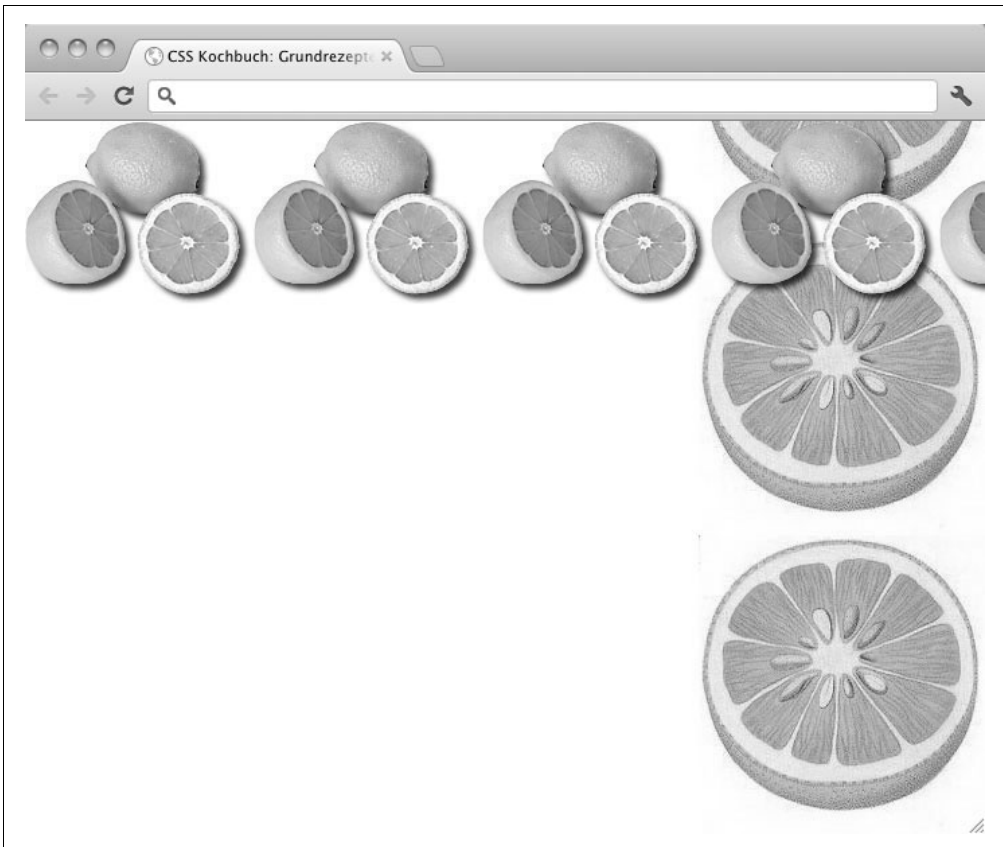
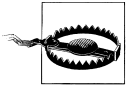


Abbildung 2-11: Zwei Hintergrundbilder werden für das gleiche Element definiert

Überlagern sich die Hintergrundbilder, steht das zuerst definierte am weitesten vorne. Danach definierte Bilder erscheinen dahinter. Aus diesem Grund überdeckt das horizontal wiederholte Hintergrundbild das danach definierte vertikale Bild.



Dieses Verfahren basiert auf dem Modul *Borders and Backgrounds* (Rahmen und Hintergründe) der CSS3-Spezifikation. Es hat zwar mittlerweile den Status einer »Candidate Recommendation« und wird von allen aktuellen Browsern unterstützt, der Internet Explorer unterstützt die Definition mehrerer Hintergrundbilder jedoch erst ab der Version 9.

Lösung 2

Definieren Sie innerhalb des Elements, das mehrere Hintergrundbilder bekommen soll, ein weiteres, leeres div-Element, dem Sie das zweite Hintergrundbild zuweisen:

```
div#erster_hintergrund {
    position: relative;
    background: url("bilder/hintergrund_1.jpg") top left repeat-x;
}

div#zweiter_hintergrund {
    position: absolute;
    top: 0;
    left: 0;
    height: 100%;
    width: 100%;
    z-index: -10;
    background: url("bilder/hintergrund_2.jpg") bottom right repeat-y;
}
```

Der HTML-Code könnte etwa so aussehen:

```
<div id="erster_hintergrund">
  <p>Lorem ipsum dolor sit amet...</p>
  <div id="zweiter_hintergrund"></div>
</div>
```

In dieser Lösung wird das div-Element `erster_hintergrund` als Positionierungskontext für das zweite div-Element definiert. Wir verwenden hier die Kurzschrifteigenschaft `background`, um das Hintergrundbild zu definieren.

Die Regeln für das zweite div-Element sind etwas komplexer. Zuerst legen wir fest, dass es innerhalb des ersten Elements absolut positioniert und an der linken oberen Ecke platziert werden soll. Durch die Angabe von `height` und `width` wird zudem dafür gesorgt, dass sich `zweiter_hintergrund` über die gesamte Fläche von `erster_hintergrund` erstreckt. Wir haben jetzt also zwei div-Elemente übereinandergelegt.

Da das zweite div-Element erst am Ende der Quellcodereihenfolge (hier: nach dem eigentlichen Inhalt) definiert wird, müssen wir es per `z-index` nach »hinten« schieben, ansonsten wäre der Inhalt von `erster_hintergrund` nicht mehr zugänglich. Nachdem die Vorarbeiten abgeschlossen sind, können wir schließlich das Hintergrundbild selbst definieren.

Diskussion

Bis vor Kurzem war die Definition mehrerer Hintergrundbilder nur möglich, wenn man, wie in Lösung 2 gezeigt, zusätzlichen Markup-Code verwendete. Nach und nach wächst aber die Unterstützung für CSS3. Dadurch werden leere `div`-Elemente und aufwendige CSS-Regeln bald nicht mehr gebraucht. Das erleichtert die Pflege der Seiten und der Stylesheets gleichermaßen.

Siehe auch

Die Rezepte 2.17 und 2.18 zur absoluten und relativen Positionierung; das Rezept 2.21 zur Verwendung von `z-index`; das *CSS3-Backgrounds and Borders*-Modul unter <http://www.w3.org/TR/css3-background/#backgrounds>.

2.13 Hintergrundbilder auf den gesamten Ansichtsbereich ausdehnen

Problem

Ein Bild soll sich über den gesamten sichtbaren Bereich des Browserfensters erstrecken, beispielsweise um es als Hintergrundbild einzusetzen (siehe Abbildung 2-12). Dies soll unabhängig von der Größe des Browserfensters funktionieren.

Lösung 1

Definieren Sie das Bild als Hintergrund für das `body`-Element. Verwenden Sie zusätzlich zu den üblichen `background`-Eigenschaften die CSS3-Eigenschaft `background-size` mit dem Wert `cover`.

```
body {
    background-image:    url("bilder/hintergrund.jpg");
    background-repeat:   no-repeat;

    -webkit-background-size: cover;
    -moz-background-size: cover;
    -o-background-size: cover;
    background-size: cover;
}
```

Im ersten Deklarationsblock definieren wir ein Hintergrundbild für das `body`-Element. Per `background-repeat: no-repeat` wird außerdem festgelegt, dass das Bild nicht wiederholt werden soll. Der entscheidende Schritt ist jedoch die Deklaration `background-size: cover`. Hiermit wird festgelegt, dass sich das Bild an die Größe des Ansichtsbereichs anpassen soll.

Zusätzlich definieren wir den Wert für einige ältere Browser in der jeweiligen Präfixschreibweise.

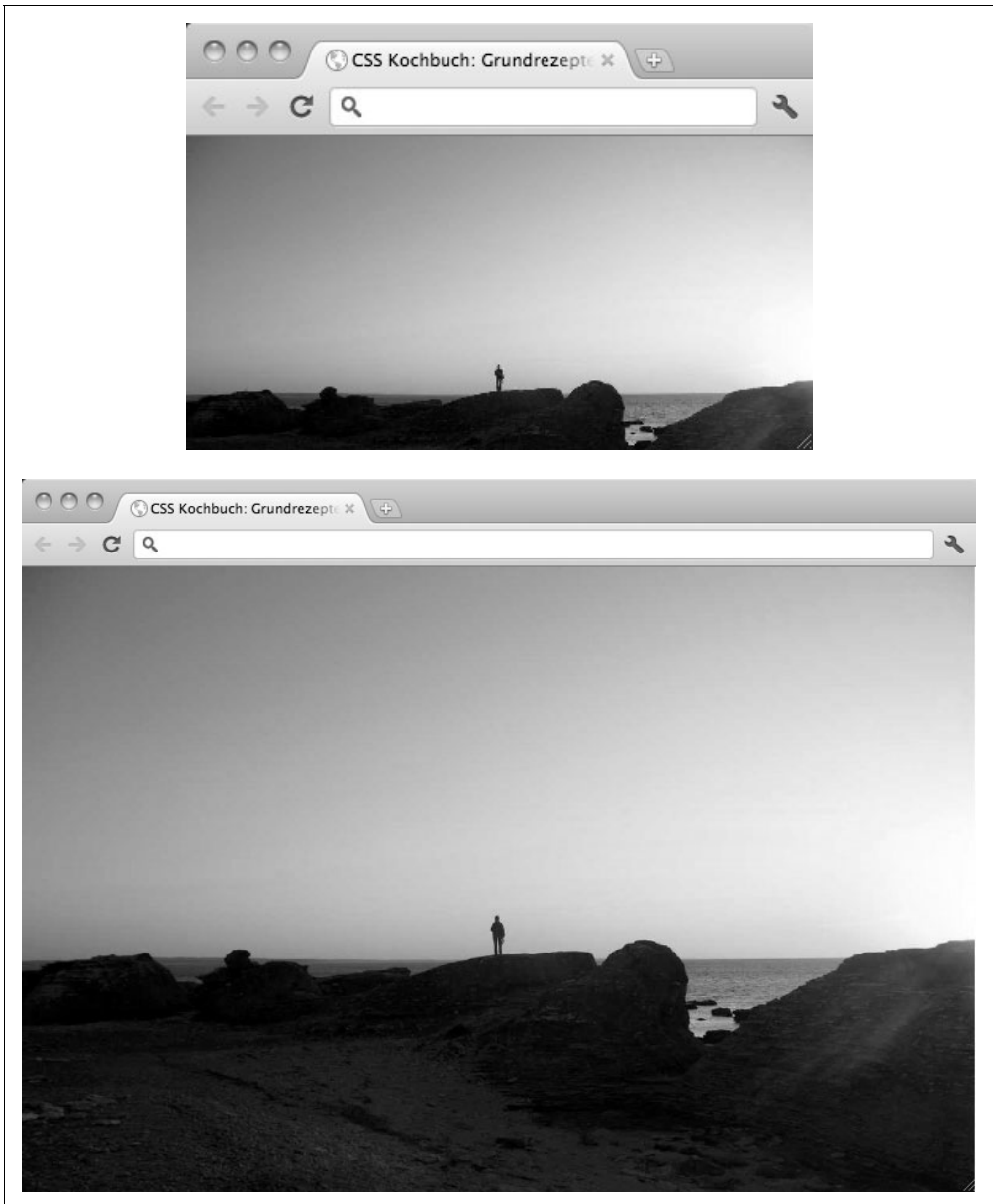


Abbildung 2-12: Das Hintergrundbild erstreckt sich über den gesamten Ansichtsbereich

Lösung 2

Definieren Sie das Bild als Kindelement des body-Elements. Verwenden Sie die absolute Positionierung, um das Bild vom normalen Dokumentenfluss auszunehmen. Breite und Höhe des Bilds geben Sie mit jeweils 100% an. Hier die CSS-Regeln:

```
body {
    margin: 0;
    padding: 0;
}

img#hintergrund {
    position: absolute;
    width: 100%;
    z-index: -10;
}
```

Der passende HTML-Code sieht etwa so aus:

```
<body>
...

</body>
```

Leider verstehen ältere Versionen des Internet Explorer (bis inklusive Version 8) die Definition von background-size noch nicht. Daher müssen wir hier eine Fallback-Lösung verwenden und ausnahmsweise ein zusätzliches Element in den HTML-Code einfügen, das das Bild einbindet. Dieses können Sie dann per CSS so anpassen, dass es sich über den gesamten Inhaltsbereich erstreckt.

Zunächst sorgen wir dafür, dass eventuell vom Browser definierte Außenabstände für das body-Element auf 0 gesetzt werden, damit der Hintergrund sich auch tatsächlich über das gesamte Browserfenster erstreckt.

Per position: absolute; wird dann das Bild vom normalen Dokumentenfluss ausgenommen. Danach wird festgelegt, dass die Breite (width) sich den Dimensionen des Inhaltsbereichs anpassen soll. Wir verzichten hier absichtlich auf eine Angabe von height, damit das Seitenverhältnis des Hintergrundbilds erhalten bleibt.

Die Angabe z-index: -10 sorgt dafür, dass das Bild mit sehr großer Wahrscheinlichkeit³ hinter den übrigen Inhalten der Seite angezeigt wird (mehr zur Eigenschaft z-index finden Sie in Rezept 2.21 in diesem Kapitel). Das funktioniert auch, wenn es erst am Schluss des HTML-Dokuments eingebunden wurde.

3 Wir schreiben hier absichtlich »mit großer Wahrscheinlichkeit«. In seltenen Fällen kann es sein, dass andere Elemente einen niedrigeren Wert für z-index haben. Dann besteht der einfachste Ausweg darin, für #hintergrund einen noch niedrigeren anzugeben.

Diskussion

Eigentlich werden Hintergründe nicht als Teil des Inhalts eines Dokuments angesehen. Daher sollten sie nach Möglichkeit auch nicht im HTML-Dokument selbst definiert werden, sondern im Stylesheet. Die zweite Lösung ist im Sinne der Trennung von Inhalt und Gestaltung also nicht ganz sauber. Andererseits funktioniert sie auch mit älteren Browsern.

Die CSS3-Eigenschaft `background-size` kennt vier verschiedene Werte:

contain

Das Bild wird so skaliert, dass es bei *kleinstmöglicher* Darstellung nach Möglichkeit den gesamten Bereich des Hintergrunds abdeckt. Das Verhältnis von Breite zu Höhe wird beibehalten, sodass das Bild nicht verzerrt erscheint.

cover

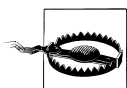
Das Bild wird so skaliert, dass es bei *größtmöglicher* Darstellung nach Möglichkeit den gesamten Bereich des Hintergrunds abdeckt. Das Verhältnis von Breite zu Höhe wird beibehalten, sodass das Bild nicht verzerrt erscheint.

Breite und/oder Höhe

Der erste Wert gibt die Breite des Hintergrundbilds an, der zweite die Höhe. Wurde nur ein Wert angegeben, wird für den zweiten Wert `auto` verwendet. Es sind prinzipiell alle in CSS gültigen Längeneinheiten (px, cm, em usw.) sowie Prozentwerte erlaubt. Prozentwerte beziehen sich auf die Größe des Hintergrundbereichs. Es dürfen keine negativen Werte angegeben werden.

auto

Höhe und/oder Breite basieren auf der tatsächlichen Größe und dem Seitenverhältnis des Hintergrundbilds. Das Bild wird hierbei nicht oder nur in einer Richtung skaliert. Dies ist der Standardwert.



Wenn Sie die Werte `contain` und `cover` verwenden, kann es passieren, dass Teile des Hintergrunds nicht vom Bild verdeckt werden, wenn `background-repeat` den Wert `no-repeat` hat. Die Beibehaltung des Seitenverhältnisses hat Vorrang.

Siehe auch

Die Seite des Mozilla Developer Network zur Eigenschaft `background-size` unter <https://developer.mozilla.org/en/CSS/background-size>; das *Borders and Backgrounds*-Modul der CSS3-Spezifikation zur Platzierung von Hintergrundbildern für das `body`-Element unter <http://www.w3.org/TR/css3-background/#special-backgrounds>; die CSS 2.1-Spezifikation zum Boxmodell unter <http://www.w3.org/TR/CSS2/box.html>; das Rezept 1.16.

2.14 Definitionen für Hintergründe in einer Deklaration zusammenfassen

Problem

Sie wollen die Angaben für den Hintergrund eines Elements möglichst effizient angeben.

Lösung

Verwenden Sie die Kurzschrifteigenschaft `background` anstelle der verschiedenen Einzeleigenschaften, z.B.:

```
body {  
    background: #fff url("bilder/hintergrund.jpg") 50% 50% no-repeat fixed;  
}
```

Das entspricht der folgenden Regel:

```
body {  
    background-color:    #fff;  
    background-image:    url("bilder/hintergrund.jpg");  
    background-position: 50% 50%;  
    background-repeat:   no-repeat;  
    background-attachment: fixed;  
}
```

Beide Regeln tun das Gleiche: Sie versehen die Seite mit einem weißen Hintergrund. Davor erscheint das unter *bilder/hintergrund.jpg* gespeicherte Hintergrundbild, das in der Mitte des Ansichtsbereichs zentriert wird.

Außerdem wird festgelegt, dass das Bild nicht wiederholt und beim Scrollen der Seite im Ansichtsbereich verankert werden soll.



Zwar unterstützen viele Browser bereits die neuen Hintergrundeneigenschaften, wie `background-size`. Allerdings mussten sie, während dieses Buch geschrieben wurde, teilweise noch in der Präfixschreibweise angegeben werden und funktionieren daher nicht zusammen mit der Kurzschrifteigenschaft `background`.

Diskussion

Für die Definition von Hintergründen gibt es auch eine Kurzschrifteigenschaft mit dem Namen `background`. Mit ihr können Sie die Werte der verschiedenen Einzeleigenschaften auf einmal definieren.

Hierbei spielt die Reihenfolge keine Rolle, allerdings müssen die Teilwerte für `background-position` direkt aufeinanderfolgen.

Siehe auch

Die CSS 2.1-Spezifikation zur Kurzschrifteigenschaft `background` unter <http://www.w3.org/TR/CSS2/colors.html#propdef-background>; das *Borders and Backgrounds*-Modul der CSS3-Spezifikation unter <http://www.w3.org/TR/css3-background/#the-background>.

2.15 Durchscheinende Hintergründe definieren

Problem

Sie wollen bestimmte Elemente mit einem durchscheinenden Hintergrund versehen.

Lösung 1

Verwenden Sie die CSS3-Funktion `rgba()`, um eine Hintergrundfarbe inklusive Transparenz anzugeben, z.B.:

```
p {  
    background-color: rgba(44%, 86%, 58%, 0.5);  
}
```

In dieser Regel erhalten alle Absätze einen hellgrünen, halb durchscheinenden Hintergrund.

Lösung 2

Verwenden Sie eine halb transparente PNG-Grafik als Hintergrundbild:

```
p {  
    background: aquamarine;  
    background: url("bilder/halbtransparent.png");  
}
```

In dieser Regel definieren wir zunächst einen schlichten hellgrünen Farbtton als Hintergrundfarbe für den Fall, dass das Bild nicht geladen werden kann. Danach geben wir als Hintergrundbild eine halb transparente PNG-Grafik an. Diese lässt sich mit wenig Aufwand in einem Bildbearbeitungsprogramm wie GIMP oder Photoshop erstellen.

Wir benutzen hier die Kurzschrifteigenschaft `background` zweimal hintereinander. Sofern das Hintergrundbild geladen werden kann, wird die mit `aquamarine` definierte Hintergrundfarbe wieder auf den Standardwert `transparent` zurückgesetzt.

Wenn wir stattdessen die Einzeleigenschaften `background-color` und `background-image` benutzen, erhalten wir bestenfalls eine halb transparente grüne PNG-Grafik vor einem grünen Hintergrund. Und das wollen wir doch nicht.

Diskussion

Durchscheinende Hintergründe sind ein beliebtes Gestaltungsmittel im Webdesign. Bis vor Kurzem konnte man hierfür nur, wie in Lösung 2 gezeigt, mit transparenten PNG-Dateien arbeiten.

Mittlerweile verstehen aber alle aktuellen Browser (außer IE7 und 8) auch die in CSS3 definierte `rgba()`-Schreibweise. Das Erstellen von PNG-Grafiken entfällt, weil Sie die Transparenz der Farbe direkt im Stylesheet angeben können.

Um für beide Fälle gerüstet zu sein, können Sie beide Lösungen auch miteinander kombinieren, indem Sie drei `background`-Deklarationen in der gleichen Regel verwenden:

```
.halbtransparent {  
    background: aquamarine;  
    background: url("bilder/halbtransparent.png");  
    background: rgba(44%, 86%, 58%,0.5);  
}
```

Diese Regel wirkt auf den ersten Blick nicht besonders intuitiv. Dreimal `background` hintereinander? Muss das sein?

Wie in Rezept 1.2 beschrieben, haben die Erfinder von CSS für Kurzschrifteigenschaften ein besonderes Verhalten definiert: Wurde für die möglichen Einzeleigenschaften (`background-color`, `background-image` usw.) kein ausdrücklicher Wert angegeben, werden sie automatisch auf ihren Standardwert zurückgesetzt (`transparent`, `none` usw.). Das machen wir uns, wie in Lösung 2 bereits angesprochen, zunutze.

Zu Beginn definieren wir wieder einen hellgrünen Hintergrund für unsere Klasse. Diese Deklaration wird verwendet, wenn die folgenden beiden nicht funktionieren.

Die nächste Deklaration versucht, das halb transparente Hintergrundbild zu laden. Aufgrund der Standardwerte wird es erwartungsgemäß so oft wiederholt, bis der gesamte Hintergrund des Elements ausgefüllt ist. Konnte das Bild aus irgendeinem Grund nicht geladen werden, greift die erste Deklaration.

Für Browser, die bereits mit `rgba()`-Werten umgehen können, gibt es die dritte Deklaration. Sie definiert ebenfalls eine hellgrüne Hintergrundfarbe, der dieses Mal jedoch zu 50% durchscheinend ist. Das funktioniert, weil die Kurzschrifteigenschaft `background` den zuvor definierten Wert für `background-image` wieder auf seinen Standardwert (`none`) zurücksetzt.



Einige Browser (Chrome 7+, Safari 5.1+, Opera 11.10+, Firefox 3.6+) sind so schlau, das in der zweiten Deklaration angegebene Bild nicht zu laden, wenn sie den Hintergrund auch mit der `rgba()`-Schreibweise erzeugen können.

Ältere Browser, die die `rgba()`-Notation noch nicht verstehen, werden diese Deklaration als ungültig betrachten und das PNG-Bild bzw. einen einfachen hellgrünen Hintergrund ohne Transparenz verwenden.

Das Schöne an dieser Lösung ist ihre Schlichtheit und die eingebaute Rückwärtskompatibilität – sie funktioniert in allen aktuellen Browsern inklusive Internet Explorer 7 und 8.

Siehe auch

Das Rezept 2.3 zur Definition von Farben; die CSS 2.1-Spezifikation zum Verhalten von Kurzschrifteigenschaften unter <http://www.w3.org/TR/CSS21/about.html#shorthand>; einen Artikel von Chris Coyier mit Informationen zum Laden von Fallback-Grafiken unter <http://css-tricks.com/css3-gradients/>.

2.16 Elemente positionieren

Problem

Sie wollen ein Element nicht an der vom Dokumentenfluss vorgegebenen Position platzieren. Stattdessen soll es an anderer Stelle im Ansichtsbereich dargestellt werden.

Lösung

Geben Sie der Eigenschaft `position` einen anderen Wert als `static` und geben Sie die Position mithilfe der Eigenschaften `top`, `right`, `bottom` und `left` an. Welcher Wert für `position` infrage kommt, hängt davon ab, wie das Element positioniert werden soll. Möglich sind: `absolute`, `fixed` und `relative`. Auf die Unterschiede zwischen diesen Einstellungen wird in den folgenden Rezepten genauer eingegangen. Hier ein kurzes Beispiel:

```
#immerda {  
    position: fixed;  
    top:    0;  
    left:   0;  
}
```

Diese Regel verankert das Element mit der ID `immerda` in der linken oberen Ecke des Browserfensters.

Diskussion

Standardmäßig werden die Elemente einer HTML-Seite gemäß ihrer Reihenfolge im Quelldokument angeordnet (dem sogenannten *Dokumentenfluss*). Hier gilt: »Wer zuerst kommt, mahlt zuerst.« Manchmal ist es aber sinnvoll oder sogar notwendig, diese Reihenfolge bei der Darstellung zu verändern, beispielsweise um mehrspaltige Layouts zu erstellen. Genau hier kommt die Eigenschaft `position` ins Spiel. Standardmäßig hat sie für alle Elemente den Wert `static`. Das heißt, das Element wird gemäß seiner Position im Dokumentenfluss dargestellt.

Hierbei nehmen die Werte `absolute` und `fixed` das Element komplett aus dem Dokumentenfluss heraus und stellen es an der angegebenen Position dar. Wird dagegen der Wert `relative` verwendet, werden die Positionsangaben relativ zur ursprünglichen Platzierung im Dokumentenfluss interpretiert. Wie in der Lösung bereits gesagt, werden für die eigentliche Positionsangabe die Eigenschaften `top`, `left`, `bottom` und `right` benutzt. Sie geben an, wie weit das positionierte Element vom jeweiligen Rand (oben, links, unten, rechts) entfernt sein soll.

Ohne weitere Angaben ist dies der Rand des Ansichtsbereichs.⁴ Für absolut positionierte Elemente lässt sich das jedoch ändern, wenn diese in andere Elemente verschachtelt sind, die einen anderen Wert für `position` haben als `static`. Hierdurch ist es möglich, Elemente auch ineinander zu positionieren. Dann beziehen sich die Positionsangaben auf den nächsten umgebenden Block, der einen Positionierungskontext bereitstellt. Als Werte können alle in CSS gültigen Längeneinheiten verwendet werden; auch negative Werte sind möglich.

Siehe auch

Die folgenden Rezepte zur Verwendung der Eigenschaft `position`; die CSS 2.1-Spezifikation zum Thema Positionierung unter <http://www.w3.org/TR/CSS2/visuren.html>.

2.17 Elemente absolut positionieren

Problem

Sie wollen ein Element statt an seiner Standardposition frei im Dokument platzieren.

Lösung

Weisen Sie im Stylesheet der Eigenschaft `position` für das betreffende Element den Wert `absolute` zu. Mithilfe der Eigenschaften `top`, `right`, `bottom` und `left` können Sie angeben, an welcher Stelle das Element positioniert werden soll. Das Ergebnis sehen Sie in Abbildung 2-13.

```
#element {  
    position: absolute;  
    top:      50px;  
    left:     100px;  
}
```

⁴ Die Ausnahme tritt auf, wenn zwei Elemente ineinander verschachtelt sind, von denen das äußere mit der CSS3-Eigenschaft `transform` verändert wurde. In diesem Fall gilt als Bezugspunkt der Rand des transformierten Elements. Das innere Element verhält sich in diesem Fall, als wäre es im umgebenden Element absolut positioniert worden.

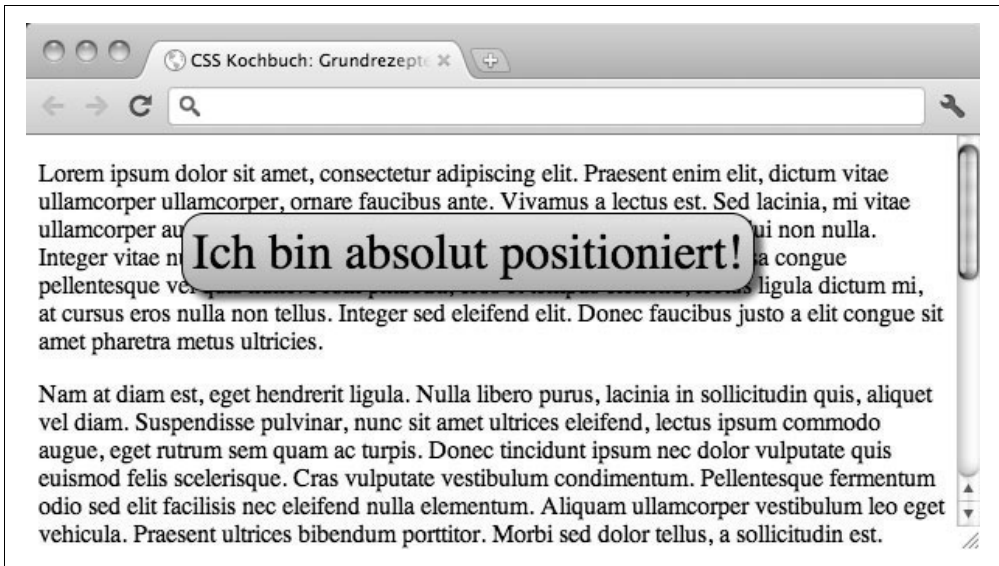
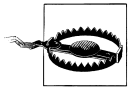


Abbildung 2-13: Die absolute Positionierung platziert ein Element unabhängig vom übrigen Dokumentenfluss



Bei dieser Technik gibt es eine Einschränkung: Ist das Element von einem weiteren absolut, relativ oder fest positionierten Element umgeben, gilt dieses als Bezugspunkt für die Positionierung und nicht der Ansichtsbe- reich des Browsers.

Diskussion

Durch den Wert `absolute` für die Eigenschaft `position` wird das Element aus dem norma- len Textfluss des Dokuments herausgenommen und anhand weiterer Angaben an einer bestimmten Stelle im umgebenden Block platziert.⁵ Der Beispielcode unserer Lösung weist den Browser an, das `#element` genau 50 Pixel unterhalb des oberen Rands (`top`) und 100 Pixel vom linken Rand (`left`) des umgebenden Blocks entfernt darzustellen.

Der Einfachheit halber gehen wir in diesem Beispiel davon aus, dass der umgebende Block dem Browserfenster entspricht. Wie in Rezept 2.19 gezeigt wird, lässt sich aber auch ein anderer umgebender Block definieren. Dadurch ist es möglich, Elemente auch vor- oder hintereinander zu positionieren.

⁵ Näheres zur Positionierung von Elementen finden Sie in *CSS – Das umfassende Handbuch* von Eric A. Meyer (O'Reilly 2007) in Kapitel 10.

Siehe auch

Die W3C-2.1-Spezifikation zum Thema absolute Positionierung unter <http://www.w3.org/TR/CSS21/visuren.html#absolute-positioning>; die übrigen Rezepte zur Positionierung in diesem Kapitel.

2.18 Elemente relativ positionieren

Problem

Sie wollen ein Element abhängig von seiner ursprünglichen Position im Dokument platzieren. Anstatt ein Element an seiner Standardposition im Textfluss darzustellen, möchten Sie es um einen bestimmten Wert nach oben, unten, links oder rechts verschieben.

Lösung

Wie im vorigen Rezept verwenden Sie auch hier die Eigenschaft `position`. Diesmal benutzen Sie allerdings den Wert `relative`. Danach können Sie anhand der Werte für die Eigenschaften `top`, `right`, `bottom` und `left` festlegen, wie weit das Element von seiner ursprünglichen Position verschoben werden soll (siehe Abbildung 2-14):

```
span {  
  position: relative;  
  top:      10px;  
  left:     20px;  
}
```

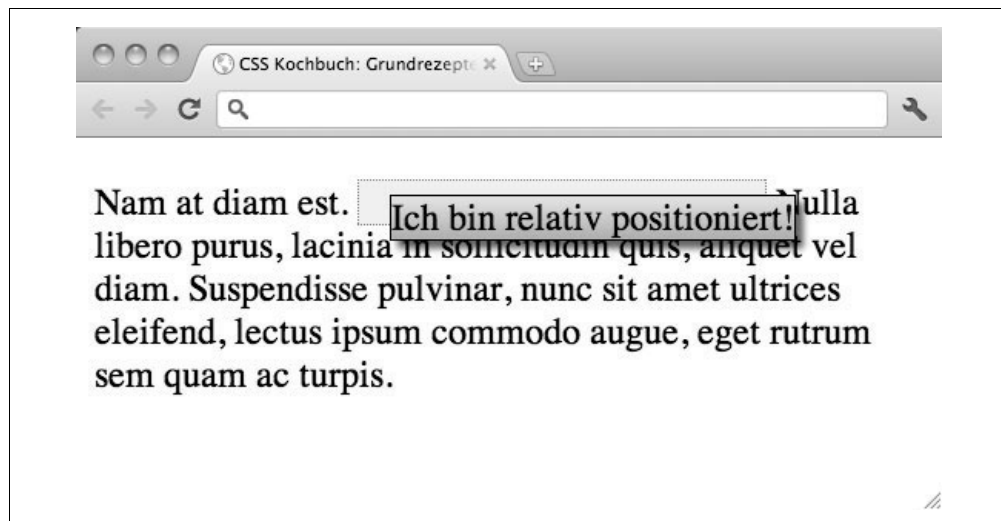


Abbildung 2-14: Die relative Positionierung platziert ein Element in Abhängigkeit von seiner ursprünglichen Position im Dokumentenfluss

Diskussion

Im Gegensatz zur absoluten Positionierung wird das betreffende Element bei relativer Positionierung nicht vom normalen Dokumentenfluss ausgenommen. Stattdessen wird als Ausgangspunkt die tatsächliche Platzierung im Textfluss benutzt.

Der Beispielcode weist den Browser an, das `span`-Element von seiner Ursprungsposition um 20 Pixel nach rechts und um 10 Pixel nach unten zu verschieben. Hätten Sie den Wert `absolute` verwendet, würde das Element stattdessen anhand der Ränder des umgebenden Blocks ausgerichtet werden, ohne dabei auf seinen ursprünglichen Platz Rücksicht zu nehmen.

Die relative Positionierung wird häufig eingesetzt, um einen neuen Positionierungskontext zu schaffen. Weitere Informationen hierzu finden Sie in Rezept 2.19.

Siehe auch

Die W3C-2.1-Spezifikation zum Thema relative Positionierung unter <http://www.w3.org/TR/CSS21/visuren.html#relative-positioning>; den Artikel zur Positionierung im Mozilla Developer Network unter <https://developer.mozilla.org/de/CSS/position>.

2.19 Elemente ineinander positionieren

Problem

Sie wollen die Position eines Elements nicht anhand des Browserfensters definieren. Stattdessen soll als Positionierungskontext der umgebende Block verwendet werden.

Lösung

Weisen Sie dem umgebenden Element eine relative Positionierung zu, ohne die Eigenschaften `top`, `left`, `bottom` oder `right` zu verwenden. Danach geben Sie dem enthaltenen Element für die Positionierung den Wert `absolute`. Jetzt bezieht sich die Position des inneren Elements nicht mehr auf das Browserfenster, sondern auf das umgebende Element.

Um beispielsweise ein Bild innerhalb eines `div`-Elements absolut zu positionieren, können Sie den folgenden Code verwenden:

```
div {  
    position: relative;  
}  
  
div > img {  
    position: absolute;  
    top: -1.1em;  
    left: -1.2em;  
}
```

Hier der passende HTML-Code:

```
<div>
  <p>Kennst Du das Land, wo die Zitronen blühen?</p>
  
</div>
```

Das Bild wird als Kindelement des `div`-Elements definiert. Danach weisen wir ihm im Stylesheet für die Eigenschaft `position` den Wert `relative` zu. Der neu entstandene Positionierungskontext sorgt dafür, dass wir das Bild nun frei im `div`-Element platzieren können, wie in Abbildung 2-15 gezeigt.



Abbildung 2-15: Das Bild verwendet den `div`-Container als Positionierungskontext

Anhand der Eigenschaften `top` und `left` haben wir das Bild an die gewünschte Position verschoben. Durch die negativen Werte erscheint es hierbei ein wenig außerhalb seines umgebenden Elements.

Diskussion

Prinzipiell gilt als *Positionierungskontext* der nächste umgebende Block, dessen Wert für `position` nicht den Standardwert `static` besitzt. Sofern Sie also keine weiteren Angaben machen, ist dies der Ansichtsbereich des Browsers. Das können Sie ändern, indem Sie für bestimmte Elemente einen eigenen Wert für `position` festlegen. Normalerweise verwendet man hier den Wert `relative`. Es ist aber durchaus möglich, ein Element in einem anderen absolut positionierten Element zu platzieren.

Um einen neuen Positionierungskontext zu schaffen, reicht es aus, einen entsprechenden Wert für `position` festzulegen. Die Eigenschaften `top`, `left`, `bottom` und `right` werden hierfür nicht benötigt.

Siehe auch

Die CSS 2.1-Spezifikation zum Thema Positionierung unter <http://www.w3.org/TR/CSS2/visuren.html#positioning-scheme>.

2.20 Elemente im Ansichtsbereich verankern

Problem

Sie wollen ein Element im Ansichtsbereich verankern. Es soll beim Scrollen der Seite nicht mitbewegt werden.

Lösung

Um zu verhindern, dass ein Element beim Scrollen der Seite aus dem Ansichtsbereich verschwindet, weisen Sie der Eigenschaft `position` den Wert `fixed` zu. Mithilfe der Eigenschaften `top`, `right`, `bottom` und `left` können Sie dann die Position angeben:

```
#fussteil {  
    position: fixed;  
    bottom: 20px;  
    right: 20px;  
}
```

Die obige Regel legt für das Element mit der ID `fussteil` fest, dass es fest im Ansichtsbereich verankert werden soll. Der Abstand zur Unterkante sowie zur rechten Seite soll jeweils 20 Pixel betragen.

Diskussion

Die feste Positionierung folgt den gleichen Regeln wie die absolute Positionierung. Der einzige Unterschied besteht darin, dass Elemente, deren Wert für `position` mit `fixed` angegeben wurde, beim Scrollen der Seite nicht mitbewegt werden. Auf diese Weise bleiben Teile einer Seite, z.B. der Navigationsbereich oder ein Fußteil, immer sichtbar.



Wurde für das umgebende Element mittels der CSS3-Eigenschaft `transform` ein anderer Wert als `none` definiert, bezieht sich die feste Positionierung nicht mehr auf die gesamte Seite, sondern auf das umgebende Element. Details finden Sie in Rezept 12.9.

Da sich der restliche Teil des Dokuments verschiebt, kann es passieren, dass ein fest positioniertes Element andere wichtige Teile der Seite überlagert.

Siehe auch

Die anderen Rezepte zum Thema Positionierung in diesem Kapitel; die CSS 2.1-Spezifikation dazu unter <http://www.w3.org/TR/CSS2/visuren.html#positioning-scheme>.

2.21 Elemente voreinander anordnen

Problem

Sie wollen steuern, in welcher Reihenfolge Elemente *voreinander* angeordnet werden.

Lösung

Geben Sie der Eigenschaft `position` für das gewünschte Element einen anderen Wert als `static` und legen Sie mittels der Eigenschaft `z-index` fest, wie weit vor oder hinter den übrigen Elementen es dargestellt werden soll, z.B.:

```
div#vorne {  
  position: absolute;  
  top: 0;  
  left: 0;  
  z-index: 10;  
}
```

Mit dieser Regel wird das `div`-Element mit der ID `vorne` in der linken oberen Ecke des Ansichtsbereichs platziert. Magisch wird es in der Deklaration `z-index: 10`. Je größer die Zahl, desto weiter vorn wird das Element dargestellt, je kleiner der Wert, desto weiter hinten. Im Prinzip können Sie hier einen beliebigen Wert einsetzen. Damit wir nicht zu viele Änderungen vornehmen müssen, wenn sich die Darstellung ändert (beispielsweise wenn zusätzliche `div`-Elemente positioniert werden sollen), geben wir den Wert hier in Zehnerschritten an.

Das funktioniert natürlich nur, sofern es keine anderen Elemente gibt, für die Sie einen höheren Wert für `z-index` definiert haben. Es empfiehlt sich, die Intervalle der Werte nicht zu eng zu wählen. Ansonsten kann es später passieren, dass ein neues Element in den HTML-Code eingefügt wird und Sie sämtliche `z-index`-Werte der bestehenden Elemente anpassen müssen. In den meisten Fällen sollten Sie mit Zehnerschritten (`z-index: 10`, `z-index: 20` usw.) auskommen.

Diskussion

Manchmal kommt es vor, dass Elemente sich z.B. durch absolute Positionierung überschneiden. Hierbei gilt: Je später im Quelltext der Seite das Element definiert wurde, desto weiter »vorn« wird es angezeigt. Manchmal soll diese sogenannte *Stapelungsreihenfolge* jedoch anders aussehen. Vielleicht soll ein am Anfang des Dokuments definiertes Element ganz vorne stehen.

Genau für diesen Zweck gibt es in CSS neben der x- und der y-Achse (horizontal bzw. vertikal) auch eine z-Achse und die entsprechende Eigenschaft `z-index`. Mit ihr kann gesteuert werden, wie Block- und ersetzte Elemente (siehe Rezept 1.17) vor- bzw. hintereinanderplatziert werden.

Voraussetzung ist, dass die Eigenschaft `position` einen anderen Wert hat als `static`. Als Wert kann eine beliebige ganze Zahl angegeben werden. Haben zwei Elemente den gleichen Wert für `z-index`, gewinnt das im Seitenquelltext zuletzt definierte. Um ein Element auf der z-Achse weiter »nach hinten« zu verschieben, können auch negative Werte angegeben werden (siehe Abbildung 2-16).

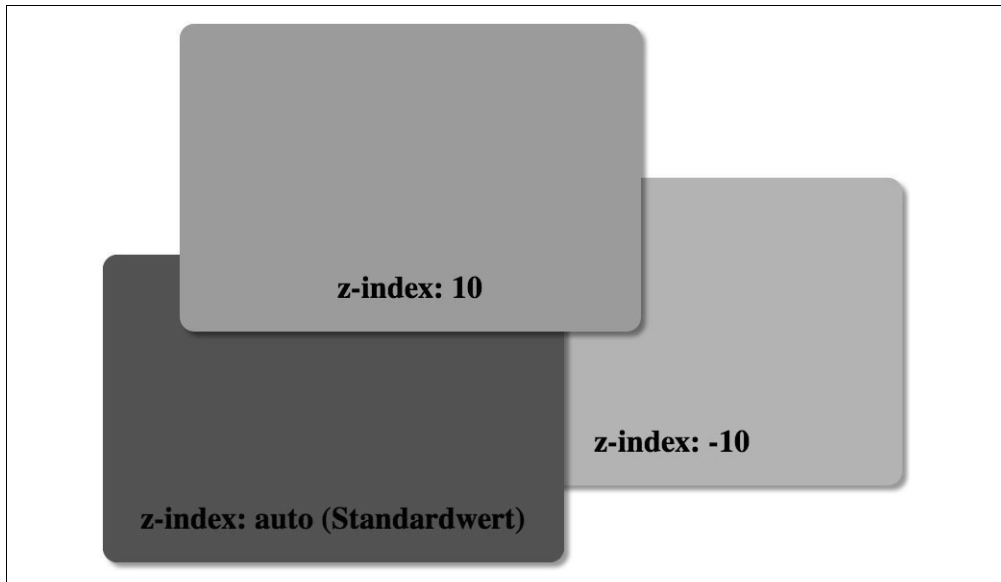


Abbildung 2-16: Die Funktionsweise von `z-index`



Der Internet Explorer 7 hält sich bei `z-index` nicht ganz an die Spielregeln. Um Fehler zu vermeiden, müssen Sie möglicherweise für das Elternelement ebenfalls einen Wert für `z-index` definieren. Dieser muss höher sein als der für das Element selbst. Einen Verweis zu einer Beschreibung der Situation finden Sie im Abschnitt »Siehe auch«.

Stapelungskontext

Sobald Sie einen ausdrücklichen Wert für `z-index` vergeben (der nicht dem Standardwert `auto` entspricht), wird für das betroffene Element ein eigener Stapelungskontext geschaffen. Nachfahren dieses Elements erhalten dadurch ihre eigene Stapelungsreihenfolge.

Siehe auch

Die CSS 2.1-Spezifikation zur Eigenschaft `z-index` unter <http://www.w3.org/TR/CSS2/visuren.html#z-index>; den Artikel *Squish the Internet Explorer z-index Bug* unter <http://brenelz.com/blog/squish-the-internet-explorer-z-index-bug/>; die Rezepte 6.11 und 6.12 zur Erstellung von Navigationsmenüs mit mehreren Ebenen.

2.22 Elemente scrollbar machen

Problem

Sie haben ein Element, dessen Inhalt scrollbar sein soll.

Lösung

Geben Sie dem Element eine feste Höhe und verwenden Sie dann die Eigenschaft `overflow-y` mit dem Wert `auto`, wie hier gezeigt:

```
#hauptteil {  
    height: 600px;  
    overflow-y: auto;  
}
```

Ein passendes HTML-Element könnte beispielsweise so aussehen:

```
<div id="hauptteil">  
    <p>Lorem ipsum larifari bla bla bumsdieldo...</p>  
</div>
```

In diesem Beispiel wurde dem `div`-Element eine feste Höhe von 600 Pixeln zugewiesen. Da die Höhe des darin enthaltenen Absatzes nicht bekannt ist, kann es passieren, dass der `div`-Container »überläuft«, d.h. sich über sein Elternelement hinaus erstreckt. Durch die Definition von `overflow-y: auto` wird dieses Überlaufen verhindert. Stattdessen wird der nicht in den Container passende Teil versteckt und gegebenenfalls über Scrollbalken zugänglich gemacht, wie in Abbildung 2-17 zu sehen ist.

Diskussion

Mit den Eigenschaften `overflow-x` bzw. `overflow-y` können Sie steuern, was passiert, wenn ein Element für seinen Inhalt zu klein wird. Standardmäßig haben die Eigenschaften den Wert `visible`. Ist der Inhalt höher oder breiter als das umgebende Element, ragt der Inhalt möglicherweise in das Elternelement hinein. Das kann zu unschönen Effekten im Layout führen, beispielsweise wenn der Benutzer die Schriftgröße verändert oder ein Blog-Posting eine Menge Kommentare erhält.



Abbildung 2-17: Ein Element mit fester Höhe; der Inhalt wird durch die Definition von `overflow: auto` gegebenenfalls durch Scrollbalken zugänglich gemacht

Browser erzeugen von sich aus Scrollbalken, wenn der Inhalt einer Webseite nicht ins Browserfenster passt. Genau dieses Verhalten können wir mit den Eigenschaften `overflow-x` und `overflow-y` nutzen. Hierbei sind folgende Werte möglich: `visible` (Standardwert, Inhalt ragt gegebenenfalls über das Element hinaus), `hidden` (der Inhalt wird gegebenenfalls teilweise verborgen, es werden keine Scrollbalken erzeugt), `scroll` (es werden Scrollbalken erzeugt, auch wenn der Inhalt in das umgebende Element passt) und schließlich `auto` (zu viel Inhalt wird verborgen und über Scrollbalken zugänglich gemacht).

Möchten Sie das Verhalten für beide Scrollbalken gemeinsam steuern, können Sie auch die Kurzschrifteigenschaft `overflow` nutzen. Die Werte sind identisch.

Siehe auch

Die Rezepte zum Thema Positionierung in diesem Kapitel; die CSS-Spezifikationen zur Eigenschaft `overflow` (sowie `overflow-x` und `overflow-y`) unter <http://www.w3.org/TR/CSS21/visufx.html> bzw. <http://www.w3.org/TR/css3-box/#overflow>; das Mozilla Developer Network zu den Eigenschaften `overflow-x` und `overflow-y` unter <https://developer.mozilla.org/en/CSS/overflow-x>.

2.23 Elemente umfließen lassen

Problem

Sie wollen Bilder und andere Elemente vom Dokumentenfluss ausnehmen und links bzw. rechts vom übrigen Inhalt »umfließen« lassen, wie in Abbildung 2-18 gezeigt.



Abbildung 2-18: Als Float definierte Elemente werden links oder rechts vom übrigen Inhalt umflossen

Lösung

Definieren Sie die gewünschten Elemente als *Floats*. Da die Regeln im Dokument oft mehr als einmal verwendet werden, empfiehlt sich hier die Verwendung von Klassen-selektoren (symbolisiert durch den vorangestellten Punkt):

```
.float_links {  
    float: left;  
}
```



```
.float_rechts {
    float: right;
}
```

Danach können Sie den Elementen im HTML-Code die Klassen zuweisen:

```

<p>Lorem ipsum dolor sit amet...</p>


<p>Aliquam pretium vulputate luctus...</p>

... noch mehr Text ...
```

In diesem Beispiel verwenden wir die im Stylesheet definierten Klassen, um das Beispielfeld einmal nach links und einmal nach rechts zu verschieben und vom folgenden Text umfließen zu lassen. Ohne die Definition als Floats würden die Bilder gemäß ihrer Position im Quelltext angezeigt.



Selbstverständlich können Sie anstelle von Klassen auch andere Selektoren zum Auswählen der Elemente verwenden und sich so möglicherweise einige class-Attribute im HTML-Code sparen.

Diskussion

Bevor sich die Browser an den CSS-Standard hielten, haben Designer das `img`-Tag oftmals mit dem Attribut `align` versehen, um Bilder nach links oder rechts zu verschieben und von Text umfließen zu lassen. Heutzutage verwendet man standardmäßig die CSS-Eigenschaft `float`.

Die Definition als Float funktioniert nicht nur mit Bildern. Sie können auf diese Weise im Prinzip alle Elemente aus dem normalen Textfluss nehmen und nach links oder rechts verschieben. Auf diese Weise lassen sich beispielsweise mehrspaltige Layouts (siehe Rezept 11.3) und speziell gestaltete Initialen (siehe Rezept 3.18) realisieren.

Das Umfließen von Elementen verhindern

In manchen Fällen muss das Umfließen präziser gesteuert werden, wie hier: In Abbildung 2-18 überschneidet sich das zweite Bild mit dem Absatz, der sich auf das erste Bild bezieht. Um dieses Verhalten zu umgehen, können Sie die Eigenschaft `clear` verwenden:

```
p {
    clear: left;
}
```

Die Ergänzung zu `float` ist die Eigenschaft `clear`. Mit `clear` legen Sie fest, auf welcher Seite eines Elements kein Float stehen darf. Anstatt auf gleicher Höhe mit dem zweiten `p`-Element dargestellt zu werden, wird das zweite Bild in unserem Beispiel so weit nach unten verschoben, dass es keine Überschneidungen mehr gibt (siehe Abbildung 2-19).

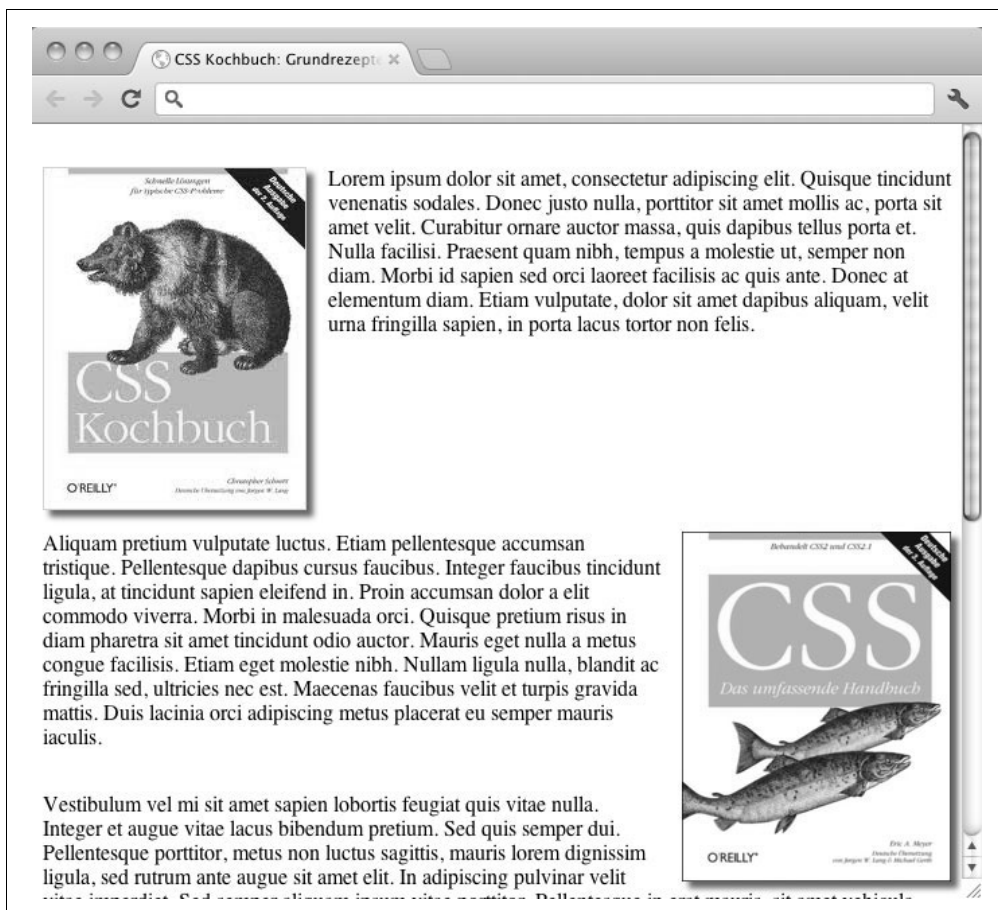


Abbildung 2-19: Mit der Eigenschaft `clear` wird das Umfließen von Elementen verhindert

Die Eigenschaft `clear` kennt vier verschiedene Werte:

- `left`: Links angeordnete Floats sollen nicht umfließen werden.
- `right`: Rechts angeordnete Floats sollen nicht umfließen werden.
- `both`: Floats sollen, unabhängig von ihrer Anordnung, nicht umfließen werden.
- `none`: Standardwert. Floats werden umfließen.

Siehe auch

Die W3C-2.1-Spezifikation zum Thema Floats unter <http://www.w3.org/TR/CSS21/visuren.html#floats>; die Rezepte in Kapitel 12, *Design mit CSS*, zur Verwendung von `float` für ein mehrspaltige Seitenlayouts; das Rezept 13.9.

2.24 Elemente horizontal zentrieren

Problem

Sie wollen ein Element horizontal zentrieren

Lösung 1: automatische seitliche Außenabstände

Handelt es sich bei dem Element um ein Blockelement (z.B. `<p>` oder `<div>`), geben Sie den Eigenschaften `margin-left`, bzw. `margin-right` den Wert `auto`. Hier setzen wir die Werte anhand der Kurzschrifteigenschaft `margin` für beide Seiten gemeinsam:

```
#nav_container {  
    width: 836px;  
    margin: auto;  
}
```

Durch diese Anweisung werden die seitlichen Außenabstände des `div`-Elements mit der ID `nav_container` vom Browser automatisch so berechnet, dass sie auf beiden Seiten gleich groß sind. Auf diese Weise erscheint das Element in der Mitte des umgebenden Elements (oder des Ansichtsbereichs) zentriert, wie in Abbildung 2-20 gezeigt.



Abbildung 2-20: Ein Element wird horizontal zentriert



Standardmäßig erstrecken sich Blockelemente über die gesamte Breite des Ansichtsbereichs/umgebenden Blocks. Daher ist es wichtig, dem Element eine bestimmte Breite zuzuweisen, damit der Effekt sichtbar wird.

Lösung 2: Prozentwerte

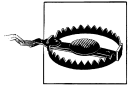
Die Methode aus Lösung 1 wird bei Weitem am häufigsten verwendet. Eine andere Möglichkeit für die Zentrierung von Blockelementen ist die Verwendung von Prozentwerten, zum Beispiel:

```
div#navigation {  
    width: 60%;  
    position: absolute;  
    left: 20%;  
}
```

Bei diesem Verfahren wird das Element absolut positioniert (siehe Rezept 2.17). Um den nötigen seitlichen Abstand zu ermitteln, subtrahieren Sie die Breite des Elements von der Gesamtbreite der Seite (immer 100%). Das halbierte Ergebnis zeigt den nötigen seitlichen Abstand. Die Formel ist einfach:

$$(100\% - \text{Breite des Elements}) : 2 = \text{seitlicher Abstand}$$

Hierbei spielt es keine Rolle, ob Sie den Abstand mit `left` oder `right` angeben – schließlich soll er auf beiden Seiten gleich sein.



Bedenken Sie, dass sich die mit `width` angegebene Breite auf den Inhaltsbereich des Elements bezieht. Um den seitlichen Abstand zu ermitteln, müssen Sie gegebenenfalls die Summe der seitlichen Rahmen sowie der Innen- und Außenabstände zur Breite des Elements hinzuzählen, bevor Sie diese halbieren. Das kann problematisch sein, weil der Wert für `border-width` nicht in Prozent angegeben werden kann.

Lösung 3: absolute Werte

Eine dritte Möglichkeit, ein Element horizontal zu zentrieren, besteht in der Verwendung absoluter Werte. Hierbei wird die Breite eines absolut positionierten Elements (plus möglicher Rahmen sowie Innen- und Außenabstände) ebenfalls halbiert. Da sich die Breite des Ansichtsbereichs ändern kann (z.B. wenn der Betrachter das Browserfenster vergrößert oder verkleinert), kann der seitliche Abstand nicht direkt angegeben werden. Stattdessen helfen wir uns wie folgt:

```
div#navigation {  
  width:      600px;  
  position:   absolute;  
  left:       50%;  
  margin-left: -300px;  
}
```

Zuerst verschieben wir das Element um 50% der Breite (des Inhaltsbereichs bzw. des umgebenden Elements) nach rechts. Danach verwenden wir einen negativen Außenabstand (siehe Rezept 1.17), um das Element um die Hälfte seiner eigenen Breite wieder zurückzuschieben. Der Vorteil dieser Methode besteht darin, dass sich auch mögliche Innenabstände und Rahmen genau berücksichtigen lassen.

Lösung 4: Inline-Elemente und Fließtext zentrieren

Handelt es sich um ein Inline-Element (z.B. `<span>`, `<abbr>` oder `<img>`) oder auch um reinen Text, etwa innerhalb eines Absatzes oder einer Überschrift (`<h1>`, `<h2>` usw.), können Sie dessen Zentrierung mithilfe der Eigenschaft `text-align` und dem dazugehörigen Wert `center` erreichen. Hierbei muss die Definition für das umgebende Blockelement vorgenommen werden. Um beispielsweise alle Menüeinträge aus Lösung 1 mittig auszurichten, können Sie schreiben:

```
#nav_container {
    width:      836px;
    margin:     auto;
    text-align: center;
}
```

Obwohl wir die Eigenschaft `text-align` eigentlich für ein Blockelement definieren, wirkt sie sich nicht auf das Element selbst, sondern auf seinen Inhalt aus. Durch die Vererbung (siehe Rezept 1.18) wird der Text der enthaltenen Listeneinträge zentriert. Ein Beispiel sehen Sie in Abbildung 2-21.



Abbildung 2-21: Horizontal zentrierter Text

Diskussion

CSS bietet verschiedene Möglichkeiten, Block- und Inline-Elemente im Ansichtsbereich oder innerhalb des umgebenden Blocks horizontal zu zentrieren. Für Blockelemente wird hierbei meist die in Lösung 1 gezeigte Methode der automatischen Außenabstände (`margin: auto;`) verwendet. Diese funktioniert allerdings nur, solange die Elemente weder absolut oder fest positioniert noch als Float definiert wurden. Falls doch, können Sie sich mit den in Lösung 2 und 3 gezeigten Verfahren behelfen. Hierbei ist die Verwendung absoluter Werte zwar komplizierter, aber meist auch präziser, weil in Pixeln angegebene Rahmenbreiten berücksichtigt werden können.

Die in Lösung 4 gezeigte Methode funktioniert nur für Inline-Elemente und Fließtext. Hierzu gehören übrigens auch Bilder (`<img>`-Elemente), sofern sie nicht als Float definiert wurden.

Siehe auch

Die CSS 2.1-Spezifikation zur visuellen Formatierung von Elementen unter <http://www.w3.org/TR/CSS2/visudet.html>; das Wiki der Mailingliste *css-discuss* zum Thema Zentrierung von Blockelementen unter http://css-discuss.incutio.com/wiki/Centering_Block_Element; die übrigen Rezepte in diesem Buch zum Thema Zentrierung; das Rezept 3.12 zur Verwendung der Eigenschaft `text-align`.

2.25 Elemente vertikal zentrieren

Problem

Sie wollen ein Element vertikal zentrieren, wie in Abbildung 2-22 zu sehen.



Abbildung 2-22: Ein Element wird vertikal zentriert

Allgemeines

Um ein Element vertikal zu zentrieren, wird im Gegensatz zur horizontalen Zentrierung ein Container benötigt, in dem das Element zentriert wird. Dies ist in den meisten Fällen ein `div`-Element, das mindestens so hoch sein sollte wie das Element, das zentriert werden soll.

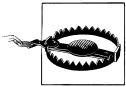
Lösung 1: table-cell

Für diese Lösung ändern Sie die Darstellungsrolle (siehe Rezept 1.17) des Container-`div` in `table-cell`. Danach können Sie die Eigenschaft `vertical-align` mit dem Wert `middle` versehen, um den Inhalt des Containers vertikal zu zentrieren. Diese Methode funktioniert auch, wenn die Höhe des Inhalts nicht bekannt ist:

```
div {
    display:      table-cell;
    vertical-align: middle;
}
```

Hier ein Beispiel für den dazu passenden HTML-Code:

```
<div>
  
</div>
```



Diese Methode hat eine kleine Nebenwirkung: div-Elemente, die per `display: table-cell` eine neue Darstellungsrolle erhalten haben, können nicht mehr per `margin: auto` horizontal zentriert werden.



Diese Methode funktioniert im Internet Explorer erst ab Version 8.

Lösung 2: Prozentwerte

Analog zur Lösung 2 des vorigen Rezept 2.24 können auch Prozentwerte für die vertikale Zentrierung verwendet werden. Allerdings muss für diese Methode die Höhe des Elements bekannt sein. Die Formel lautet in diesem Fall:

$(100\% - \text{Höhe des Elements}) : 2 = \text{vertikaler Abstand}$

Beträgt die Höhe des Elements zum Beispiel 30% des Ansichtsbereichs (bzw. des umgebenden Elements), können Sie das Element mit folgender Regel vertikal zentrieren:

```
img {
    position: absolute;
    height: 30%;
    top: 35%;
    overflow-y: auto;
}
```

Die erste Deklaration sorgt dafür, dass das Element absolut positioniert wird. (Vergessen Sie nicht, bei Bedarf den nötigen Positionierungskontext zu schaffen, wie in Rezept 2.19 beschrieben.) Im nächsten Schritt geben wir an, dass seine Höhe 30% des umgebenden Blocks betragen soll. Gemäß der oben gezeigten Formel definieren wir für `top` einen Wert von 35% (die Hälfte der verbleibenden 70%). Zum Schluss erhält die Eigenschaft `overflow-y` noch den Wert `auto`. Damit beugen wir dem Fall vor, dass die Höhe von 30% nicht ausreicht, um den gesamten Inhalt aufzunehmen.

Im Gegensatz zur ersten Lösung wird bei diesem Verfahren kein umgebendes Element benötigt. Die Regeln werden für das Element definiert, das zentriert werden soll. Der Preis sind die absolute Positionierung und die ausdrückliche Höhenangabe.

Lösung 3: absolute Werte

Wie die vorige Lösung verwendet auch dieser Ansatz die absolute Positionierung (siehe Rezept 2.17), um das gewünschte Element vertikal zu zentrieren. Auch hier entspricht der Ansatz der Methode für die horizontale Zentrierung:

```
#vertikal_zentrieren {  
  position: absolute;  
  height: 400px;  
  top: 50%;  
  margin-top: -200px;  
  overflow: auto;  
}
```

Entsprechend der horizontalen Zentrierung wird das Element zuerst absolut positioniert und dann um die Hälfte der Höhe des umgebenden Blocks nach unten verschoben. Danach verwenden wir die Eigenschaft `margin-top` mit einem negativen Wert, um das Element um die Hälfte seiner Höhe wieder nach oben zu rücken.

Diskussion

Lange Zeit schien es nicht möglich, Elemente im Ansichtsbereich oder im umgebenden Block vertikal zu zentrieren, ohne dass dessen Höhe bekannt ist. Mithilfe der `vertical-align`-Methode (Lösung 1) ist zumindest für aktuelle Browser (außer IE7) eine Lösung gefunden, die ohne zusätzliche HTML-Elemente auskommt.

Bei Verwendung der Lösungen 2 und 3 sollten Sie daran denken, dass eine ausdrückliche Höhenangabe zu Problemen führen kann, wenn der Inhalt mehr Platz benötigt als vorgesehen. Das kann bei dynamisch erzeugten Webseiten (z.B. Blogs), aber auch bei einer Änderung der Schriftgröße durch den Benutzer der Fall sein. Daher wird empfohlen, den Standardwert der Eigenschaft `overflow-y` (`visible`) entsprechend abzuändern. Durch den Wert `auto` erzeugt der Browser bei Bedarf Scrollbalken, um den Inhalt zugänglich zu machen, falls der Platz nicht ausreicht.

Siehe auch

Die Linksammlung im Wiki der Mailingliste *css-discuss* zum Thema vertikale Zentrierung unter http://css-discuss.incutio.com/wiki/Main_Page; das Rezept 2.24 zur horizontalen Zentrierung von Elementen.

Typografie für das Web

Einführung

Kaum eine Website kommt völlig ohne Schrift aus. Selbst Seiten, die Bilder oder Filme zum Thema haben (zum Beispiel Flickr oder YouTube), verwenden Texte, um die eigentlichen Inhalte zu erläutern. Ursprünglich war das Web sogar ein komplett bilderloses Medium. Und auch heute noch spielen Texte auf den meisten Websites die Hauptrolle.

Daher ist es auch nicht verwunderlich, dass CSS vor Möglichkeiten strotzt, Text zu gestalten. Fast alle Parameter der Schriftdarstellung lassen sich anpassen – von der Schriftfarbe über den Zeilenabstand bis hin zur Darstellung von Kapitälchen, und das alles, ohne hierfür den HTML-Quelltext verändern zu müssen.

Ein großes Problem bei der Darstellung von Texten mit CSS bestand in der relativ geringen Anzahl von Schriften, die gefahrlos im Web verwendet werden konnten. Mittlerweile sind aber alle aktuellen Browser in der Lage, externe Schriftdateien einzubetten, sodass statt der zwanzig bis dreißig Standardschriften plötzlich Hunderte, wenn nicht Tausende frei verfügbarer Schriften benutzt werden können.

In diesem Kapitel werden Sie sehen, wie sich die verschiedenen typografischen Aspekte einer Website an die eigenen Bedürfnisse anpassen lassen, ohne dass Sie hierfür den Markup-Code der Seite ändern müssen. Willkommen in der schönen neuen Welt der Webtypografie.

3.1 Schriftarten festlegen

Problem

Sie wollen in Ihren Webseiten Schriften verwenden, die nach Möglichkeit von allen Browsern verstanden werden.

Lösung

Verwenden Sie die Eigenschaft `font-family`, um die gewünschte Schrift und eine Reihe von Alternativen anzugeben. Beschränken Sie sich bei der Auswahl Ihrer Schriften auf solche, die mit größter Wahrscheinlichkeit auf allen Computern installiert sind. Hier ein paar Beispiele:

Serifenlose Schriften

```
font-family: Verdana, Geneva, sans-serif;  
font-family: Arial, Helvetica, sans-serif;  
font-family: Tahoma, Geneva, sans-serif;  
font-family: "Trebuchet MS", Arial, Helvetica, sans-serif;  
font-family: "Lucida Sans Unicode", "Lucida Grande", sans-serif;
```

Serifenschriften:

```
font-family: Georgia, "Times New Roman", Times, serif;  
font-family: "Palatino Linotype", "Book Antigua", Palatino, serif;  
font-family: "MS Serif", New York, serif;
```

Festbreitenschriften:

```
font-family: "Courier New", Courier, monospace;  
font-family: "Lucida Console", Monaco, monospace;
```

Kursive Schriften:

```
font-family: "Comic Sans MS", cursive;
```

Diskussion

Wenn Sie eine durch Kommata getrennte Liste für die Eigenschaft `font-family` angeben, können Sie festlegen, welche Schriftarten der Browser für die Darstellung einer Webseite verwenden soll. Kann der Browser die erste Schrift in der Liste nicht finden, versucht er es mit der nächsten, bis eine passende Schrift gefunden ist.

Weniger bekannt ist die Tatsache, dass die Schriftersetzung auch funktioniert, wenn nur einzelne Zeichen einer Schrift nicht gefunden werden können. Ist ein bestimmtes Zeichen in der ersten angegebenen Schrift nicht enthalten, versuchen die meisten Browser (außer Internet Explorer vor der Version 8), das Zeichen in der zweiten genannten Schrift zu finden, und so weiter.

Wie in Tabelle 3-1 zu sehen ist, gibt es ungefähr 13 Schriften, die sowohl auf Windows-Systemen wie auch auf dem Mac zu finden sind.

Tabelle 3-1: Websichere Schriften

Schrift	Allgemeine Schriftfamilie	Beispiel
Arial	sans-serif	ABCDEFGHIJKLMNOPQRSTUVWXYZ abcdefghijklmnopqrstuvwxyz 0123456789
Arial Black	sans-serif	ABCDEFGHIJKLMNOPQRSTUVWXYZ abcdefghijklmnopqrstuvwxyz 0123456789
Comic SansMS*	cursive	ABCDEFGHIJKLMNOPQRSTUVWXYZ abcdefghijklmnopqrstuvwxyz 0123456789
Courier New	monospace	ABCDEFGHIJKLMNOPQRSTUVWXYZ abcdefghijklmnopqrstuvwxyz 0123456789
Georgia	serif	ABCDEFGHIJKLMNOPQRSTUVWXYZ abcdefghijklmnopqrstuvwxyz 0123456789
Helvetica	sans-serif	ABCDEFGHIJKLMNOPQRSTUVWXYZ abcdefghijklmnopqrstuvwxyz 0123456789
Impact	sans-serif	ABCDEFGHIJKLMNOPQRSTUVWXYZ abcdefghijklmnopqrstuvwxyz 0123456789
Tahoma	sans-serif	ABCDEFGHIJKLMNOPQRSTUVWXYZ abcdefghijklmnopqrstuvwxyz 0123456789
Times	serif	ABCDEFGHIJKLMNOPQRSTUVWXYZ abcdefghijklmnopqrstuvwxyz 0123456789
Times New Roman	serif	ABCDEFGHIJKLMNOPQRSTUVWXYZ abcdefghijklmnopqrstuvwxyz 0123456789
Trebuchet MS	sans-serif	ABCDEFGHIJKLMNOPQRSTUVWXYZ abcdefghijklmnopqrstuvwxyz 0123456789
Verdana	sans-serif	ABCDEFGHIJKLMNOPQRSTUVWXYZ abcdefghijklmnopqrstuvwxyz 0123456789

* Über die Verwendung dieser Schrift gehen die Meinungen sehr weit auseinander. Entscheiden Sie selbst.

Tabelle 3-1: Websichere Schriften (Fortsetzung)

Schrift	Allgemeine Schriftfamilie	Beispiel
Wingdings	symbol	



Courier, Helvetica und Times sind außerdem auf den meisten Unix/Linux-Systemen mit X11-Unterstützung zu finden. Die anderen »websicheren« Schriften in der obigen Tabelle sind dagegen nicht sehr häufig vertreten.

Besteht der Schriftname aus mehreren Wörtern (z.B. Times New Roman) bzw. enthält er Leerzeichen, sollten Sie ihn mit einfachen oder doppelten Anführungszeichen umgeben werden, wie hier:¹

```
font-family: Georgia, "Times New Roman", Times, serif;
```

Am Ende der Liste mit expliziten Schriftnamen sollten Sie mindestens eine *allgemeine Schriftfamilie* angeben. Sie gibt das Hauptmerkmal der verwendeten Schrift an und ist quasi der kleinste gemeinsame Nenner zwischen Ihren Wünschen und dem, was der Browser darstellen kann. In CSS können Sie aus fünf Schriftfamilien wählen (siehe Tabelle 3-2).

Tabelle 3-2: Allgemeine Schriftfamilien in CSS

Allgemeine Schriftfamilie	Erklärung	Beispiele für Schriftnamen dieser Familie
serif	Schrift mit Verzierungen am Ende der Buchstabenstriche (Serifen)	Georgia, Times, Times New Roman, Garamond und Century Schoolbook
sans-serif	Schrift ohne Verzierungen am Ende der Buchstabenstriche	Verdana, Arial, Helvetica, Trebuchet und Tahoma
monospace	Festbreitenschrift	Courier, MS Courier New und Prestige
cursive	Geneigte Schrift	Lucida Handwriting und Zapf-Chancery
fantasy	Dekorative Schrift	Whimsey, Critter und Cottonwood

Alle Webbrowser enthalten eine Liste mit Schriften, die den Familien in Tabelle 3-2 zugeordnet werden können. Kann der Browser keine der mit Namen angegebenen Schriften darstellen (z.B. weil sie nicht auf dem Rechner des Benutzers verfügbar ist), verwendet er einen anderen Font aus einer dieser Schriftfamilien.

¹ Die CSS 2.1-Spezifikation sagt dazu: »Um Missverständnisse zu vermeiden, wird empfohlen, Schriftnamen, die Leerzeichen, Zahlen oder (andere) Interpunktionszeichen (als Bindestriche) enthalten, mit Anführungszeichen zu umgeben.«

Wir empfehlen, als letzten Wert für die Eigenschaft `font-family` entweder `serif`, `sans-serif` oder `monospace` zu verwenden. Das Ergebnis der Werte `cursive` und `fantasy` ist nur schwer vorhersehbar.

Der Wert der Eigenschaft `font-family` wird an Kindelemente vererbt. Mehr zum Thema Vererbung finden Sie in Rezept 1.18.

Die Liste der websicheren Schriften erweitern

Die Softwarepakete Microsoft Office und Apple iWork enthalten zusätzliche Schriften, die zusammen mit den Programmen installiert werden. Wenn man davon ausgeht, dass eine große Zahl von Computerbenutzern zumindest eines dieser Pakete installiert hat, kann man die Liste möglicher Schriften erweitern.

Der Webdesigner Jason Cranford Teague hat hierzu eine erweiterte Liste websicherer Schriften zusammengestellt. Außerdem gibt er an, mit welcher Wahrscheinlichkeit die Schriften verfügbar sind. Sie finden die Liste unter <http://tr.im/xGGi>.

Siehe auch

Genauer zu den Werten für `font-family` finden Sie in der CSS 2.1-Spezifikation unter <http://www.w3.org/TR/CSS21/fonts.html#propdef-font-family>; eine Vorschau websicherer Schriften bei The Web Safe Fonts Preview unter http://www.fonttester.com/web_safe_fonts.html; eine Liste mit Linux-Entsprechungen der gängigsten Webschriften unter <http://mondaybynoon.com/20070402/linux-font-equivalents-to-popular-web-typefaces/>.

3.2 Schriftgrößen und -maße festlegen

Problem

Sie wollen die in einer Webseite zu verwendende Schriftgröße festlegen.

Lösung

Definieren Sie die nötigen Werte mithilfe der Eigenschaft `font-size`. Nutzen Sie bei der Gestaltung von Webseiten als Wert möglichst eine relative Maßeinheit wie beispielsweise `em` (siehe die Rezepte 2.1 und 2.2):

```
p {  
    font-size: 0.9em;  
}
```

Mit dieser Regel wird festgelegt, dass der Text in Absätzen um ein Zehntel kleiner sein soll als die Schriftgröße des Elternelements.

Diskussion

Die Definition der Schriftgröße hat bei der Gestaltung vieler Websites eine besondere Bedeutung. Oftmals wollen Sie anstelle der im Browser eingestellten Standardschriftgröße (meistens 16 Pixel) eigene Werte verwenden. Dies gilt speziell für Überschriften, Navigationselemente und Ähnliches. Im Prinzip können Sie für die Eigenschaft `font-size` alle in CSS erlaubten Längeneinheiten verwenden (siehe Rezept 2.1). Negative Werte, wie `-25cm`, sind nicht erlaubt.

Für die Bildschirmdarstellung werden Schriftgrößen fast immer in relativen Maßeinheiten (z.B. `em`) oder auch in Prozentwerten angegeben. Außerdem sind bestimmte Schlüsselwörter und numerische Werte möglich.

Absolute Werte

Um die Darstellung von Text möglichst genau kontrollieren zu können, setzen manche Designer auf die Verwendung absoluter Maßeinheiten, z.B. Punkt (`pt`) oder Zentimeter (`cm`). Allerdings werden Punkangaben eigentlich nur beim Ausdrucken verwendet (siehe Kapitel 9, *Drucken*).



Historisch gesehen, kann die Maßeinheit `pt` verschiedene Bedeutungen haben. In CSS entspricht `1pt` $\frac{1}{72}$ eines Zolls (`1in`).

Relative Werte

Auf der anderen Seite steht die Auffassung, dass man die Schriftgröße gar nicht genau kontrollieren sollte bzw. kann. Daher verlassen sich einige Designer lieber auf die im Browser eingestellte Standardschriftgröße als Referenzwert. Diese beträgt zwar standardmäßig 16 Pixel, lässt sich aber vom Benutzer in den Browsereinstellungen problemlos ändern.

Die am häufigsten verwendeten relativen Maßeinheiten sind `em`-Einheiten und Prozentwerte. Dabei bezieht sich die tatsächliche Schriftgröße darauf, welcher Wert vom Browser für das Elternelement berechnet wurde. Mehr dazu im folgenden Abschnitt.

Vererbung

Bedenken Sie, dass die Werte für `font-size` vererbt werden (siehe Rezept 1.18). Das bedeutet, wenn Sie die Schriftgröße anhand von Prozentwerten oder `em`-Einheiten festlegen, berechnet der Browser das Ergebnis anhand der Schriftgröße des Elternelements. Hier ein Beispiel:

CSS:

```
body { font-size: 20px; }
p    { font-size: 125%; } /* 25 Pixel */
em   { font-size: 80%; } /* 20 Pixel oder 16 Pixel? */
```

HTML:

```
<body>
<p>Die Schriftgröße für diesen Absatz beträgt 25 Pixel (125% des für body definierten
Werts). <em>In diesem em-Element sind es wieder 20 Pixel (80% von 25 Pixel).</em> Danach
geht es weiter wie vorher.</p>

<div>Für dieses div-
Element wurde keine eigene Schriftgröße festgelegt. Sie entspricht also dem für body
definierten Wert (20 Pixel). <em>Dadurch ist die Schriftgröße in diesem em-Element
kleiner als im Absatz (80% von 20 Pixel = 16 Pixel).</em></div>
</body>
```

Das Ergebnis dieser Einstellungen sehen Sie in Abbildung 3-1. (Zur Verbesserung der Lesbarkeit wurden die em-Elemente mit einem hellgrauen Hintergrund versehen.)

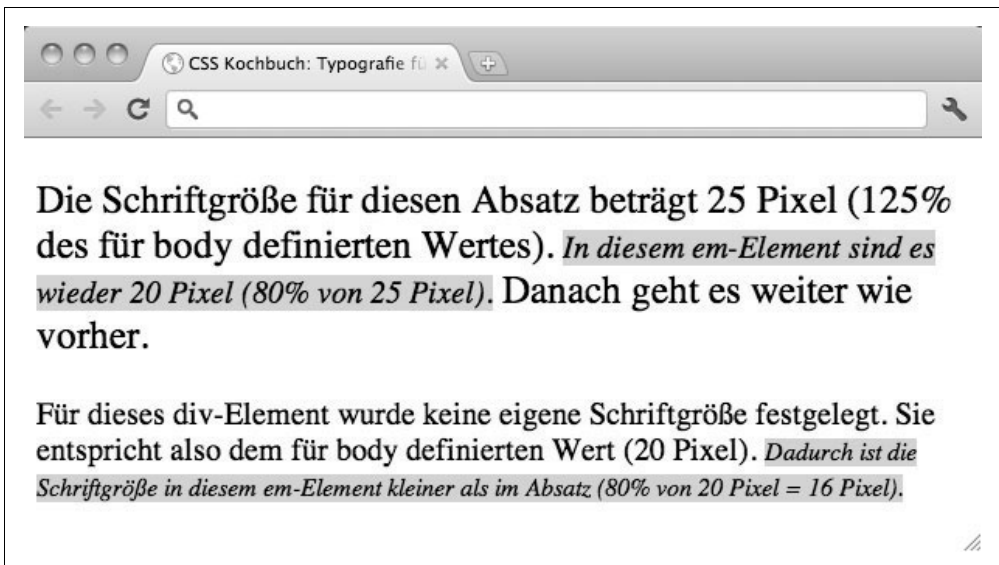


Abbildung 3-1: Die Werte für font-size werden vererbt

Schlüsselwörter

Neben absoluten und relativen Maßeinheiten definiert die CSS 2.1-Spezifikation sieben Schlüsselwörter, mit denen die Schriftgröße gesteuert werden kann. Die tatsächliche Größe leitet der Browser von der eingestellten Standardschriftgröße (16 Pixel) ab, sodass es sich im weitesten Sinne auch hier um relative Angaben handelt. Die folgenden Schlüsselwörter sind möglich: xx-small (am kleinsten), x-small (extraklein), small (klein), medium (Standardwert, 16 Pixel entsprechend, mittel), large (groß), x-large (extragroß) und xx-large (am größten).

Zwei weitere Schlüsselwörter sorgen für eine relative Änderung der Schriftgröße: larger (größer) und smaller (kleiner). Die Angaben beziehen sich dabei auf den Wert, den der Browser für das Elternelement berechnet hat.

Die Verwendung von Schlüsselwörtern hat zwei Vorteile: Erstens lässt sich der Text in den meisten Browsern auf diese Weise einfach vergrößern oder verkleinern. Zweitens sinkt die Schriftgröße bei CSS-konformen Browsern nie unter 9 Pixel und stellt auf diese Weise sicher, dass der Text prinzipiell lesbar bleibt. Ein kleiner Wermutstropfen bleibt: Die Schlüsselwörter werden von den Browsern unterschiedlich umgesetzt und sollten daher mit Bedacht eingesetzt werden.

rem-Einheiten

Die neuesten Versionen der wichtigsten Browser (FF3.6+, IE9+, S5+, C9+, leider kein Opera) unterstützen außerdem die Verwendung einer mit CSS3 neu hinzugekommenen Maßeinheit: `rem` (root em). Im Gegensatz zu den »normalen« `em`-Einheiten beziehen sich `rem`-Werte immer auf die Schriftgröße des Wurzelements, egal wie tief das betreffende Element verschachtelt ist.

Da sich die Browserdarstellung aufgrund unterschiedlicher Betriebssysteme und Videoeinstellungen stark unterscheiden kann, ist der Einsatz von *festen* (oder *absoluten*) Werten wenig sinnvoll. Am besten vermeiden Sie absolute Maßeinheiten für Webdokumente ganz, es sei denn, Sie gestalten Dokumente, bei denen die Ausgabegrößen vorher festgelegt sind. Für die Druckausgabe sind absolute Einheiten jedoch die bessere Wahl. Näheres zum Anlegen von Stylesheets für die Druckausgabe finden Sie in Kapitel 9, *Drucken*.

Siehe auch

Das Rezept 2.1 zur Verwendung verschiedener Längeneinheiten; den Artikel *How to size text in CSS* unter <http://www.alistapart.com/articles/howtosizetextincss/>; den Abschnitt »Schriftgrößen« in Kapitel 5 von *CSS – Das umfassende Handbuch* von Eric A. Meyer (O'Reilly 2007); die CSS 2.1-Spezifikation zur Eigenschaft `font-size` unter <http://www.w3.org/TR/CSS21/fonts.html#font-size-props>; das CSS3-Modul *Values and Units* unter <http://www.w3.org/TR/css3-values/#numbers0>; die *Caniuse*-Website zur Browserunterstützung für `rem`-Einheiten unter <http://www.caniuse.com/#search=rem>.

3.3 Die Schriftfarbe ändern

Problem

Sie wollen bestimmte Elemente in einer eigenen Schriftfarbe gestalten

Lösung

Verwenden Sie die Eigenschaft `color` mit einem in CSS gültigen Farbwert, wie hier gezeigt:

```
h2 {  
    color: #2F4F4F;  
}
```


In diesem Beispiel geben wir den Wert der Eigenschaft `color` als hexadezimalen Wert an. Durch diese Regel werden alle Überschriften zweiter Ordnung in einem dunklen Schiefergrau dargestellt.

Diskussion

Standardmäßig stellen so gut wie alle Browser Text in schwarzer Schrift dar. Je nachdem, welches Farbschema Sie für Ihre Website verwenden, wollen Sie möglicherweise einen eigenen Wert für die Schriftfarbe definieren. Dies geschieht in CSS mithilfe der Eigenschaft `color`. Als Wert können Sie eine beliebige, in CSS gültige Farbangabe inklusive transparent verwenden, wie im Grundrezept 2.3 beschrieben wird.



Tatsächlich bestimmt die Eigenschaft `color` die Vordergrundfarbe eines Elements. Bei HTML ist das in der Regel der dargestellte Text. Zudem gilt der Wert aber auch für die Farbe von Rahmen um bestimmte Elemente, sofern kein spezieller Wert für die Eigenschaft `border-color` (oder die entsprechenden Einzeleigenschaften) festgelegt wurde.

Siehe auch

Das Grundrezept 2.3 zur Definition von Farben; die CSS 2.1-Spezifikation zur Eigenschaft `color` unter <http://www.w3.org/TR/CSS2/colors.html#propdef-color>.

3.4 Das Schriftgewicht ändern

Problem

Bestimmte Textteile sollen mehr oder weniger fett dargestellt werden, ohne dass hierfür spezielle HTML-Tags verwendet werden müssen.

Lösung

Passen Sie das Schriftgewicht mithilfe der Eigenschaft `font-weight` an die eigenen Wünsche an, z.B.:

```
strong {  
    font-weight: bold;  
}
```

Der dazugehörige HTML-Code könnte so aussehen:

```
<p>Hier ist ein Absatz mit belanglosem Text. <strong>Aber dieser Abschnitt ist wirklich  
wichtig.</strong> Diesen Teil können Sie vergessen.</p>
```

Diskussion

Eines der größten Probleme früherer Versionen von HTML bestand in der Vermischung von Inhalt und Darstellung. Viele Tags existierten allein zu dem Zweck, bestimmte Ele-

mente auf bestimmte Weise darzustellen. Über das allzu beliebte `<font>`-Tag breiten wir hier lieber das Mäntelchen des Schweigens. Seine Zeit ist hoffentlich endgültig vorbei!

Trotzdem haben auch in HTML5 einige reine Layout-Tags überlebt, namentlich `<b>` und `<i>`. Mit diesen beiden Tags können Sie den Browser anweisen, Text fett bzw. kursiv darzustellen (was sich natürlich mit CSS auch leicht wieder ändern lässt).

Da diese Methode jedoch nicht besonders flexibel ist und zudem die gewünschte Trennung unterwandert, ist es besser, den Text neutral zu markieren (wie hier mit einem `span`-Tag) und dann im Stylesheet festzulegen, wie er angezeigt werden soll.²

Als Wert für die Eigenschaft `font-weight` können Sie entweder ein Schlüsselwort oder einen numerischen Wert angeben. Die Schlüsselwörter sind: `normal`, `bold`, `bolder` und `lighter`. (Wenn Sie wollen, dass `font-weight` seinen Wert erbt, verwenden Sie das Schlüsselwort `inherit`.) Die beiden ersten Werte legen fest, dass der Text mit normalem Schriftgewicht oder fett dargestellt werden soll. Die Schlüsselwörter `bolder` (fetter) und `lighter` (weniger fett) verändern das Schriftgewicht basierend auf dem aktuellen Wert.

Neben den Schlüsselwörtern kann das Schriftgewicht auch in Hunderterschritten auf einer Skala zwischen 100 (sehr leicht) und 900 (sehr fett) angegeben werden. Hierbei entspricht die Zahl 400 meistens dem Schriftgewicht `normal`, während der Wert 700 für `bold` steht. Die tatsächliche Zuordnung hängt allerdings auch von der verwendeten Schriftfamilie (`font-family`) ab. Daher ist es meistens einfacher, die Schlüsselwörter zu benutzen, wie in der Lösung gezeigt.

Siehe auch

Die CSS 2.1-Spezifikation zur Eigenschaft `font-weight` unter <http://www.w3.org/TR/CSS2/fonts.html#font-boldness>; eine ausführliche Erläuterung der möglichen Werte für `font-weight` in Kapitel 5 von *CSS – Das umfassende Handbuch* von Eric A. Meyer (O'Reilly 2007).

3.5 Den Schriftstil anpassen

Problem

Sie wünschen sich für bestimmte Textteile einen eigenen Schriftstil (z.B. *kursiv*).

Lösung

Verwenden Sie die Eigenschaft `font-style` und geben Sie ihr einen passenden Wert:

```
.wichtig {  
    font-style: italic;  
}
```

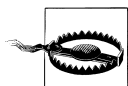
² Okay, Sie *könnten* auch eine Regel für das `<b>`-Element definieren ...

Die obige Regel sorgt dafür, dass Elemente, die mit der Klasse wichtig gekennzeichnet wurden, in kursiver Schrift dargestellt werden.

Diskussion

Ähnlich wie das im vorigen Rezept angesprochene `b`-Element hat auch das `i`-Tag keine wirklich semantische (den Inhalt beschreibende) Bedeutung. Um Inhalt und Gestaltung möglichst getrennt zu halten, wird empfohlen, stattdessen die entsprechende CSS-Eigenschaft `font-style` zu verwenden. Mögliche Werte sind `italic` (»echte« Kursivschrift), `oblique` (geneigte Schrift) und `normal`. (Soll der Wert von einem Vorfahrenelement geerbt werden, verwenden Sie den Wert `inherit`.)

Einige Schriften enthalten speziell gestaltete Buchstaben und Zeichen für die kursive Darstellung. Um diese zu verwenden, geben Sie den Wert `italic` an. Bei der Angabe von `oblique` benutzt der Browser dagegen eine geneigte Variante der Ausgangsschrift. Hierdurch wird sichergestellt, dass die Darstellung jedenfalls ungefähr den Erwartungen des Autors entspricht. (Im Printbereich würde man darüber vermutlich eher die Nase rümpfen.)



Bitte verwechseln Sie kursive Schriften nicht mit der in Rezept 3.1 beschriebenen Schriftfamilie *cursive*.

Enthält eine Schrift keine kursive Variante, wird an ihrer Stelle die geneigte Version verwendet und umgekehrt. Das hat zur Folge, dass `italic` und `oblique` in vielen Fällen gleich ist. Einen Vergleich von normaler und kursiver Schrift sehen Sie in Abbildung 3-2.



Abbildung 3-2: Normale und kursive Schrift

Siehe auch

Die CSS 2.1-Spezifikation zur Eigenschaft `font-style` unter <http://www.w3.org/TR/CSS2/fonts.html#font-styling>; eine ausführliche Erläuterung von `font-style` in Kapitel 5 von *CSS – Das umfassende Handbuch* von Eric A. Meyer (O'Reilly 2007).

3.6 Text in Kapitälchen darstellen

Problem

Einige Textteile sollen in Kapitälchen dargestellt werden.

Lösung

Verwenden Sie die Eigenschaft `font-variant` und geben Sie ihr den Wert `small-caps`:

```
h3 {  
    font-variant: small-caps;  
}
```

Mithilfe dieser Regel werden Überschriften dritter Ordnung (`<h3>`) in sogenannten Kapitälchen dargestellt (siehe Abbildung 3-3).



Abbildung 3-3: Eine Überschrift wird in Kapitälchen dargestellt

Diskussion

Kapitälchen bestehen komplett aus Großbuchstaben. Trotzdem wird zwischen Groß- und Kleinschreibung unterschieden. Hierbei wird die Höhe der »Kleinbuchstaben« der

Größe tatsächlicher Kleinbuchstaben angepasst. Manche Schriften enthalten neben einer kursiven Variante (siehe voriges Rezept) einen speziellen Kapitälchen-Schriftschnitt.

Die Eigenschaft `font-variant` kennt nur zwei Werte: `small-caps`, `normal` und `inherit`. Der erste Wert sorgt für die Anzeige in Kapitälchen, der zweite (Standardwert) setzt sie gegebenenfalls wieder auf die Normaldarstellung zurück. (Dieser Wert wird nicht vererbt. Soll ein Element den Wert für `font-variant` erben, müssen Sie der Eigenschaft den Wert `inherit` zuweisen.)

Siehe auch

Den Wikipedia-Eintrag zu Kapitälchen unter <http://de.wikipedia.org/wiki/Kapitälchen>; die CSS 2.1-Spezifikation unter <http://www.w3.org/TR/CSS2/fonts.html#small-caps>; das Rezept 1.18 zum Thema Vererbung.

3.7 Text in Groß- und Kleinschreibung darstellen

Problem

Sie möchten bestimmte Textteile komplett in Groß- oder Kleinbuchstaben darstellen, ohne hierfür den Text selbst verändern zu müssen. Optional soll jedes Wort mit einem großgeschriebenen Anfangsbuchstaben versehen werden.

Lösung

Verwenden Sie die Eigenschaft `text-transform` mit einem passenden Wert. Ein Beispiel:

```
#einleitung {  
  text-transform: uppercase;  
}
```

Der HTML-Code enthält einen Absatz mit der ID `einleitung`. Mithilfe der Eigenschaft `text-transform` kann der Text ohne weitere Änderungen am Quellcode in Großbuchstaben dargestellt werden:

```
<p id="einleitung">Dies ist die Einleitung ...</p>  
<p>Noch mehr Text ...</p>
```

Mit der neuen CSS3-Pseudoklasse `:first-of-type` können Sie sich sogar das `id`-Attribut sparen. Stattdessen schreiben Sie im Stylesheet einfach:

```
p:first-of-type {  
  text-transform: uppercase;  
}
```

Diskussion

Im Gegensatz zur Eigenschaft `font-variant`, mit der Sie Textteile in Kapitälchen darstellen können, sorgt `text-transform` für eine Umwandlung in Groß- bzw. Kleinbuchstaben. Hierfür gibt es die Werte `uppercase` (gesamten Text in Großbuchstaben anzeigen), `lowercase` (Kleinbuchstaben) und `capitalize`. Während die ersten beiden Werte ganze Wörter verändern, sorgt `capitalize` dafür, dass jedes Wort in der Darstellung mit einem Großbuchstaben beginnt.

Siehe auch

Die CSS 2.1-Spezifikation zur Eigenschaft `text-transform` unter <http://www.w3.org/TR/CSS2/text.html#caps-prop>.

3.8 Zeilenhöhen ändern

Problem

Sie wollen den Abstand zwischen den Zeilen ändern, wie in Abbildung 3-4 gezeigt.

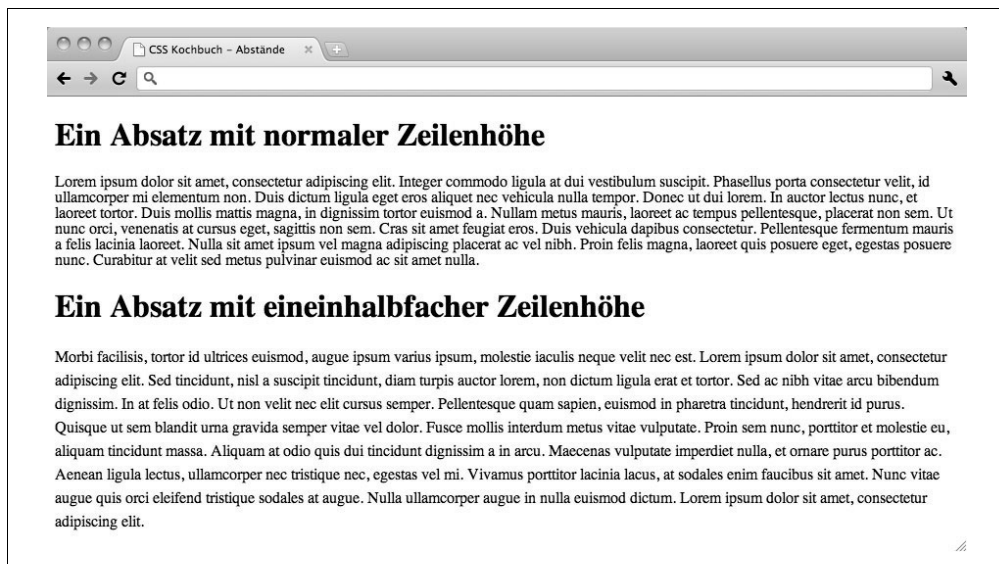


Abbildung 3-4: Zwei Absätze mit unterschiedlichen Zeilenhöhen

Lösung

Passen Sie den Wert für die Eigenschaft `line-height` an:

```
p {  
  line-height: 1.5;  
}
```

Die hier gezeigte Regel legt fest, dass die Zeilenhöhe das Eineinhalbfache der Standardschriftgröße betragen soll.

Diskussion

Durch eine Erhöhung des Werts für die Eigenschaft `line-height` vergrößert sich die Zeilenhöhe. Wird der Wert verringert, wird auch der Abstand zwischen den Zeilen kleiner, bis sich die Zeilen möglicherweise überschneiden.

Indirekt kontrollieren Sie mit dem Wert von `line-height` den zusätzlichen Raum ober- und unterhalb einer Textzeile (*Durchschuss*). Der Durchschuss entspricht also der Differenz zwischen dem errechneten Wert für `line-height` und dem errechneten Wert für `font-size`.

Normalerweise wird der Wert für `line-height` als Zahl angegeben, die einen bestimmten Skalierungsfaktor definiert. Daneben sind aber auch alle anderen in CSS gültigen Maßeinheiten für Längenangaben verwendbar. Negative Werte können nicht angegeben werden, Prozentwerte beziehen sich auf die Schriftgröße des Elements.

Siehe auch

Die CSS 2.1-Spezifikation zur Eigenschaft `line-height` unter <http://www.w3.org/TR/CSS21/visudet.html#propdef-line-height>; das Rezept 3.10 für weitere Informationen zur Kurzschrifteigenschaft `font`.

3.9 Den Abstand zwischen Buchstaben und Wörtern anpassen

Problem

Sie wollen den horizontalen Abstand zwischen Buchstaben und Wörtern (die *Laufweite*) im HTML-Text anpassen.

Lösung

Den Abstand zwischen Buchstaben können Sie mit der Eigenschaft `letter-spacing` einstellen; der Wortabstand wird durch die Eigenschaft `word-spacing` definiert.

```
h2 {  
    letter-spacing: -0.05em;  
    word-spacing: 1em;  
    font-family: Times, "Times New Roman", serif;  
}
```

Während in der ersten Deklaration der Buchstabenabstand um 0,05 em-Einheiten verringert wird, erhöhen wir in der zweiten Anweisung den Abstand zwischen den Wörtern um eine em-Einheit. Das Ergebnis sehen Sie in Abbildung 3-5.

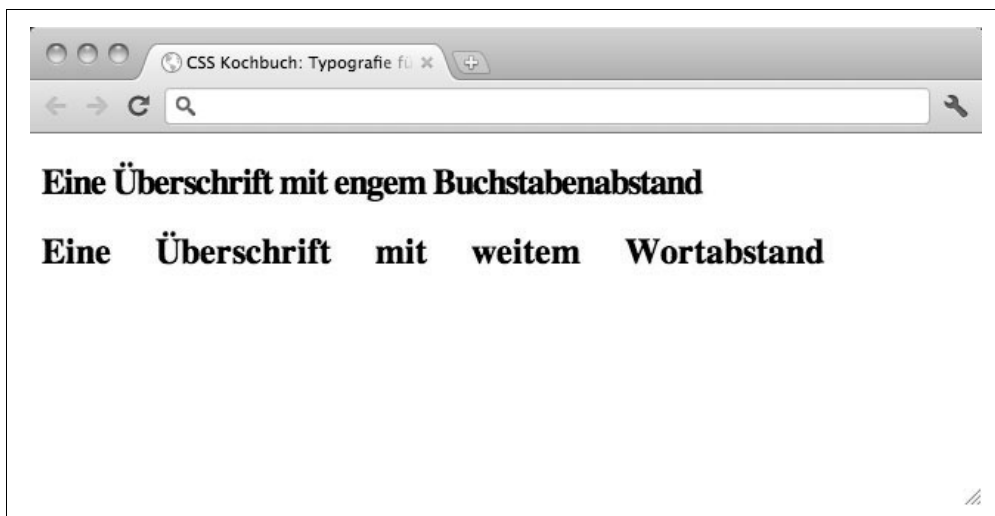


Abbildung 3-5: Veränderte Buchstaben- und Wortabstände

Diskussion

Die Anpassung der Laufweite für eine bessere Ästhetik ist eine alte Tradition in der Typografie. Zwei Fachbegriffe beschreiben die im Text vorgenommenen Abstandsänderungen: *Unterschneidung* (Verringern der Laufweite, auch *Kerning* genannt) und *Spationierung* (Vergrößern der Laufweite, auch *Sperren* genannt). In der Praxis kommt vor allem die Unterschneidung zum Einsatz.

Die Unterschneidung beschreibt die Reduzierung des Leerraums zwischen zwei Buchstaben, um eine bessere Lesbarkeit zu erreichen. Ein Beispiel für eine Unterschneidung ist die Verringerung des Abstands zwischen einem großen *T* und einem folgenden kleinen *i*. Wird die Laufweite über mehrere Textblöcke hinweg verändert, spricht man auch von *Tracking*.

Zwar können Sie für letter-spacing und word-spacing sämtliche in CSS gültigen Längeneinheiten verwenden. Allerdings gilt es allgemein als beste Vorgehensweise, die Werte in relativen Längeneinheiten anstelle von absoluten anzugeben.

Siehe auch

Die CSS 2.1-Spezifikation zum Thema letter-spacing unter <http://www.w3.org/TR/CSS2/text.html#propdef-letter-spacing> und zum Thema word-spacing unter <http://www.w3.org/TR/CSS2/text.html#propdef-word-spacing>; nähere Informationen zum Thema Laufweite und Unterschneidung finden Sie auch in der Wikipedia unter [http://de.wikipedia.org/wiki/Unterschneidung_\(Typografie\)](http://de.wikipedia.org/wiki/Unterschneidung_(Typografie)) bzw. <http://de.wikipedia.org/wiki/Schriftart#Laufweite>.

3.10 Die Kurzschrifteigenschaft font verwenden

Problem

Sie wollen die wichtigsten Einstellungen für die Schrift eines Elements nicht über die Einzeleigenschaften angeben, sondern auf einer Zeile zusammenfassen.

Lösung

Verwenden Sie die Kurzschrifteigenschaft `font`. Die folgende Regel definiert Werte für alle schriftbezogenen Eigenschaften und `line-height`:

```
h1 {  
  font-style: italic;  
  font-variant: small-caps;  
  font-weight: bold;  
  font-size: 3em;  
  line-height: 1.2;  
  font-family: Times, "Times New Roman", serif;  
}
```

Um Platz zu sparen, können Sie alle diese Definitionen auch auf einer Zeile zusammenfassen. Das sieht dann so aus:

```
h1 {  
  font: italic small-caps bold 3em/1.2 Times, "Times New Roman", serif;  
}
```

Hier sehen Sie alle Einzeleigenschaften auf einer Zeile versammelt. Die Reihenfolge ist wichtig, wie weiter unten im Abschnitt »Die Beliebigkeitsregel« in der Diskussion erklärt wird. Auffällig ist hier außerdem, dass die Werte für `font-size` und `line-height` nicht wie sonst üblich durch Leerzeichen oder Kommata, sondern durch einen Schrägstrich getrennt werden.

Diskussion

Durch die Verwendung von `font` können Sie sich viel Schreibarbeit sparen – sofern Sie wissen, was Sie tun! Die Kürze hat nämlich ihren Preis, es gibt einiges zu beachten:

Die Beliebigkeitsregel

Die Werte der drei ersten Einzeleigenschaften (`font-style`, `font-style` und `font-weight`) können in beliebiger Reihenfolge angegeben werden. Die Werte für `font-size/line-height` und `font-family` müssen dagegen in genau dieser Reihenfolge und als letzte definiert werden.

Mindestanforderungen

Damit die Angaben für `font` gültig sind, müssen zumindest Werte für `font-size` und `font-family` vorhanden sein. Die Angabe eines Werts für `line-height` ist optional. Wird dieser definiert, folgt er, durch einen Schrägstrich getrennt, direkt auf den Wert für `font-size`. Nicht angegebene Werte werden automatisch durch ihren Standardwert ersetzt. Ein Beispiel:

```
h1 {  
  font: 110% Verdana, Arial, sans-serif;  
}  
  
h2 {  
  font: 18px/20px "Courier New", Courier, monospace;  
}  
  
h3 {  
  font: lighter 14px;  
}
```

Die erste Regel erfüllt die Mindestanforderungen. Die weggelassenen Werte werden auf ihren Standardwert zurückgesetzt. Die zweite Regel funktioniert ebenfalls. Der Wert für `line-height` wird durch einen Schrägstrich von der Angabe für `font-size` getrennt. Die dritte Regel ist dagegen nicht korrekt. Sie wird verworfen, weil kein Wert für `font-family` angegeben wurde.

Überschreiben zuvor definierter Werte

Wir haben es im vorigen Abschnitt schon kurz erwähnt: Lassen Sie die Angaben für `font-style`, `font-variant` oder `font-weight` weg, werden die entsprechenden Werte automatisch auf ihren Standardwert (`normal`) zurückgesetzt. Wenn Sie nicht aufpassen, kann das zu überraschenden Ergebnissen führen. Andererseits kann dies aber auch eine praktische Möglichkeit sein, zuvor definierte Werte zu überschreiben.

Systemzeichensätze

Nur mit der Kurzschrifteigenschaft `font` ist es möglich, Elementen bestimmte vom Betriebssystem des Benutzers verwendete Zeichensätze zuzuweisen. Das kann praktisch sein, wenn eine Webapplikation das Aussehen der grafischen Benutzeroberfläche des Benutzersystems nachahmen soll.

Hierfür gibt es eine Reihe von speziellen Schlüsselwörtern, die anstelle der sonst notwendigen Werte angegeben werden können. Die Regel, dass die Werte für `font-size` und `font-family` immer angegeben werden müssen, gilt in diesem Fall nicht. Folgende Schlüsselwörter sind möglich:

caption

Die Schrifteinstellungen für Steuerelemente wie Buttons oder Aufklappmenüs.

icon

Die Schrifteinstellungen, die zum Beschriften von Icons des Betriebssystems – wie beispielsweise von Festplatten, Ordnern und Dateien – benutzt werden.

menu

Die Schrifteinstellungen für Elemente von Aufklappmenüs und Menülisten.

message-box

Die Schrifteinstellungen für Text innerhalb von Dialogfeldern.

small-caption

Die Schrifteinstellungen für kleine Überschriften-Kontrollelemente.

status-bar

Die Schrifteinstellungen für Statusbalken von Fenstern.

Um also die systemeigene Schrift für Buttons zu verwenden, könnten Sie schreiben:

```
button {  
    font: caption;  
}
```

Erst danach ist es möglich, die Werte bestimmter Einzeleigenschaften wie `font-size` zu ändern. Wenn Sie also die Systemschrift für Buttons verwenden und diese größer als üblich darstellen wollen, reicht es nicht, zu schreiben:

```
button {  
    font: 2em caption; /* Funktioniert nicht! */  
}
```

Stattdessen müssen Sie erst das passende Schlüsselwort als einzigen Wert für `font` angeben. Danach können Sie die einzelnen Werte anpassen, wie hier gezeigt:

```
button {  
    font:      caption; /* Korrekt. Erst das Schlüsselwort ... */  
    font-size: 2em;    /* ... dann der Wert für font-size.   */  
}
```



Beachten Sie, dass die Gestaltung von Formularelementen von den Browsern unterschiedlich umgesetzt wird. Mehr dazu finden Sie in Kapitel 7, *Formulare*.

Siehe auch

Die CSS 2.1-Spezifikation zur Kurzchrifteigenschaft `font` unter <http://www.w3.org/TR/CSS2/fonts.html#font-shorthand>; ein »Schummelzettel« für `font` unter <http://www.impressivewebs.com/css-font-shorthand-property-cheat-sheet/>; Details zur Verwendung von Systemschriften per `font` in Kapitel 13 von *CSS – Das umfassende Handbuch* von Eric A. Meyer (O'Reilly 2007).

3.11 Webfonts verwenden

Problem

Sie möchten auf Ihrer Website Schriften verwenden, die mit ziemlicher Sicherheit nicht auf dem System des Benutzers installiert sind, wie Abbildung 3-6 zeigt.

Lösung 1

Verwenden Sie eine `@font-face`-Regel, um die gewünschte Schrift zu laden. Diese können Sie dann nach Bedarf mit der Eigenschaft `font-family` in Ihrem Dokument verwenden. Hier ein Beispiel für ein externes Stylesheet, das die nötigen Regeln enthält:

```
/* Generated by Font Squirrel (http://www.fontsquirrel.com) on May 14, 2011 */
@font-face {
  font-family: 'TangerineRegular';
  src: url('Tangerine_Regular-webfont.eot');
  src: url('Tangerine_Regular-webfont.eot?#iefix') format('embedded-opentype'),
        url('Tangerine_Regular-webfont.woff') format('woff'),
        url('Tangerine_Regular-webfont.ttf') format('truetype'),
        url('Tangerine_Regular-webfont.svg#TangerineRegular') format('svg');
  font-weight: normal;
  font-style: normal;
}
```

Um zu zeigen, dass es sich hierbei um ein automatisch erzeugtes Stylesheet handelt, haben wir den Kommentar am Anfang absichtlich stehen lassen. Dieses Stylesheet können Sie nun, wie in Rezept 1.1 erklärt, in Ihre Seiten einbinden:

```
<link rel="stylesheet" href="fonts/tangerine/stylesheet.css" />
```

Damit sind die Vorarbeiten bereits abgeschlossen, und Sie können den Webfont ganz normal in Ihren `font-family`-Deklarationen verwenden:

```
p {
  font-family: TangerineRegular, Helvetica, sans-serif;
}
```

Die ganze Wahrheit erfahren Sie in der Diskussion.

Lösung 2

Verwenden Sie die Google Web Fonts-API, um die `@font-face`-Regel automatisch erzeugen zu lassen und als externes Stylesheet einzubinden, wie hier gezeigt:

```
<link href="http://fonts.googleapis.com/css?family=MedievalSharp" rel="stylesheet"
type='text/css'>
```

Danach können Sie die Schrift, wie in Lösung 1, wie gewohnt in Ihren Stylesheets verwenden:

```
h1 { font-family: "MedievalSharp", arial, serif; }
```



Abbildung 3-6: Der Absatz wird in einer Schrift dargestellt, die nicht auf dem Rechner des Benutzers installiert sein muss

Bei der hier gezeigten Lösung brauchen Sie die `@font-face`-Regel nicht einmal selbst anzugeben. Stattdessen wird sie passend für den verwendeten Browser von Google dynamisch erzeugt. Sie müssen das Ganze nur noch in Form eines externen Stylesheets in Ihre Seiten einbinden. Danach kann die gewünschte Schrift per `font-family` benutzt werden, als wäre sie auf dem Rechner des Benutzers installiert.

Diskussion

Einer der größten Mängel bei der Typografie für das Web besteht in der äußerst geringen Zahl benutzbarer Schriften. Gerade einmal bei 13 Fonts kann man davon ausgehen, dass sie auf allen Benutzerrechnern installiert sind. Das ist ungefähr so, als müssten Sie sich auf die VGA-Farbpalette beschränken, obwohl Ihr Monitor mehrere Millionen Farben darstellen kann. Glücklicherweise gibt es einen Ausweg: *Webfonts*.

Zum ersten Mal wurde die Möglichkeit herunterladbarer Schriften in der Spezifikation für CSS2 definiert. Mit der Aktualisierung auf CSS 2.1 wurde diese Option jedoch wieder entfernt. Im CSS3 erleben die `@font-face`-Regeln nun ihre Wiedergeburt.

Inzwischen werden Webfonts von allen aktuellen Browsern und selbst von älteren Versionen des Internet Explorer unterstützt.³ Einer Verwendung von Webfonts steht also eigentlich nichts mehr im Wege, oder?

Leider doch. Das größte Problem besteht darin, dass es kein einheitliches Dateiformat für Webfonts gibt, das von allen Browsern unterstützt wird. Daher müssen in der `@font-face`-Regel Alternativen angegeben werden. Aus diesem Grund beschränken wir uns in diesem Rezept auf die Nutzung verschiedener Webdienste, die die Regeln für Sie erzeugen. Ein weiteres Problem stellt die rechtliche Lage dar: Da Schriften offiziell als Compu-

³ Dies gilt für IE4+, FF3.5+, Saf3.2+, Ch9+, O10.5+, iOS Saf 3.2+, O Mobile 10.0+, Android 2.1+.

terprogramme gelten, ist es nicht erlaubt, urheberrechtlich geschützte Fonts ohne entsprechende Lizenz in Ihren Seiten zu verwenden.

Glücklicherweise gibt es mittlerweile Tausende von Schriften, die Sie für private und kommerzielle Zwecke kostenlos und lizenzfrei nutzen können. Wir möchten Ihnen hier zwei Dienste vorstellen, die Ihnen die Konvertierungsarbeit in die verschiedenen Formate und die Erstellung der korrekten Regeln abnehmen.

Font Squirrel

Unter der Adresse <http://www.fontsquirrel.com/fontface> finden Sie eine Liste mit kostenlosen Schriften, die Sie auch in kommerziellen Webseiten lizenzfrei verwenden können. Um eine solche Schrift in Ihre Seiten einzubinden, gehen Sie wie folgt vor:

1. Suchen Sie sich eine Schrift aus und klicken Sie auf den Link *Get Kit*, um die Schrift auf Ihren Rechner herunterzuladen.
2. Nach dem Entpacken finden Sie auf Ihrem Rechner einen Ordner, benannt nach der heruntergeladenen Schrift. Er enthält die benötigten Schriftdateien und ein Stylesheet (*stylesheet.css*) mit den zum Einbinden nötigen @font-face-Regeln. Außerdem enthält der Ordner die Datei *demo.html*. Sie zeigt eine Vorschau auf die eingebundene Schrift.
3. Kopieren Sie den gesamten Ordner in Ihr Webverzeichnis. In unserem Fall haben wir einen Unterordner namens *fonts* erstellt, in dem wir den heruntergeladenen Ordner (in unserem Beispiel heißt dieser *Florante-at-Laura-fontfacekit*) speichern. Den Pfad müssen Sie gegebenenfalls an Ihre eigene Ordnerstruktur anpassen.
4. Binden Sie das enthaltene Stylesheet per link-Tag in den head-Teil Ihrer Seiten ein, wie hier gezeigt:

```
<link rel="stylesheet" href="fonts/Florante-at-Laura-  
fontfacekit/stylesheet.css" type="text/css" charset="utf-8">  
  
<style type="text/css">  
  h1 {  
    font: 40px "FloranteatLauraRegular", Arial, sans-serif;  
  }  
</style>
```

Nach dem Einbinden des Webfonts können Sie diesen auf die gleiche Weise verwenden wie eine der Standardschriften. Sobald ein Besucher die Seite aufruft, werden die Überschriften nun im Webfont dargestellt. Auch hier ist es wichtig, mindestens eine Fallback-Schrift (hier: Arial) bzw. eine allgemeine Schriftfamilie (z.B. sans-serif) anzugeben. Das Ergebnis sehen Sie in Abbildung 3-7.

Font Squirrel bietet außerdem die Möglichkeit, eigene Schriften hochzuladen und daraus ein neues Schriftpaket, inklusive konvertierter Fonts und Stylesheets, zu erstellen. Den @font-face-Generator finden Sie unter der Adresse <http://www.fontsquirrel.com/font-facegenerator>.



Abbildung 3-7: Die Überschrift wird in der Schrift FloranteLauraRegular dargestellt

Google Web Fonts-API

Eine weitere Möglichkeit, Webfonts zu nutzen, besteht in der Verwendung der Google Web Fonts-API. Hierbei ist es nicht nötig, Dateien herunterzuladen oder zu installieren. Gehen Sie stattdessen vor wie folgt:

1. Rufen Sie die Adresse <http://www.google.com/webfonts> auf und suchen Sie nach einer Schrift, die Ihnen gefällt.
2. Wenn Sie eine Schrift gefunden haben, die Sie verwenden möchten, klicken Sie auf den Link *Quick-use*. Falls Sie den Namen der gewünschten Schrift bereits kennen, können Sie ihn auch in das Suchfeld links oben eingeben. Danach werden die Schriften angezeigt, auf die der Suchbegriff passt.
3. Auf der folgenden Seite können Sie das gewünschte Schriftgewicht (normal, fett, etc.) und den Zeichensatz auswählen. In den meisten Fällen sollte hier *Latin* ausreichen – es sei denn, Ihre Seite soll uigurische Zeichen enthalten. Der dritte Punkt auf der Seite bietet verschiedene Optionen zum Einbetten der Schrift. Die Standardmethode verwendet ein `link`-Tag. Kopieren Sie den Code unter dem Punkt *Add this code to your website* und fügen Sie ihn im `head`-Teil Ihrer Seite ein, wie hier gezeigt:
`<link href='http://fonts.googleapis.com/css?family=Lobster' rel='stylesheet' type='text/css'>`
4. Danach können Sie die Schrift wie gewohnt im Stylesheet verwenden, zum Beispiel:

```
<style type="text/css">
  h1 {
    font-family: "Lobster", arial, serif;
  }
</style>
```

(Wenn Sie dem Autor der Schrift etwas Gutes tun wollen, klicken Sie auf die Registerkarte *Download*. Hier haben Sie die Möglichkeit, die Schrift gegen eine Spende zur weiteren Benutzung auf Ihren Rechner herunterzuladen.)

FOUT – Der Flash Of Unstyled Text

Wie Bilder und andere eingebundene Inhalte müssen Webfonts zuerst vom Browser geladen werden, damit sie in den Seiten verwendet werden können. Gleichzeitig sollen die Inhalte der Seite möglichst schnell dargestellt werden.

Das führt zwangsläufig zu einem Konflikt. Sollen mit Webfonts gestaltete Elemente bis zum vollständigen Herunterladen in einer Standardschrift dargestellt werden oder verborgen bleiben, bis die gewünschte Schrift zur Verfügung steht?

In einigen älteren Versionen von Firefox (3.5/3.6) hat man sich für die erste Lösung entschieden. Das heißt, der Text ist zwar sofort zu sehen, ändert aber seine Gestalt, sobald der Webfont zur Verfügung steht. Dies ist der berühmte »Flash Of Unstyled Text«. Bei einer schnellen Verbindung fällt das vermutlich gar nicht weiter auf. Der Webentwickler Paul Irish hat auf seiner Website eine Reihe von Lösungen gesammelt, wie sich der FOUT abmildern oder auch ganz vermeiden lässt. Die Adresse lautet <http://paulirish.com/2009/fighting-the-font-face-fout/>.

Alternativen

Neben Font Squirrel gibt es noch eine Reihe weiterer Dienste, die ebenfalls die Möglichkeit bieten, Webfonts auf die eine oder andere Weise einzubinden. Hierzu gehören beispielsweise Fontspring, der kommerzielle Ableger von Font Squirrel (<http://www.fontspring.com>), Fontdeck (<http://www.fontdeck.com>) und TypeKit (<http://www.typekit.com>). Bei den meisten dieser Websites können Sie die angebotenen Schriften kostenlos testen. Teilweise ist die Benutzung bis zu einer bestimmten Anzahl von Aufrufen sogar komplett frei.

Siehe auch

Die FontSquirrel-Website unter <http://www.fontsquirrel.com>; Google Webfonts unter <http://www.google.com/webfonts>; die CSS3-Spezifikation zur @font-face-Regel unter <http://www.w3.org/TR/css3-fonts/#font-face-rule>; den Blogartikel *Bulletproof @font-face syntax* von Paul Irish, der beschreibt, wie Sie Ihre @font-face-Regeln auch selbst schreiben können, unter <http://tr.im/Gxhf>; den englischen Wikipedia-Artikel zu Webfonts unter http://en.wikipedia.org/wiki/Web_typography#Web_fonts; eine Liste mit weiteren Websites, die Webfonts anbieten, finden Sie unter <http://webfonts.info>.

3.12 Die Textausrichtung festlegen

Problem

Sie wollen selbst bestimmen, ob der Text Ihrer Webseiten links- oder rechtsbündig, zentriert oder im Blocksatz dargestellt wird.

Lösung

Verwenden Sie die Eigenschaft `text-align` mit einem passenden Wert:

```
h1 { text-align: center; }  
p  { text-align: justify;}
```

Diese beiden Regeln sorgen dafür, dass Überschriften erster Ordnung zentriert dargestellt werden, während der Text innerhalb von Absätzen im Blocksatz erscheint.

Diskussion

Im Gegensatz zur horizontalen Zentrierung von Blockelementen (siehe Rezept 2.24) wird die Ausrichtung von Text in Inline-Elementen mit der Eigenschaft `text-align` gesteuert. Wie in der Lösung zu sehen, ist der Gebrauch relativ unkompliziert. Die Namen der Werte orientieren sich daran, auf welche Weise der Text ausgerichtet werden soll.

Folgende Werte sind für `text-align` möglich:

left

Der Text wird linksbündig ausgerichtet. Dies ist der Standardwert.

right

Der Text wird rechtsbündig ausgerichtet.

center

Der Text wird zentriert dargestellt.

justify

Der Text wird im Blocksatz dargestellt.

(Soll ein Element den Wert von seinen Vorfahren erben, verwenden Sie den Wert `inherit`).

Das einzige Problemkind ist der Wert `justify`. Die Darstellung von Text im Blocksatz hängt stark davon ab, wie der Browser die Abstände berechnet. Während bei »echten« Textverarbeitungsprogrammen und im Schriftsatz für gedruckte Medien Wort- und Buchstabenabstände variiert werden, verändern Browser oft nur die Abstände zwischen den Wörtern. Zudem beherrschen nur die wenigsten Browser die automatische Silbentrennung (siehe Abbildung 3-8).

Als Ergänzung zum Blocksatz mit `text-align: justify` bietet sich daher die Verwendung von »weichen Trennstrichen« an, wie in Lösung 3 von Rezept 3.13 beschrieben.

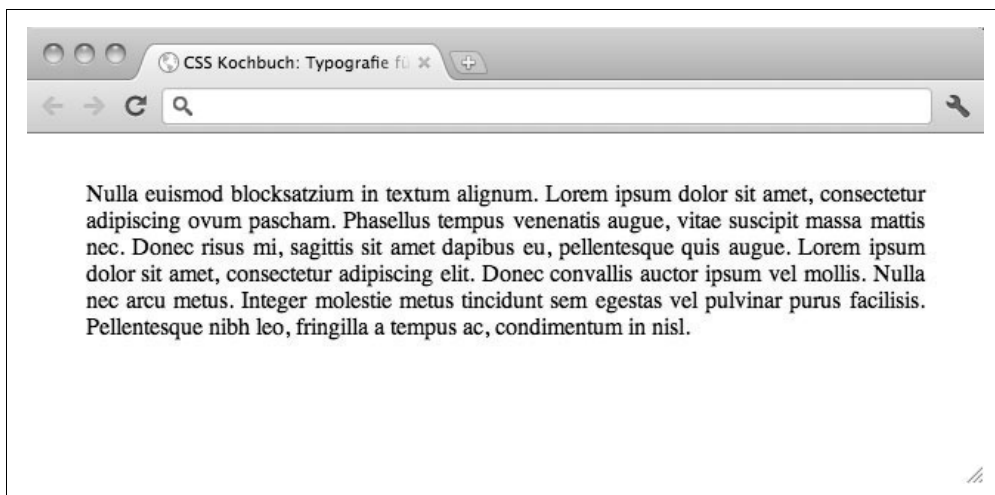


Abbildung 3-8: Ein Absatz wird im Blocksatz dargestellt

Siehe auch

Die CSS 2.1-Spezifikation zur Eigenschaft `text-align` unter <http://www.w3.org/TR/CSS21/text.html#alignment-prop>; Informationen zum Zentrieren verschiedener anderer Elemente innerhalb einer Webseite finden Sie in den Rezepten 2.24 und 2.25; die Rezepte 3.13 zur manuellen Silbentrennung und 3.25 zur Änderung der Schreibrichtung.

3.13 Zeilenumbrüche steuern

Problem

Ein Blockelement (zum Beispiel ein Absatz) erhält eine feste Breite. Ohne weitere Angaben würden längere Wörter über die Grenzen des Elements hinausragen, wie in Abbildung 3-9 gezeigt wird. Sie möchten selbst steuern, wie die Zeilen umbrochen (getrennt) werden, wenn der horizontale Platz nicht ausreicht.

Lösung 1

Verwenden Sie die CSS3-Eigenschaft `word-wrap`, um lange Wörter bei Bedarf »auf die harte Tour« zu umbrechen, wie hier gezeigt:

```
.schmal {  
  width: 150px;  
  word-wrap: break-word;  
}
```

Hier wurde für Elemente mit der Klasse `schmal` eine Breite von 150 Pixeln festgelegt. Ohne weitere Angaben würden besonders lange Wörter möglicherweise darüber hinaus-

ragen, wie in Abbildung 3-9 zu sehen. Durch den Wert `break-word` für die Eigenschaft `word-wrap` werden lange Wörter automatisch umbrochen. Allerdings werden hierbei weder Silbengrenzen berücksichtigt noch Trennstriche eingefügt.

Lösung 2

Verwenden Sie die Eigenschaft `overflow-x` mit dem Wert `auto`:

```
.schmal {  
  width: 150px;  
  overflow-x: auto;  
}
```

Durch den Wert `auto` für die Eigenschaft `overflow-x` wird automatisch ein horizontaler Scrollbalken eingefügt (siehe Abbildung 3-9). Alternativ können Sie zu breiten Text auch ausdrücklich verbergen, indem Sie `overflow-x` den Wert `hidden` geben.

Lösung 3

Versehen Sie den Text dort, wo er gegebenenfalls umbrochen werden darf, mit einem sogenannten »weichen« Trennzeichen (Soft Hyphen, `­`), zum Beispiel:

```
<p class="schmal">Ein Absatz mit langen Wörtern: Ab&shy;fluss&shy;rohr&shy;ver&shy;  
stop&shy;fung&shy;ge&shy;fahr, Do&#173;nau&#173;dampf&#173;schiff&#173;fahrts&#173;  
ka&#173;pi&#173;täns&#173;pa&#173;tent&#173;amt.</p>
```

Hier wurde überall dort, wo eine Silbentrennung möglich sein soll, ein weiches Trennzeichen eingefügt (Abbildung 3-9). Das sieht nicht nur furchtbar aus, sondern es ist nicht einmal eine CSS-Lösung. Wir haben sie hier dennoch vorgestellt, weil sie momentan die einzig verlässliche Möglichkeit bietet, Wörter zu trennen. Bevor Sie jetzt drei Wochen am Rechner sitzen, um jeden Text anzupassen, lesen Sie bitte erst die Diskussion.

Diskussion

Bei Elementen mit fester Breite kann es passieren, dass darin enthaltener Text über die Grenzen des Elements hinausläuft. Um das zu verhindern, gibt es verschiedene Möglichkeiten. Alle haben ihre Eigenheiten, Vor- und Nachteile.

word-wrap

Im *Text*-Modul der CSS3-Spezifikation werden mehrere neue Eigenschaften definiert, die für bessere Zeilenumbrüche sorgen sollen. Hierbei ist `word-wrap` eigentlich als »Notfalleigenschaft« definiert, die zum Einsatz kommen soll, wenn andere Möglichkeiten fehlschlagen. Wir verwenden in der ersten Lösung den Wert `break-word`. Dieser erzwingt einen Zeilenumbruch, sobald der Text zu lang zu werden droht. So erzeugte Umbrüche besitzen keine Trennstriche und orientieren sich auch nicht an Silbengrenzen. Der Standardwert für diese Eigenschaft ist `normal`. Er sorgt dafür, dass Umbrüche nur an Wortgrenzen und anderen erlaubten Stellen vorgenommen werden. (Der dritte mögliche Wert, `hyphenate`, wurde beim Schreiben dieses Buchs noch nicht unterstützt.)

umgehen, indem Sie für das Element einen unteren Innenabstand von 20 Pixeln (der üblichen Breite eines Scrollbalkens im IE) definieren. Die zusätzliche Regel sollte nach Möglichkeit mit einem Conditional Comment (siehe Rezept 13.13) umgeben werden, damit andere Browser sie nicht sehen:

```
<!--[if lte IE 9]>
<style type="text/css">
.schmal {
    overflow-y:    hidden;
    padding-bottom: 20px;
}
<![endif-->
```

Weiche Trennzeichen (Soft Hyphens)

Ausnahmsweise stellen wir hier eine Lösung vor, die nicht auf CSS basiert, sondern im HTML-Quellcode umgesetzt wird. Die sogenannten weichen Trennzeichen (Soft Hyphens) werden im normalen Text nicht angezeigt, sondern dienen dem Browser als Anhaltspunkte, an welcher Stelle gegebenenfalls ein Zeilenumbruch eingefügt werden darf. Freundlicherweise wird dann auch gleich der passende Trennstrich mit ausgegeben, wie in Abbildung 3-9 (siehe oben) zu sehen ist.

Die Verwendung von weichen Trennzeichen wird von allen aktuellen Browsern unterstützt und korrekt dargestellt, wobei das manuelle Einfügen dieses Zeichens (­ bzw. ­) ziemlich umständlich und fehlerträchtig ist. Zum Glück gibt es einige web-basierte Werkzeuge, die Ihnen die Arbeit abnehmen.⁴ (Einen Link finden Sie im Abschnitt »Siehe auch«.)

Bevor Sie jetzt allerdings sämtliche Texte Ihrer Website mit weichen Trennzeichen vollpacken, sollten Sie überlegen, ob dies wirklich notwendig ist. Oft reicht es aus, kürzere Passagen, die im Blocksatz (siehe Rezept 3.12) dargestellt werden sollen, auf diese Weise zu behandeln.

Siehe auch

Das Text-Modul der CSS3-Spezifikation zur Eigenschaft `word-wrap` unter <http://www.w3.org/TR/css3-text/#word-wrap>; den Trennstrich-Generator unter <http://yellowgreen.de/soft-hyphenation-generator> (deutsch); das Grundrezept 2.22 zur Verwendung der Eigenschaften `overflow-x` und `overflow-y`; das Rezept 13.13 zur Verwendung von Conditional Comments.

⁴ Der Gegenpart zum weichen Trennzeichen ist übrigens das nicht umbrechende Leerzeichen ` ` bzw. ` `.

3.14 Text mit Über-, Unter- oder Durchstreichungen versehen

Problem

Sie wollen bestimmte Textteile mit Über-, Unter- oder Durchstreichungen versehen, wie in Abbildung 3-10 gezeigt.

Lösung 1

Verwenden Sie die Eigenschaft `text-decoration` mit einem passenden Wert:

```
.drunter { text-decoration: underline; }  
.drueber { text-decoration: overline; }  
.durch { text-decoration: line-through; }  
a:link { text-decoration: none; }
```

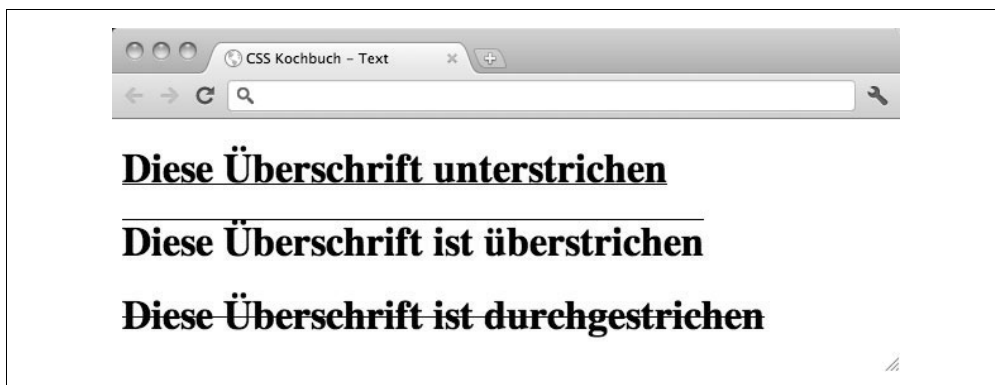


Abbildung 3-10: Unter-, über- und durchgestrichener Text

Dieser Code zeigt die vier wichtigsten Werte für die Eigenschaft `text-decoration`. Besonders der letzte Wert (`none`) wird gern verwendet, um Links mit eigenen Stildefinitionen zu versehen (siehe Rezept 6.1). (Wollen Sie, dass das Element den Wert für `text-decoration` von einem Vorfahrenelement erbt, verwenden Sie den Wert `inherit`.)



Sie können der Eigenschaft `text-decoration` übrigens mehr als einen Wert übergeben, um Text beispielsweise über- und unterstrichen darzustellen.

Diskussion

Der Eigenschaft `text-decoration` können vier verschiedene Werte zugewiesen werden (siehe Tabelle 3-3).⁵

⁵ Also gut – es gibt noch einen weiteren Wert: `blink`. Dieser simuliert das im vorigen Jahrtausend teilweise recht beliebte `<blink>`-Tag. Vermutlich haben sich die Autoren der CSS 2.1-Spezifikation hier einen Spaß gemacht, denn die Unterstützung für `blink` ist den Browserherstellern überlassen, das heißt: mal blinkt der Text – mal nicht.

Tabelle 3-3: Mögliche Werte für die Eigenschaft *text-decoration*

Werte für <i>text-decoration</i>	Ergebnis
<code>underline</code>	Unter dem Text wird eine Linie gezogen (Unterstreichung).
<code>overline</code>	Über dem Text wird eine Linie gezogen (Überstreichung).
<code>line-through</code>	Der Text wird durchgestrichen.
<code>none</code>	Der Text erhält keine zusätzlichen Verzierungen.

Definieren Sie für ein Element einen passenden Wert, und das Element wird unter-, über- oder durchgestrichen dargestellt. Hierbei gibt es allerdings eine Besonderheit: Angenommen, Sie haben den folgenden HTML-Code:

```
<p>Dieser Text wird unterstrichen dargestellt. <em>Und dieser Teil auch.</em> Danach geht es unterstrichen weiter.</p>
```

Dafür definieren Sie die folgenden CSS-Regeln:

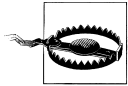
```
p { text-decoration: underline; }
em { text-decoration: none !important; }
```

Eigentlich sollte man annehmen, dass unter dem `em`-Element keine Unterstreichung zu sehen ist. Das haben wir schließlich ausdrücklich so festgelegt. Abbildung 3-11 zeigt, dass es anders ist. Tatsächlich ist der per `em` hervorgehobene Teil nicht unterstrichen, sondern der umgebende Absatz.



Abbildung 3-11: Die Unterstreichung ist trotz der `!important`-Anweisung unterhalb des `em`-Elements zu sehen, aber tatsächlich handelt es sich hier um die Unterstreichung des Absatzes

Was Sie hier sehen, ist aber nicht die Unterstreichung des Absatzes, der das `em`-Element enthält. Wenn Sie das nächste Mal an Ihrem Verstand zweifeln, wissen Sie, dass es nicht an Ihnen liegt, sondern an einer Eigenart von CSS.



Bei der Gestaltung Ihrer Seiten sollten Sie nicht vergessen, dass Unterstreichungen im Web primär für die Markierung von Hyperlinks verwendet werden. Wenn Sie die Unterstreichungen von Links entfernen oder andere Elemente mit Unterstreichungen versehen, sollten Sie auf jeden Fall dafür sorgen, dass Links auch weiterhin als solche intuitiv erkennbar sind.

Unter- und Überstreichungen per `border-bottom` und `border-top`

Alternativ zu `text-decoration`: `underline` bzw. `overline` können Sie auch die Eigenschaften `border-top` bzw. `border-bottom` einsetzen, z.B.:

```
.drunter {  
    border-bottom: 3px dotted red;  
}  
.drueber {  
    border-top: 1px solid blue;  
}
```

Hier verwenden wir die Kurzschrifteigenschaften für bestimmte Rahmenseiten, um Absätze, deren `class`-Attribut den Wert `drunter` hat, mit einer rot gepunkteten 3 Pixel breiten Unterstreichung zu versehen. Absätze mit der Klasse `drueber` erhalten eine 1 Pixel breite blaue Überstreichung.

Während bei `text-decoration: underline` die Unterstreichung auf der Grundlinie angezeigt wird, erscheint sie bei der Verwendung von `border-bottom` unterhalb aller Buchstaben (z.B. p, q oder g). Im Gegensatz zu `text-decoration` haben Sie bei der Verwendung der Eigenschaften `border-bottom` bzw. `border-top` deutlich mehr Einstellungsmöglichkeiten zur Verfügung.

Bei dieser Methode sollten Sie beachten, dass sich die Ausschmückungen über die gesamte Breite des Elements erstrecken. Bei Inline-Elementen wie `p`, `em` oder `span` entspricht dies auch dem erwarteten Verhalten. Allerdings sind Überschriften (`h1`, `h2` usw.) als Blockelemente definiert.

Siehe auch

Die CSS 2.1-Spezifikation zur Eigenschaft `text-decoration` unter <http://www.w3.org/TR/CSS2/text.html#lining-striking-props>; das Rezept 1.16 zum Boxmodell; das Rezept 2.7 zur Definition von Rahmen.

3.15 Grafische Unterstreichungen verwenden

Problem

Anstelle der in CSS möglichen Unterstreichungen per `text-decoration` oder `border-top`/`border-bottom` möchten Sie die Gestaltung selbst bestimmen

Lösung

Definieren Sie ein eigenes Hintergrundbild für die Passage und platzieren Sie es an der gewünschten Stelle:

```
h1.lachs {  
    background-image:    url("bilder/lachs.gif");  
    background-repeat:   repeat-x;  
    background-position: 0 100%;  
    padding-bottom:      10px;  
}
```

Oder als Kurzschrifteigenschaft:

```
h1.lachs {  
    background:    url("bilder/lachs.gif") repeat-x 0 100%;  
    padding-bottom: 10px;  
}
```

Das Ergebnis sehen Sie in Abbildung 3-12.



Abbildung 3-12: Die Verwendung von `background-image` zur Erstellung eigener Unterstreichungen

Diskussion

Mit dieser Lösung können Sie eine beliebige Grafik als Unterstreichung benutzen. Da das in diesem Beispiel verwendete Bild nur eine Breite von knapp 60 Pixeln hat, wiederholen wir es per `background-repeat: repeat-x; horizontal`. Mit der Eigenschaft `background-position` platzieren wir die Grafik in der linken unteren Ecke der *Padding-Box* (siehe Rezept 1.16). Da das Bild eine Höhe von 20 Pixeln hat, verschieben wir es per `padding-bottom` ein Stück nach unten, damit sich unsere »Unterstreichung« nicht mit dem Text überschneidet. Vermutlich müssen Sie diesen Wert abhängig von der Höhe Ihres eigenen Hintergrundbilds ein wenig anpassen.

Wollen Sie anstelle der Unterstreichung eine Überstreichung erzeugen, nehmen Sie für `background-position` einfach die Werte `0 0` (oder auch `left top` – wie Sie wollen). Muss das Bild noch weiter nach oben verschoben werden, verwenden Sie in diesem Fall `padding-top` mit einem passenden Wert.

Soll der Text durchgestrichen erscheinen, wählen Sie für `background-position` die Werte `0 50%` (bzw. `left center`). In diesem Fall ist keine `padding`-Eigenschaft nötig.

Siehe auch

Die CSS 2.1-Spezifikation zu den `background`-Eigenschaften unter <http://www.w3.org/TR/CSS2/colors.html#background-properties>; das Grundrezept 2.8 zur Verwendung von Hintergrundbildern.

3.16 Sonderzeichen aus einem anderen Zeichensatz einfügen

Problem

Sie wollen bestimmte Zeichen (z.B. das Ampersand-Zeichen `&`) aus einem anderen Zeichensatz verwenden.

Lösung

Umgeben Sie das gewünschte Zeichen im HTML-Quelltext mit `span`-Tags und definieren Sie für diesen Textteil eine passende Schrift, wie hier gezeigt:

```
span.amp {  
    font-family: Baskerville, "Book Antiqua", serif;  
    font-style: italic;  
    font-weight: normal;  
}
```

Das ist der HTML-Code:

```
<h1>Webdesign mit HTML5 <span class="amp">&</span> CSS3</h1>
```

Hier benutzen wir die kursive Variante der Schrift Baskerville, um ein optisch möglichst ansprechendes Ampersand-Zeichen in unseren Text einzubinden (siehe Abbildung 3-13).

Da wir das Zeichen in unserem Beispiel in einer Überschrift verwenden, die standardmäßig fett dargestellt wird, müssen wir das Schriftgewicht (`font-weight`) zudem noch auf `normal` zurücksetzen. Baskerville besitzt nämlich einen eigenen Zeichensatz für kursiv und fett dargestellte (bold italic) Schrift, in der das Ampersand-Zeichen ziemlich langweilig daherkommt.



Abbildung 3-13: Das Ampersand-Zeichen wird in einer anderen Schrift dargestellt

Diskussion

Wie in Rezept 3.5 bereits angesprochen, enthalten manche Schriften einen speziellen kursiven Zeichensatz. Hier lassen sich einige besonders schöne Exemplare des Ampersand-(Kaufmanns-Und)Zeichens finden – nicht weil die Zeichen kursiv sind, sondern weil sie oft speziell gestaltet werden.

Leider gibt es in CSS noch keinen Mechanismus, einzelne Wörter über spezielle Selektoren auszuwählen. Daher kommen Sie in diesem Fall nicht umhin, das gewünschte Zeichen mit einem `span`-Element zu umgeben, das Sie dann im Stylesheet mit eigenen Regeln versorgen können. Diese Methode ist natürlich nicht auf einzelne Zeichen beschränkt. Sie können beliebig lange Textpassagen kennzeichnen und ihnen eigene Stile zuweisen.

Siehe auch

Den Artikel von Dan Cederholm über die Verwendung des bestmöglichen Ampersand-Zeichens unter <http://simplebits.com/notebook/2008/08/14/ampersands-2/>; die Rezepte zur Verwendung der Eigenschaften `font-family` (Rezept 3.1), `font-weight` (Rezept 3.4) und `font-style` (Rezept 3.5) in diesem Kapitel.

3.17 Typografisch korrekte Anführungszeichen verwenden

Problem

Sie möchten eine Textpassage (z.B. ein Zitat) mit typografisch korrekten Anführungszeichen versehen, haben aber keinen Zugriff auf den HTML-Code. Dies soll auch bei verschachtelten Zitaten funktionieren (siehe Abbildung 3-14).

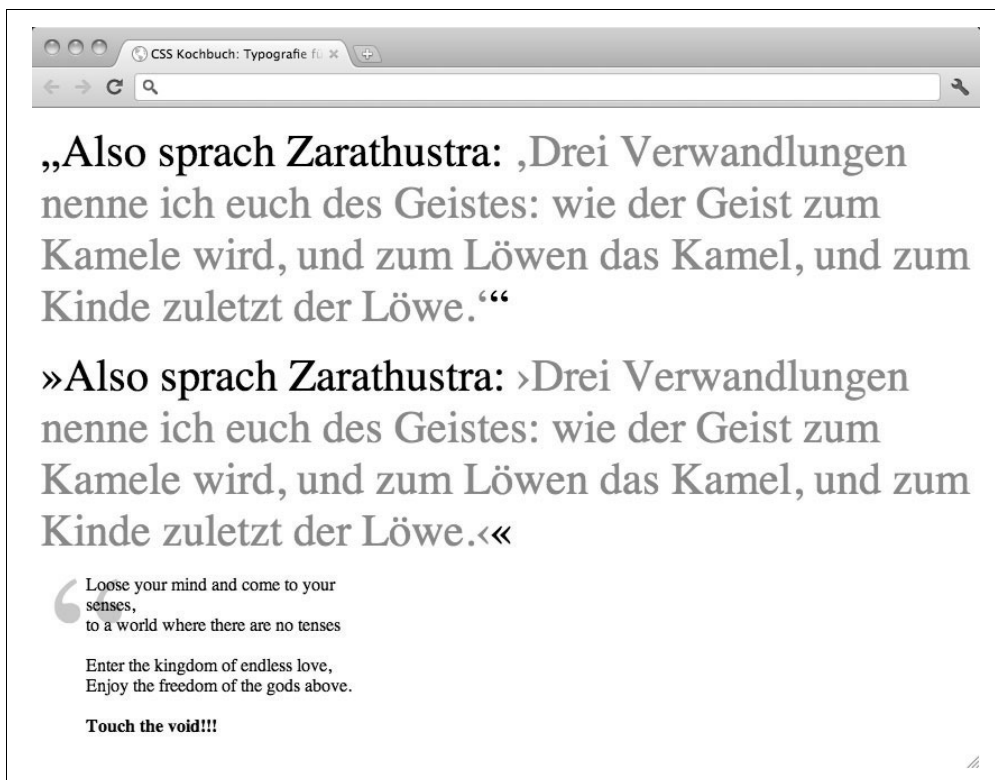


Abbildung 3-14: Typografisch korrekte Anführungszeichen

Lösung

Verwenden Sie die CSS-Pseudoelemente `:before` und `:after` zusammen mit der Eigenschaft `content`, um vor und nach der Passage die passenden Zeichen als UTF-8-Sequenz einzufügen:

```
/* Typografisch korrekte Anführungszeichen */
```

```
q:before { content: "\201E"; } /* „ */
q:after  { content: "\201C"; } /* “ */
q q:before { content: "\201A"; } /* , */
q q:after  { content: "\2018"; } /* ‘ */
```

```
/* Französische Version */
```

```
q:before { content: "\00BB"; } /* » */
q:after  { content: "\00AB"; } /* « */
q q:before { content: "\203A"; } /* › */
q q:after  { content: "\2039"; } /* ‹ */
```

Dazu passt folgender HTML-Code:

```
<p><q>Also sprach Zarathustra: <q>Drei Verwandlungen nenne ich euch des Geistes: wie der Geist zum Kamele wird, und zum Löwen das Kamel, und zum Kinde zuletzt der Löwe.</q></q></p>
```

Die Regeln folgen alle dem gleichen Schema: Sie beginnen mit dem Elementselektor für Zitate im Fließtext (q), gefolgt von einem Pseudoelement (:before bzw. :after). Um verschachtelte Zitate gesondert zu gestalten, kommt hier der Nachfahrenselektor zum Einsatz (q q). Innerhalb der Regel verwenden wir die Eigenschaft content, um vor bzw. nach dem q-Element die gewünschten Anführungszeichen zu erzeugen, die hier als Unicode-Zeichenreferenz angegeben wurden. Der Kommentar nach der Regel zeigt das jeweilige Zeichen im Klartext.

Diskussion

Standardmäßig legt der Browser fest, welche Anführungszeichen für Kurzzitate (q) verwendet werden. Während Webkit-basierte Browser (Safari, Chrome) für Anfang und Ende eines Zitats ein nicht typografisches Anführungszeichen (") verwenden, benutzen Internet Explorer 8+ und Gecko-basierte Browser (Firefox & Co.) von sich aus typografische Anführungszeichen.



Bei der Verwendung des Elements cite werden vom Browser keine Zeichen eingefügt. Das müssen Sie bei Bedarf selbst erledigen.

Wenn Ihnen eine browserübergreifende einheitliche Darstellung wichtig ist, haben Sie zwei Möglichkeiten: die automatische Darstellung von Anführungszeichen mit einer »Reset«-Regel komplett zu unterbinden (siehe Rezept 13.16) oder selbst zu definieren, welche Zeichen benutzt werden, wie in unserer Lösung gezeigt.

Anders als im HTML-Code, in dem Sie die üblichen HTML-Zeichen-Entities (z.B. „ oder “) verwenden können, muss das Zeichen im Stylesheet entweder im Klartext oder als hexadezimale Unicode-Zeichenreferenz (mit vorangestelltem Backslash) angegeben werden. Da der Browser oft nur raten kann, mit welchem Zeichensatz Ihr Stylesheet kodiert ist (z.B. ISO Latin 1, Mac OS Roman oder etwas noch Exotischeres), ist es sicherer, Sonderzeichen in der Unicode-Schreibweise anzugeben, damit sie auch wie gewünscht dargestellt werden.



Leider unterstützt der Internet Explorer erzeugte Inhalte und Pseudoelemente erst ab Version 8. In Kapitel 13, *Tipps und Tricks*, finden Sie einige Möglichkeiten, wie Sie älteren IE-Versionen das Verhalten trotzdem beibringen können.

Und was ist mit »quotes«?

Wenn Sie das Rezept aufmerksam gelesen haben, wird Ihnen aufgefallen sein, dass wir Ihnen nicht alles erzählt haben. Tatsächlich definiert die CSS 2.1-Spezifikation die Eigenschaft `quotes`, mit der Sie die Anführungszeichen eigentlich auch in einem Aufwasch angeben können, wie hier:

```
[lang="de"] { quotes:"\201E" "\201C" "\201A" "\2018"; }
[lang="en"] { quotes:"\201C" "\201D" "\2018" "\2019"; }
[lang="fr"] { quotes:"\00AB\00A0" "\00A0\00BB" "\2039\00A0" "\00A0\203A"; }
```

Hier werden die Anführungszeichen abhängig vom Wert des Attributs `lang` festgelegt. Danach können Sie die Werte für `content` auch wie folgt angeben:

```
q:before { content: open-quote; }
q:after  { content: close-quote; }
```

Das Problem ist Apples Browser Safari, der eigentlich als besonders standardkonform gilt. Bei der Darstellung von Anführungszeichen ist er jedoch ziemlich renitent. Daher geben wir in unserer Lösung die Anführungszeichen einfach direkt als Wert für `content` an. Das bedeutet zwar mehr Schreiarbeit, aber auch mehr zufriedene Benutzer.

Siehe auch

Die CSS 2.1-Spezifikation zu den Pseudoelementen `:before` und `:after` sowie der Eigenschaft `content` unter <http://www.w3.org/TR/CSS21/generate.html>; das Rezept 1.14 zur Verwendung von Pseudoelementen; den deutschen Wikipedia-Artikel zu Anführungszeichen unter <http://de.wikipedia.org/wiki/Anführungszeichen>; den englischen Wikipedia-Artikel zur Verwendung von HTML-Zeichen-Entities unter http://en.wikipedia.org/wiki/List_of_XML_and_HTML_character_entity_references; die CSS 2.1-Spezifikation zum Schützen (Escaping) von Sonderzeichen unter <http://www.w3.org/TR/CSS2/syndata.html#escaping>.

3.18 Hängende Initialen zu Beginn eines Absatzes einfügen

Problem

Ein Absatz soll mit einer sogenannten *hängenden Initialen* beginnen. Als *Initialen* bezeichnet man besonders ausgeschmückte Buchstaben am Wort- oder Absatzanfang, die deutlich größer, oftmals auch mehrere Zeilen hoch, dargestellt werden.

Lösung

Benutzen Sie das Pseudoelement `:first-letter`, um den ersten Buchstaben des Absatzes mit einer speziellen Stildefinition zu versehen (siehe Abbildung 3-15):

```
p:first-letter {
  font-size: 3em;
  float:     left;
```

```
margin: 0.1em 0.2em -0.1em 0;
padding: 0 0.1em;
}
```

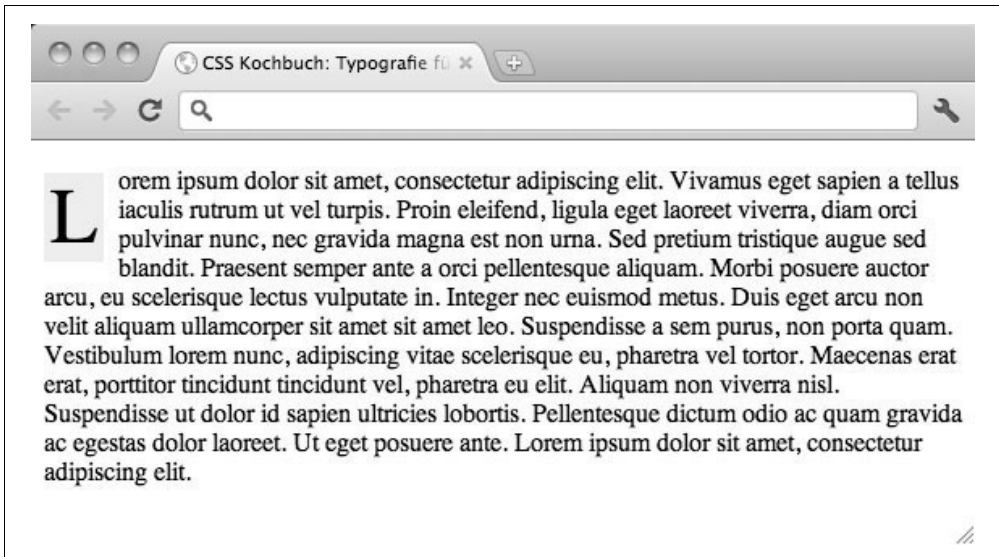


Abbildung 3-15: Eine einfache Initiale

Diskussion

Die CSS-Spezifikation bietet mit dem Pseudoelement `:first-letter` eine einfache Möglichkeit, den ersten Buchstaben eines Absatzes als hängendes Initial darzustellen. In dieser Lösung verwenden wir die Eigenschaft `float` mit dem Wert `left`, um den ersten Buchstaben innerhalb des Absatzes nach links zu verschieben und vom übrigen Text umfließen zu lassen. Dies hat den Vorteil, dass Sie sich um Probleme mit der Zeilenhöhe keine Sorgen zu machen brauchen.



Gemäß der CSS-Spezifikation kann `:first-letter` nur auf Buchstaben und Zahlen angewendet werden. Die meisten Browser bis auf Firefox lassen aber auch HTML-Entities wie `&` zu.

Zu Beginn definieren wir für den ersten Buchstaben eine Schriftgröße von 3em-Einheiten, damit er ausreichend zur Geltung kommt. Mit der Kurzschrifteigenschaft `margin` schaffen wir oberhalb und rechts der Initiale etwas Platz. Der dritte Wert für `margin` (`-0.1em`) sorgt dafür, dass die Zeilen unterhalb des ersten Buchstabens nicht zu weit unten beginnen. Der vierte Wert (`0`) besagt, dass auf der linken Seite kein Außenabstand eingefügt werden soll.

Die Kurzschrifteigenschaft `padding` wird verwendet, damit die Initiale horizontal nicht zu eingezwängt erscheint. Ober- und unterhalb wird der Innenabstand auf null gesetzt.

Möglicherweise müssen Sie mit den Werten für `margin` und `padding` ein wenig herumspielen, um die für Ihr Design optimalen Einstellungen zu finden.

Siehe auch

Die CSS 2.1-Spezifikation zum Pseudoelement `:first-letter` unter <http://www.w3.org/TR/CSS21/selector.html#x52>; weitere Informationen zu Initialen finden Sie unter <http://fonts.lordkyl.net/fonts.php?category=vers> und <http://bweaver.net/wrangling-drop-caps>.

3.19 Bilder als Initialen einsetzen

Problem

Für die Gestaltung einer Initiale soll eine Grafik eingesetzt werden.

Lösung

Verwenden Sie, wie im vorigen Rezept beschrieben, das Pseudoelement `:first-letter`, um den ersten Buchstaben des Absatzes zu selektieren und mit einem eigenen Hintergrundbild zu versehen (siehe Abbildung 3-16):

```
p:first-letter {
  text-indent: -9999em;
  float:      left;
  display:    block;
  height:     87px;
  width:      104px;

  background-image: url("bilder/initial_1.png");
  background-repeat: no-repeat;

  margin: 0 3px -5px 0;
}
```

Diskussion

Der erste Schritt dieser Lösung besteht darin, ein Bild für die Initiale zu erstellen. Danach notieren Sie sich dessen Breite und Höhe. In unserem Beispiel ist das Bild 104 mal 87 Pixel groß.

Als Nächstes verwenden Sie die »Off-Left«-Technik, um den ersten Buchstaben aus dem sichtbaren Bereich des Browserfensters (dem *Viewport*) herauszuschieben. Hierfür setzen Sie den Wert der Eigenschaft `text-indent` auf `-9999em`. Im Gegensatz zur Verwendung von `display: none` oder `visibility: hidden` bleibt der erste Buchstabe bei dieser Methode für Menschen mit Sehproblemen, die z.B. ein Bildschirmlesegerät oder eine Braillezeile benutzen, weiter zugänglich.

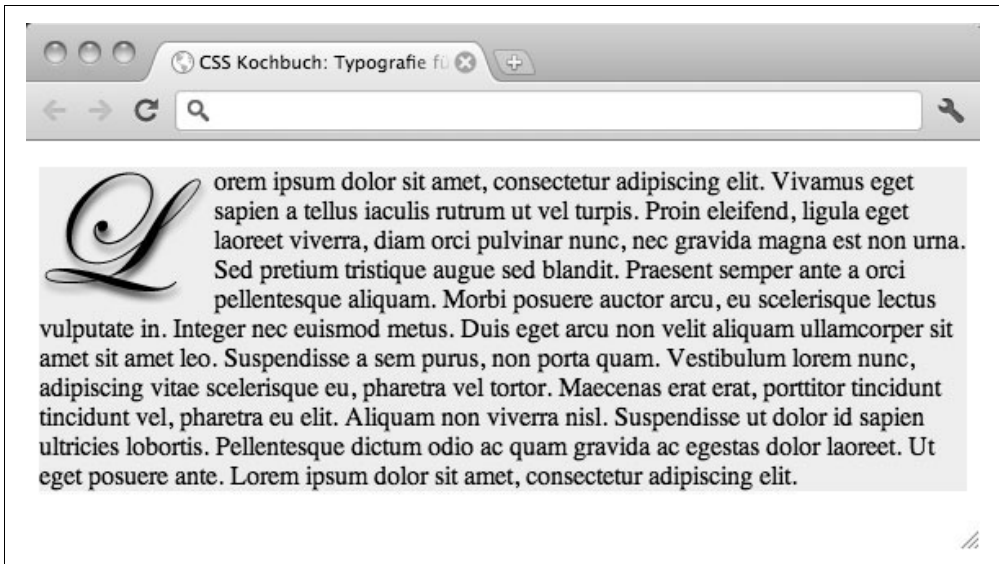


Abbildung 3-16: Die Benutzung einer Grafik als Initiale

Jetzt können wir das Pseudoelement als Float definieren, wodurch es vom übrigen Text umflossen wird. Standardmäßig ist `:first-letter` als Inline-Element definiert. Um das Bild genau einpassen zu können, definieren wir den ersten Buchstaben per `display: block` als Blockelement. Dadurch können wir eine Höhe (`height`) und eine Breite (`width`) festlegen und zudem die Außenabstände per `margin` anpassen.

Danach platzieren Sie das Bild im Hintergrund von `:first-letter` und stellen dabei sicher, dass das Bild nicht wiederholt wird. Dazu erhält die Eigenschaft `background-repeat` den Wert `no-repeat`.

Am Schluss kommt das Feintuning: Mit der Eigenschaft `margin` schaffen wir rechts der Initiale 3 Pixel Platz, während der Leerraum unterhalb der Initiale durch den Wert `-5px` verringert wird. Diese Werte hängen vom verwendeten Bild ab. Daher werden Sie vermutlich ein wenig experimentieren müssen, um die optimalen Einstellungen zu finden.

Siehe auch

Die CSS 2.1-Spezifikation zum Pseudoelement `:first-letter` unter <http://www.w3.org/TR/CSS2/selector.html#first-letter>; das Rezept 3.18 zum Erstellen einfacher Initialen; die Website von Jens O. Meiert mit einer umfassenden Liste von Methoden zur Textersatzung unter <http://meiert.com/de/publications/articles/20050513/>; das Rezept 4.6 zur Verwendung der »Off-Left«-Methode zur Textersatzung.

3.20 Stile für die erste Absatzzeile definieren

Problem

Sie wollen die erste Zeile eines Absatzes speziell gestalten, ohne hierfür zusätzlichen Markup-Code schreiben zu müssen.

Lösung

Verwenden Sie die Pseudoklasse `:first-line`, um der ersten Textzeile eines Elements eigene Stile zuzuweisen. Wollen Sie die erste Zeile ein- oder ausrücken, verwenden Sie die Eigenschaft `text-indent` ohne die Pseudoklasse:

```
p:first-line {  
  font-variant: small-caps;  
  font-weight: bolder;  
}  
  
p {  
  text-indent: 1em;  
}
```

Das Ergebnis sehen Sie in Abbildung 3-17.

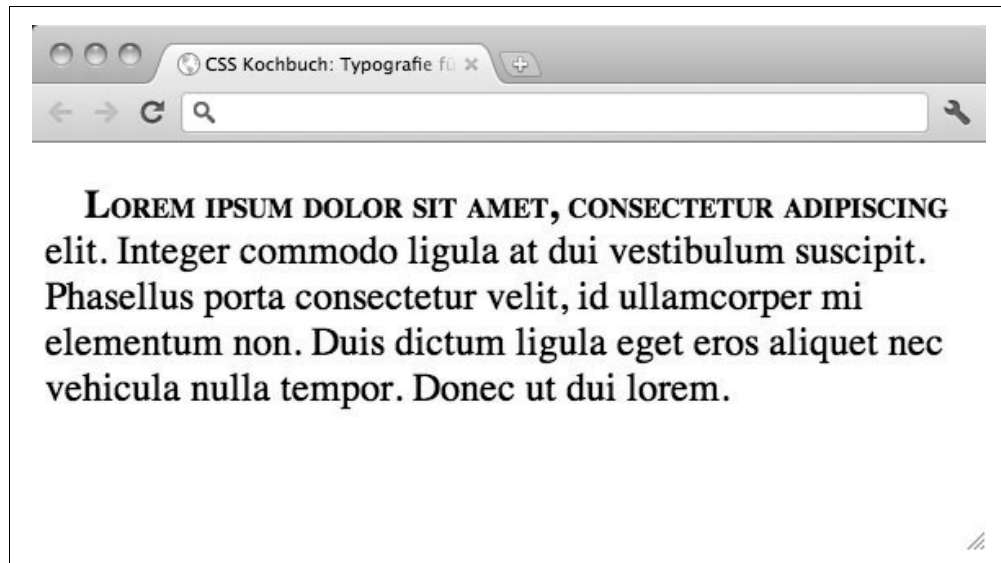


Abbildung 3-17: Die erste Textzeile wird eingerückt und in fett gestalteten Kapitälchen dargestellt

Diskussion

Vielleicht fragen Sie sich, warum es für die erste Zeile eines Elements ein spezielles Pseudoelement gibt. Man könnte sie doch einfach mit `span`-Tags umgeben und dann gestalten, oder?

Leider ist die Sache nicht ganz so einfach, da sich die Länge einer Zeile mit der Breite des Browserfensters und der verwendeten Schriftgröße verändern kann. Es gibt also keine Möglichkeit, die tatsächlich im Browser des Benutzers verwendete Zeilenlänge auf einfache Weise zu ermitteln.



Ähnlich wie die Eigenschaft `text-align` muss auch `text-indent` für das umgebende Element (den Absatz) definiert werden, um zu funktionieren.

Das Pseudoelement `:first-line` schlägt hier also zwei Fliegen mit einer Klappe: Zum einen ermöglicht es die zuverlässige Gestaltung der ersten Zeile, ohne dass wir hierfür die tatsächliche Länge kennen müssen, zum anderen entfällt dadurch die Notwendigkeit, zusätzliche Elemente im HTML-Quelltext zu verwenden. Mit `:first-line` können alle bei Inline-Elementen möglichen CSS-Eigenschaften verwendet werden.

Siehe auch

Die CSS 2.1-Spezifikation zum Pseudoelement `:first-line` unter <http://www.w3.org/TR/CSS2/selector.html#first-line>; das Rezept 1.14 zur Verwendung von Pseudoelementen.

3.21 Überschriften mit Grafiken überlagern

Problem

Sie möchten eine Überschrift mit einer Grafik überlagern, um den Text beispielsweise mit einem Farbverlauf zu versehen, wie in Abbildung 3-18 gezeigt wird.

Lösung

Wandeln Sie die Lösung aus dem vorigen Rezept so ab, dass das Pseudoelement `:after` (bzw. das eingefügte `span`-Element) die gesamte Überschrift überlagert. Definieren Sie hierfür ein Hintergrundbild, dessen Höhe exakt dem Wert für `line-height` entspricht und vertikal wie horizontal wiederholt wird. Hier der Code:

HTML:

```
<h1>CSS Kochbuch: Dritte Auflage<!--[if lte IE8]><span></span><[endif]>--></h1>
```

CSS:

```
h1 {  
    font-size: 50px;  
    line-height: 60px;
```

```

position:   relative;
}

h1 span, h1:after {
position:   absolute;
left:       0;
top:        0;
width:      100%;
height:     100%;
background: url("bilder/weisser-verlauf.png");

content:    "";
}

```



Abbildung 3-18: Eine Überschrift wird mit einer Grafik überlagert

Zunächst geben wir der Überschrift einen expliziten Wert für `line-height`. Damit die Lösung auch bei mehrzeiligen Überschriften funktioniert, muss die Grafik die gleiche Höhe haben wie der für `line-height` definierte Wert. Auch hier schaffen wir per `position: relative` den nötigen Positionierungskontext für `:after` bzw. `span`.

In der folgenden Regel wird das Pseudoelement bzw. das `span`-Element absolut in der linken oberen Ecke der Überschrift positioniert. Durch die Angabe von `height: 100%` und `width: 100%` wird dafür gesorgt, dass das Element exakt die gleichen Maße erhält wie die darunterliegende Überschrift.

In der folgenden Deklaration wird es magisch: Hier laden wir das Hintergrundbild für das Pseudoelement, das standardmäßig horizontal und vertikal wiederholt wird. Die Eigenschaft `content` wird gebraucht, damit das Pseudoelement überhaupt erzeugt wird.

Diskussion

Normalerweise werden Hintergrundbilder hinter dem betreffenden Element angezeigt. In diesem Fall brauchen wir jedoch ein Vordergrundbild, um damit die Überschrift zu überlagern. Da Vordergrundbilder in CSS jedoch nicht vorgesehen sind, wenden wir einen Trick an: Wir erzeugen nach der Überschrift ein Pseudoelement (bzw. fügen ein `span`-Element ein) und positionieren es absolut in seinem Elternelement.

Durch die ausdrücklichen Höhen- und Breitenangaben erhält es die gleichen Maße wie sein Elternelement. Und weil es später im Dokumentenfluss erscheint, wird das neue Element automatisch vor der Überschrift dargestellt.

In der Lösung wurde bereits darauf hingewiesen, dass der Wert für `line-height` der Überschrift und die Höhe des Hintergrundbilds gleich sein sollten. Auf diese Weise funktioniert die Darstellung auch bei mehrzeiligen Überschriften, weil das Hintergrundbild nicht nur von rechts nach links, sondern bei Bedarf auch horizontal wiederholt wird. Und weil Zeilen- und Bildhöhe gleich sind, kommt es auch bei mehrzeiligen Überschriften nicht zum Bruch.



Wenn Sie horizontale Rahmen oder Innenabstände für die Überschrift definieren, müssen Sie diese Werte gegebenenfalls bei der Erstellung der Grafik mit berücksichtigen.

(Die hier gezeigte Lösung hat einen Nebeneffekt: Der Text der Überschrift kann nicht mehr ausgewählt werden. Das kann gut sein – oder auch nicht. Zumindest sollten Sie darüber Bescheid wissen.)

Internet Explorer

Anders als im vorigen Rezept wird hier auch für den Internet Explorer 8 ein `span`-Element benötigt, obwohl dieser Browser eigentlich bereits mit den Pseudoelementen `:before` und `:after` umgehen kann. Das Problem ist hier, dass IE8 die Angewohnheit hat, den Hintergrund des Pseudoelements, selbst bei einem passenden Wert für `z-index`, hinter der Überschrift zu platzieren. Das Einfügen des `span`-Elements, das für IE7 sowieso gebraucht wird, schafft hier Abhilfe.

Übrigens funktioniert diese Methode nicht nur mit Überschriften, sondern auch mit Bildern und anderen Elementen.

Siehe auch

Das Grundrezept 2.8 zur Definition von Hintergrundbildern; das Grundrezept 2.19 zum Positionierungskontext; das Grundrezept 2.21 zur Eigenschaft `z-index`.

3.22 Die Gestaltung von ausgewähltem Text anpassen

Problem

Sie möchten selbst steuern, wie mit dem Cursor ausgewählter Text dargestellt wird.

Lösung

Nutzen Sie das Pseudoelement `::selection`, wie hier gezeigt:

```
p::selection, p::-moz-selection {  
  color: #ffffff;  
  background-color: #000000;  
}
```

Wie Sie sehen, verwenden wir hier zwei Selektoren, damit die Regel von möglichst vielen Browsern unterstützt wird. Während die meisten Browser das Pseudoelement direkt unterstützen, wird für Gecko-basierte Programme (Firefox usw.) teilweise noch die Präfixschreibweise `::-moz-selection` benötigt.

Diskussion

Mit dem Pseudoelement `::selection` (mit zwei vorangestellten Doppelpunkten!) können Sie eigene Stile für vom Benutzer ausgewählten Text definieren. Ursprünglich war geplant, es als Teil des *Selectormoduls* von CSS3 zu definieren. Allerdings war das Verhalten nicht präzise genug festgelegt, und die CSS-Arbeitsgruppe (CSSWG) hielt es für wichtiger, das Modul zügig fertigzustellen und die Entwicklung auf eine spätere Version von CSS zu verschieben.⁶

Trotzdem wird `::selection` von allen aktuellen Browsern (S ab S1.5, IE ab IE9, FF in der Präfixschreibweise, Opera ab 9.5) unterstützt.

Siehe auch

Den Artikel unter <http://toscho.de/2010/css-mit-selection-markierten-text-hervorheben/>; die Nachricht des Arbeitsgruppenmitglieds fantasai, in der erklärt wird, warum `::selection` aus der Spezifikation entfernt wurde, unter <http://lists.w3.org/Archives/Public/www-style/2009Oct/0277.html>; die Quirksmode-Seite zu `::selection` unter <http://www.quirksmode.org/css/selection.html>; das Rezept 1.14 zu Pseudoelementen.

⁶ Tatsächlich sind die Arbeiten an CSS4 bereits in vollem Gang.

3.23 Text mit einem Schatten versehen

Problem

Sie möchten Textteile, zum Beispiel in einer Überschrift, mit einem eigenen Schatten versehen, wie in Abbildung 3-19 gezeigt.



Abbildung 3-19: Verschiedene Möglichkeiten, die Eigenschaft `text-shadow` einzusetzen

Lösung

Verwenden Sie die Eigenschaft `text-shadow` bzw. die entsprechenden Präfixschreibweisen, um Farbe, Streuung und Abstand des Schattens zu bestimmen:

```
h1 {  
  text-shadow: 3px 3px 3px #666;  
}
```

Diskussion

Der erste Wert der Eigenschaft `text-shadow` steuert die horizontale Entfernung des Schattens vom Text, der zweite die vertikale. Die dritte Angabe besagt, wie groß die Streuung des Schattens sein soll. Je größer dieser Wert ist, desto »unschärfer« wird der

Schatten dargestellt. Der vierte Wert gibt schließlich die Farbe des Schattens an. In unserer Lösung ist das ein dunkler Grauton.

Die Transparenz des Schattens steuern

Wenn Sie die Farbe in der `rgba()`-Schreibweise angeben, können Sie die Transparenz des Schattens ebenfalls definieren. Hierdurch gehen Schatten und Hintergrund besser ineinander über:

```
h1 {  
  text-shadow: 0.05em 0.05em 0.05em rgba(10,10,10,0.5);  
}
```

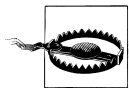
Geprägte Schrift

Durch geschicktes Kombinieren von Hintergrund-, Text- und Schattenfarben lassen sich einige interessante Effekte erzielen, zum Beispiel können Sie der Überschrift ein geprägtes Erscheinungsbild verleihen:

```
h1 {  
  background: #666;  
  color: #444;  
  text-shadow: 1px 1px 1px #888, -1px -1px 1px #000;  
}
```

Wie Sie sehen, haben wir hier einen hellen Schatten links unten und einen dunklen Schatten rechts kombiniert, um den gewünschten Effekt zu erzielen.

Tatsächlich ist in der aktuellen Version der CSS3-Spezifikation keine bestimmte Anzahl möglicher Schatteneffekte festgelegt. Im Prinzip könnten Sie also beliebig viele Schattenwürfe miteinander kombinieren, indem Sie die Werte durch Kommata getrennt voneinander angeben.



Bei diesem Effekt sollten Sie allerdings vorsichtig sein, da die Farben für Hintergrund, Schrift und Schattenwürfe ziemlich nah beieinanderliegen. Unterstützt ein Browser die Eigenschaft `text-shadow` nicht, kann die Lesbarkeit leiden.

Textschatten für den Internet Explorer

Die Eigenschaft `text-shadow` wird von allen aktuellen Browsern mit Ausnahme des Internet Explorer unterstützt. Allerdings können Sie mit der Microsoft-eigenen Eigenschaft `filter` zumindest einfache Schattenwürfe auch hier erzeugen. Dazu ein Beispiel:

```
<!--[if lte IE 9]>  
<style type="text/css">  
h1 {  
  -ms-filter: "progid:DXImageTransform.Microsoft.filter:  
shadow(color=#999999,direction=270, strength=1)";  
}
```



```

    filter: "progid:DXImageTransform.Microsoft.filter:
shadow(color=#999999,direction=270,
strength=1)";
}
</style>
<![endif]-->

```

Damit andere Browser nicht durcheinanderkommen, haben wir die Anweisung für den Internet Explorer in einen Conditional Comment (siehe Rezept 13.13) verpackt. Die Farbe (color) wird als hexadezimaler Wert angegeben, die Schattenrichtung (direction) wird als Winkel zwischen 0 und 360 Grad definiert. Der Parameter strength gibt die Größe des Schattens in Pixeln an.



Wenn Sie die Schriftenglättung ClearType von Microsoft verwenden, werden Sie feststellen, dass das Anti-Aliasing nicht funktioniert, wenn für ein Element gleichzeitig eine filter-Eigenschaft verwendet wird. Einen Link zu einer möglichen Lösung finden Sie im Abschnitt »Siehe auch«.

Noch eine Möglichkeit

Wenn es Ihnen nichts ausmacht, bei Bedarf zusätzliche Inhalte in Ihren HTML-Text einzufragen, können Sie auch die folgende Methode anwenden, bei der das Anti-Aliasing-Problem des Internet Explorer vor der Version 10 mehr oder weniger elegant umgangen wird. Hierfür fügen Sie bei Bedarf ein span-Tag in die Überschrift ein, das den Text der Überschrift ein zweites Mal enthält:

```

<h1><!--[if lte IE 9]><span>Überschrift mit Stil</span><![endif]-->Überschrift
mit Stil</h1>

```

Der CSS-Code sieht so aus:

```

<!--[if lte IE 9]>
<style type="text/css">
h1 {
    position: relative;
}

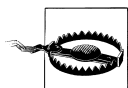
h1 span {
    position: absolute;
    left:    0;
    top:    0;

    /* IE 8+ */
    -ms-filter: progid:DXImageTransform.Microsoft.Blur(PixelRadius=3,MakeShadow=true,
ShadowOpacity=0.5);
    /* IE 7 */
    filter: progid:DXImageTransform.Microsoft.Blur(PixelRadius=3,MakeShadow=true,
ShadowOpacity=0.5);
}
</style>
<![endif]-->

```

Auch hier benutzen wir Conditional Comments, um die Anweisungen vor anderen Browsern als IE zu verstecken. Dann müssen wir den Text der Überschrift und das span-Element passgenau übereinanderlegen. Also schaffen wir per `position: relative` den nötigen Positionierungskontext für `span`.

Jetzt können wir das `span`-Element innerhalb der Überschrift absolut positionieren. Per `left: 0` und `top: 0` legen wir als Ursprung die linke obere Ecke der Überschrift fest.



Wenn Sie für die Innen- und Außenabstände der Überschrift an anderer Stelle eigene Werte definiert haben, müssen Sie diese hier möglicherweise anpassen, damit die beiden Elemente wirklich übereinanderliegen.

Es folgt der Weichzeichnungsfilter `Blur`. Der Parameter `PixelRadius` gibt die Größe des Schattens in Pixeln an. Dieser wird standardmäßig rechts unterhalb des Elements gezeichnet. Die Anweisung `MakeShadow='true'` sorgt dafür, dass der von der Überschrift verdeckte Teil des Schattens nicht dargestellt wird. Dadurch wirkt der Schattenwurf realistischer und weniger wuchtig. Der Parameter `ShadowOpacity` legt schließlich fest, wie durchscheinend der Schatten sein soll. Die möglichen Werte liegen zwischen 1 (nicht durchscheinend) und 0 (komplett durchsichtig).

Ab Internet Explorer 8 verwendet der Browser für nicht standardmäßige Eigenschaften die Präfixschreibweise. Aus diesem Grund sehen Sie die gleiche Anweisung gleich zweimal: einmal als `-ms-filter` für IE8 und 9 (wir hoffen sehr, dass IE ab der Version 10 die Eigenschaft `text-shadow` direkt unterstützt!) und einmal als `filter` für ältere Versionen.

Siehe auch

Die CSS 2.1-Spezifikation zur Eigenschaft `text-shadow` unter <http://www.w3.org/TR/REC-CSS2/text.html#text-shadow-props>; weitere Informationen zur Präfix-Schreibweise in IE unter <http://blogs.msdn.com/b/ie/archive/2008/09/08/microsoft-css-vendor-extensions.aspx>; einen Lösungsansatz des Webentwicklers Zoltan »Du Lac« Hawryluk für das ClearType-Problem im Internet Explorer unter <http://tinyurl.com/cleartype-filter>.

3.24 Hoch- und tiefgestellten Text ohne Grundlinienversatz darstellen

Problem

Sie möchten bestimmte Textteile hoch- oder tiefgestellt darstellen, ohne dass die Zeilen unterschiedlich hoch sind, wie in Abbildung 3-20 gezeigt.

Lösung

Markieren Sie die gewünschten Textteile mit den HTML-Elementen `sub` und `sup`. Danach können sie per Stylesheet-Regel vertikal auf der Grundlinie ausgerichtet werden.

Mithilfe relativer Positionierung können Sie den hoch- und tiefgestellten Elementen eine passende Platzierung zuweisen. Ein Beispiel:

```
sup {
  vertical-align: baseline;
  position: relative;
  top: -0.4em;
}

sub {
  vertical-align: baseline;
  position: relative;
  top: 0.4em;
}

p {
  line-spacing: 1.3em;
}
```

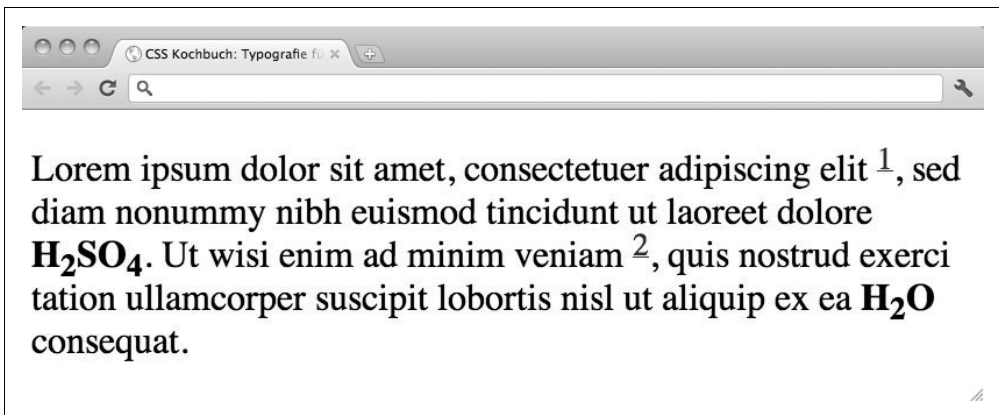


Abbildung 3-20: Trotz hoch- und tiefgestellten Texts bleiben regelmäßige Zeilenhöhen erhalten

Der HTML-Code sieht so aus:

```
<p>Lorem ipsum dolor sit amet, consectetur adipiscing elit <sup><a href="#fussnote1">1</a></sup>, sed diam nonummy nibh euismod tincidunt ut laoreet dolore H<sub>2</sub>S<sub>O<sub>4</sub></sub>. Ut wisi enim ad minim veniam <sup><a href="#fussnote">2</a></sup>, quis nostrud exerci tation ullamcorper suscipit lobortis nisl ut aliquip ex ea H<sub>2</sub>O consequat.</p>
```

Im ersten Schritt verschieben wir die Elemente zurück auf die Grundlinie, d.h., sie werden auf gleicher Höhe angezeigt wie der übrige Text. Durch die relative Positionierung können wir die Elemente anschließend basierend auf ihrer ursprünglichen Platzierung im Text nach oben oder unten verschieben.

In der letzten Regel wird schließlich dafür gesorgt, dass auf den Zeilen genügend Leer- raum nach oben und unten bleibt, damit sich die Zeichen nicht überschneiden.

Diskussion

In wissenschaftlichen Abhandlungen oder Gesetzestexten werden häufig Sub- oder Superskripten verwendet, um Formeln, Querverweise und Fußnoten zu kennzeichnen.

In der Standarddarstellung des Browsers wird für die Anzeige von höher- und tiefergestelltem Text die Eigenschaft `vertical-align` mit den Werten `super` bzw. `sub` verwendet. Um eine feinere Kontrolle zu erlangen können auch eigene Werte (Pixel, em-Einheiten, Prozentwerte usw.) angegeben werden. Hierbei wird die Zeilenhöhe automatisch angepasst, wenn der Platz nicht ausreicht. Das ist gut gemeint, entspricht aber nicht immer dem Gewünschten – beispielsweise wenn Sie einen einheitlichen Grundlinienrhythmus verwenden wollen (siehe Rezept 12.13).

Bei diesem Rezept wird der Zeilenabstand nicht automatisch vom Browser angepasst, da wir statt `vertical-align` die relative Positionierung wählen (siehe Grundrezept 2.18). Die Verschiebung nach oben bzw. unten wird mit der Eigenschaft `top` definiert.

Damit trotzdem genügend vertikaler Platz zwischen den Zeilen vorhanden ist, ohne dass sich die Zeichen überschneiden, setzen wir die Eigenschaft `line-height` für Absätze außerdem auf einen Wert von `1.3em`. Dies ist nur ein grober Richtwert. Wenn Ihre Formeln, Texte usw. komplexer sind, müssen Sie den Wert möglicherweise stärker erhöhen.

Siehe auch

Den Artikel von Paul Armstrong über die hier verwendete Methode unter <http://paularmstrongdesigns.com/weblog/stop-superscripts-from-breaking-line-heights-once-and-for-all/>; den Artikel von Jukka Korpela zur Anpassung des Zeilenabstands unter <http://www.cs.tut.fi/~jkorpela/www/linespacing.html>; die Website des XML Maiden-Projekts zur Verwendung von CSS für die Gestaltung mathematischer Formeln unter <http://xml-maiden.com/>.

3.25 Die Schreibrichtung festlegen

Problem

Sie wollen in mehrsprachigen Dokumenten steuern, ob der Text von links nach rechts oder andersherum fließt.

Lösung

Kennzeichnen Sie die verschiedensprachigen Inhalte im HTML-Dokument mit den passenden Attributen:

```
<div lang="de">...</div>
<div lang="ar">...</div>
<div lang="he">...</div>
```

Wählen Sie dann die Dokumentteile mit einem passenden Attributselektor aus, dem Sie in diesem Fall keinen weiteren Elementselektor voranzustellen brauchen, und legen Sie mit der Eigenschaft `direction` fest, welche Schreibrichtung verwendet werden soll:

```
[lang="de"] { direction: ltr; }  
[lang="ar"] { direction: rtl; }  
[lang="he"] { direction: rtl; }
```

Alternativ dazu können Sie auch die spezielle Pseudoklasse für Sprachelemente verwenden. Diese wird im Internet Explorer allerdings erst ab Version 8 unterstützt:

```
*:lang(de) { direction: ltr; }  
*:lang(ar) { direction: rtl; }  
*:lang(he) { direction: rtl; }
```

Diskussion

Standardmäßig wird Text in den meisten westlichen Ländern von links nach rechts geschrieben und gelesen. Gelegentlich kann es aber passieren, dass Sie Inhalte im gleichen Dokument vorhalten wollen, die beispielsweise in Hebräisch oder Arabisch verfasst sind. In diesen Sprachen wird die umgekehrte Schreibrichtung verwendet.

Standardmäßig verwenden Browser die Schreibrichtung, die zur jeweiligen Spracheinstellung passt. Mit der CSS-Eigenschaft `direction` können Sie diese Einstellung für bestimmte Dokumentteile (oder auch das gesamte Dokument) selbst steuern.



Wird die Schreibrichtung für die gesamte Website geändert, ist es unter Umständen sinnvoll, auch das Layout zu spiegeln. Um den Lesegewohnheiten der Leser Rechnung zu tragen, sollte ein links angeordnetes Navigationsmenü dann beispielsweise auf der rechten Seite stehen.

Siehe auch

Die CSS-Spezifikation zur Eigenschaft `direction` unter <http://www.w3.org/TR/CSS2/visuren.html#direction>; einen deutschsprachigen Artikel dazu auf der Website von The Styleworks unter <http://www.thestyleworks.de/ref/direction.shtml>.

3.26 Pull Quotes gestalten

Problem

Sie wollen ansprechende *Pull Quotes* (auch »Lockzitat« oder »Seitenansprache« genannt) auf Ihrer Website nutzen. Das Zitat soll typografisch hervorgehoben werden, indem die Schrift deutlich vergrößert, andersfarbig und rechtsbündig angezeigt wird, wie in Abbildung 3-21 gezeigt.

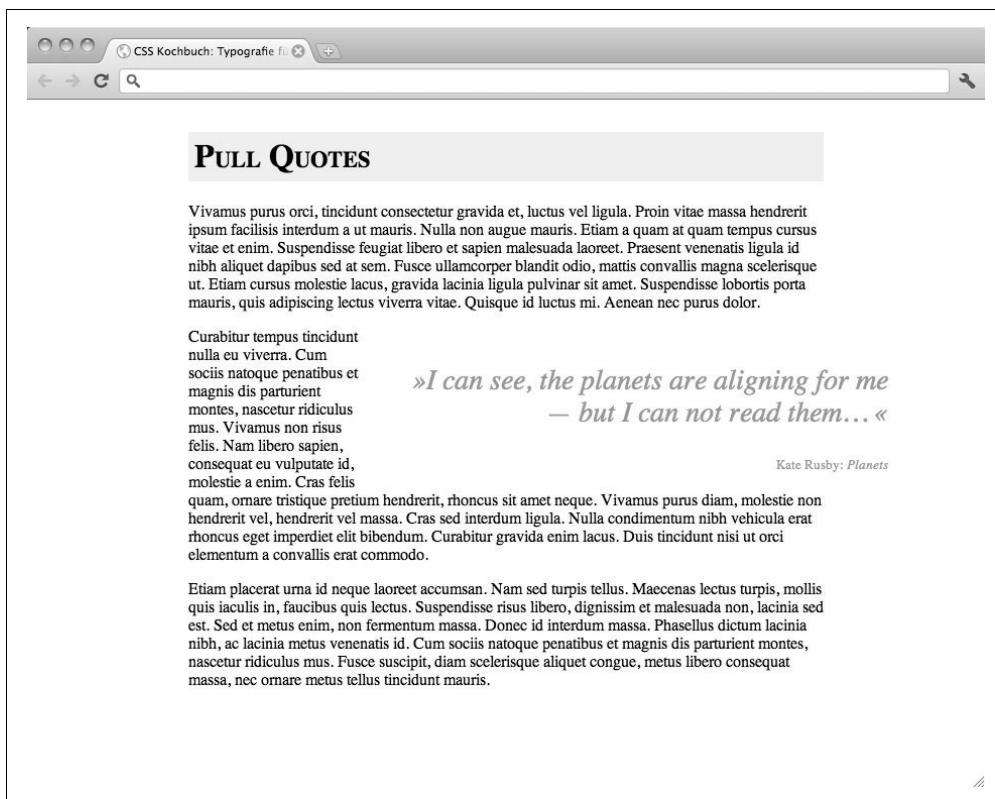


Abbildung 3-21: Ein Pull Quote

Außerdem soll das Zitat rechts ein Stück aus dem Textfluss ausgerückt werden, während es auf der linken Seite vom übrigen Text umflossen wird. Zudem sollen vor und nach dem Zitattext typografisch korrekte »französische« Anführungszeichen eingefügt werden. Die Quellenangabe soll etwas kleiner als der normale Absatztext dargestellt werden.

Lösung

Markieren Sie zunächst den Zitattext und die Autorenangabe mit passenden HTML-Elementen. Danach umgeben Sie beides mit einem `div`-Element, das ein aussagekräftiges `id`-Attribut erhält:

```
<blockquote>
<p>I can see, the planets are aligning for me<br /> – but I can not read them...</p>
<p>Kate Rusby: <cite>Planets</cite></p>
</blockquote>
```

```
<p>Curabitur tempus tincidunt nulla eu viverra...</p>
```

Und hier das Stylesheet:

```
blockquote {
```

```

float: right;
padding: 0.5em 1em;
margin: 0 -5em 0.5em 2em;

color: #999;
text-align: right;
}

blockquote p:first-child {
font-size: 28px;
font-style: italic;
}

blockquote p + p {
font-size: smaller;
margin-bottom: 0;
}

blockquote p:before { content: "\00BB "; } /* » */
blockquote p:after { content: " \00AB"; } /* « */

blockquote p + p:before { content: ""; }
blockquote p + p:after { content: ""; }

```

Die erste Regel sorgt per `float: right` dafür, dass Zitat und Quellenangabe links vom übrigen Text umflossen werden können. Es folgt die Definition von Innen- und Außenabständen. Wichtig ist hier vor allem der zweite Wert für `margin`. Durch den negativen Wert `-5em` wird das Zitat um 5 em-Einheiten nach rechts verschoben. Danach kommt die Definition der Textfarbe und die Textausrichtung für die enthaltenen Elemente.

Im nächsten Schritt legen wir fest, dass die Schriftgröße für das eigentliche Zitat 28 Pixel betragen und der Text in kursiver Schrift dargestellt werden soll.



Zwar werden in vielen Browsern vor und nach dem `q`-Element automatisch Anführungszeichen eingefügt. Hier sollen aber auf jeden Fall die französischen doppelten spitzen Klammern verwendet werden. Daher greifen wir, wie bereits in Rezept 3.17 beschrieben, auf die Pseudoelemente `:before` und `:after` zurück. Mit der Eigenschaft `content` geben wir jeweils das entsprechende Unicode-Zeichen an.

Die folgende Regel widmet sich der Quellenangabe. Sie soll kleiner als der normale Absatztext angezeigt werden. Mit dem Selektor für benachbarte Geschwisterelemente wählen wir den zweiten Absatz aus und definieren anhand der Eigenschaft `font-size` den Wert `smaller`. Die Deklaration `margin-bottom: 0` sorgt schließlich dafür, dass unter der Pull Quote nicht zu viel Leerraum entsteht.

Für das `cite`-Element haben wir hier auf eigene Stildefinitionen verzichtet. Die standardmäßig kursive Darstellung reicht aus.

Danach definieren wir, wie in Rezept 3.17 beschrieben, die gewünschten Anführungszeichen. Damit nicht auch der zweite Absatz mit Anführungszeichen versehen wird, müssen wir den Wert der Eigenschaft `content` hier ausdrücklich zurücksetzen.

Diskussion

Pull Quotes werden benutzt, um die Aufmerksamkeit des Lesers zu erregen, damit er Ihren Text auch wirklich beachtet. Eine einfache Methode, Pull Quotes zu erstellen, besteht darin, die typografischen Eigenschaften wie Schriftgröße, -farbe und -familie zu ändern.

Wir haben diese Lösung bewusst einfach gehalten, um Ihnen die Grundlagen zur Gestaltung von Pull Quotes zu vermitteln. Prinzipiell sind Ihrer Fantasie aber kaum Grenzen gesetzt. Durch geschicktes Kombinieren der in diesem Kapitel gezeigten Rezepte lässt sich eine Vielzahl von Variationen erzielen.

Wie wäre es beispielsweise mit grafischen Anführungszeichen? Kein Problem. Verwenden Sie einfach eine der Techniken zur Bildersetzung (Rezept 4.6), um mithilfe der Pseudoelemente `:before` und `:after` Bilder einzufügen, ohne hierfür den Quelltext ändern zu müssen.

Vielleicht wollen Sie das Zitat aber auch in einer Schrift darstellen, die nicht auf dem Benutzerrechner installiert ist. Das wäre eine Aufgabe für die in Rezept 3.11 vorgestellten Webfonts.

Grundsätzlich stehen alle Möglichkeiten der Textgestaltung mit CSS offen. Der Werkzeugkasten ist gut gefüllt. Jetzt brauchen Sie nur noch einen guten Text für Ihr Zitat.

Siehe auch

Die übrigen Rezepte zur Gestaltung von Text in diesem Kapitel; das Rezept 3.17 zur Verwendung typografisch korrekter Anführungszeichen in diesem Kapitel; das Grundrezept 2.23 zur Verwendung von Floats; die HTML5-Spezifikation zum `cite`-Tag unter <http://www.whatwg.org/specs/web-apps/current-work/multipage/text-level-antics.html#the-cite-element>; die Rezepte 1.9 und 1.14 zu Selektoren für benachbarte Geschwisterelemente bzw. zu den Pseudoelementen `:before` und `:after`.

Einführung

Als Marc Andreessen, der Gründer von Netscape, Anfang der 1990er-Jahre dafür sorgte, dass der Browser auch Bilder darstellen konnte, setzte er damit nicht nur eine weltweite Diskussion in Gang, sondern ergänzte das Web gleichzeitig um eine weitere visuelle Komponente. Plötzlich waren Webdokumente nicht mehr dazu verdammt, als textüberladene akademische Schriften ihr Dasein zu fristen. Der Grundstein für ein tatsächliches Web-»Design« war gelegt.

Seit diesen Anfangstagen haben Designer Grafiken eingesetzt, um Websites über die Platzierung einzelner Bilder hinaus grafisch zu erweitern. Da die Entwicklung der Webbrowser nicht stehen bleibt, ist auch die Arbeit mit Bildern deutlich einfacher geworden.

In diesem Kapitel finden Sie eine Reihe von Rezepten, die Ihnen bei der Arbeit mit CSS für die Darstellung von Bildern helfen können. Sie finden hier Informationen zur Arbeit mit Rahmen, zu Änderungen an Hintergrundbildern, zur Ersetzung von HTML-Text durch Bilder, zur Verwendung von Sprites und vieles mehr.

4.1 Bilder mit Rahmen versehen

Problem

Sie wollen einen eigenen Rahmen für ein Bild verwenden.

Lösung

Legen Sie anhand der Kurzschrifteigenschaft `border` eigene Stile für den Rahmen fest, wie in Abbildung 4-1 gezeigt.

```
img {  
  border: 6px double #666;  
  padding: 6px;  
}
```

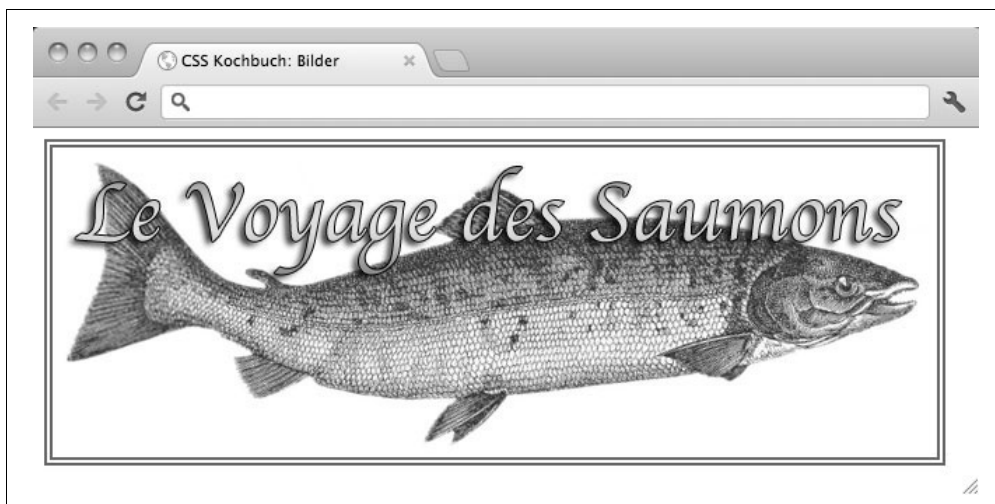


Abbildung 4-1: Ein Bild wird mit einem doppelten Rahmen versehen

Diskussion

Wir verwenden in dieser Lösung die Kurzschrifteigenschaft `border`, um für alle vier Rahmenseiten die gleichen Werte in einer Deklaration anzugeben. Der erste der drei Werte besagt, dass der Rahmen eine Breite von 6 Pixeln erhalten soll, der zweite Wert (`double`) legt fest, dass der Rahmen als doppelte durchgezogene Linie um das Bild herumlaufen soll, während der dritte Wert (`#666`) den Rahmen dunkelgrau einfärbt.

Die Kurzschrifteigenschaft `padding` schafft außerdem ein wenig Platz zwischen dem Bild und dem Rahmen.

Laut CSS-Spezifikation sind Bilder ersetzte Inline-Elemente. Es wird also nicht der Inhalt eines Tags (wie beispielsweise `<p>Ein Absatz</p>`) dargestellt. Stattdessen wird das Tag in der Darstellung durch die im `src`-Attribut angegebene Datei ersetzt. Gleichzeitig sind Bilder Inline-Elemente. Das heißt, Sie können, wie im Boxmodell (siehe Rezept 1.16) beschrieben, Rahmen und Innenabstände definieren. Da sich die Innenabstände und Rahmen für die verschiedenen Seiten nicht unterscheiden, verwenden wir hier die Kurzschrifteigenschaften `border` und `padding`.

Siehe auch

Das Rezept 4.2 zum Entfernen der Rahmen von Bildern; die Grundrezepte 2.6 und 2.7 zur Definition von Innenabständen bzw. zum Erstellen von Rahmen; das folgende Rezept zum Entfernen von Rahmen.

4.2 Rahmen um Bilder nicht darstellen

Problem

Eine allgemeine CSS-Regel definiert für alle Bilder der Website einen Rahmen, z.B.:

```
img {  
    border: 1px solid black;  
}
```

Dieser soll für bestimmte Bilder entfernt werden.

Lösung

Verwenden Sie eine Regel mit höherer Spezifität und setzen Sie den Rahmenstil auf den Wert `none`. Am einfachsten geht das, wenn Sie die betreffenden Bilder mit einem `class`-Attribut versehen, das Sie dann im Stylesheet auswählen können.

```
img.rahmenlos {  
    border-style: none;  
}
```

Hier die Verwendung des `class`-Attributs im Zusammenhang:

```
<img class='rahmenlos' src='bilder/beispiel.gif' alt='Beispiel' />
```

Für den Fall, dass Sie nicht auf den HTML-Code zugreifen können, etwa weil dieser automatisch erzeugt wird, können Sie eine höhere Spezifität auch durch die Kombination von CSS-Selektoren erreichen. Vielleicht befinden sich die Bilder immer innerhalb eines Absatzes, der von einem `div`-Element umgeben ist, wie hier:

```
<div>  
<p>Etwas Text.</p>  
<p>Etwas Text mit einem Bild. <img src='bilder/beispiel.gif' alt='Beispiel' /></p>  
<p>Noch mehr Text</p>  
<img src='bilder/beispiel.gif' alt='Beispiel' />  
</div>
```

Danach verwenden wir einen einfachen Nachfahrenselektor, um das Bild innerhalb des Absatzes auszuwählen und mit einer eigenen Regel zu versehen:

```
p img {  
    border-style: none;  
}
```

Hierdurch wird lediglich das Bild im zweiten Absatz ohne Rahmen dargestellt. Die Regel gilt nur für Bilder, die ein Kindelement eines Absatzes sind. Dieser muss seinerseits ein Kindelement eines `div`-Elements sein.

Diskussion

In dieser Lösung spielen zwei Dinge zusammen: einerseits die Spezifität der Selektoren und andererseits das Verhalten der Eigenschaft `border` (bzw. der entsprechenden Einzeleigenschaften).

Anhand der Spezifität wird entschieden, welche Regel vor einer anderen Vorrang erhält, wenn sie versuchen, das gleiche Element zu gestalten. Je »spezieller« eine Regel ist, desto größer ist der Vorrang, den sie vor anderen Regeln erhält.

Laut der CSS-Spezifikationen haben Regeln, die einen Klassenselektor verwenden (`img.raahmenlos`), Vorrang vor reinen Elementselektoren. Noch mehr Gewicht erhalten Regeln mit einem oder mehreren Kindselektoren (`div > p > img`). Mithilfe der Spezifität lassen sich allgemeine Stilvorgaben bei Bedarf leicht überschreiben.

Für den zweiten Teil der Lösung müssen Sie wissen, dass immer ein Rahmenstil definiert sein muss, damit der Rahmen auch dargestellt wird. Angaben zu Rahmenbreite und -farbe werden ansonsten schlicht ignoriert, so als gäbe es den Rahmen gar nicht. Daher reicht die Deklaration `border-style: none` vollkommen aus, um auch komplexe Rahmendefinitionen zu unterdrücken.

Kürzer geht es nur noch per Kurzschrifteigenschaft:

```
border: 0;
```

Hierbei wird jedoch nicht der Wert von `border-style` verändert, sondern von `border-width`. Das Resultat ist das Gleiche: Es wird kein Rahmen mehr angezeigt.

Siehe auch

Das Rezept 1.20 zur Spezifität von CSS-Regeln; das Grundrezept 2.7 zur Erstellung von Rahmen von Elementen; die CSS 2.1-Spezifikation zum Thema Spezifität unter <http://www.w3.org/TR/CSS2/cascade.html#specificity>; die CSS 2.1-Spezifikation zur Definition von Rahmen unter <http://www.w3.org/TR/CSS2/box.html>.

4.3 Die Rahmenfarbe ändern, wenn sich der Mauszeiger über dem Bild befindet

Problem

Ein Bild soll als Link verwendet werden. Damit dies leichter erkennbar ist, soll sich die Rahmenfarbe ändern, wenn sich der Mauszeiger über dem Bild befindet, wie in Abbildung 4-2 gezeigt.

Lösung

Verwenden Sie die Pseudoklasse `:hover` für den Link und geben Sie für den Rahmen des Bilds die gewünschte Farbe an:

```
a img {  
    border: 3px solid black;  
}  
  
a:hover img {  
    border: red;  
}
```

Und hier der dazu gehörige HTML-Code:

```
<a href="http://www.csskochbuch.com">  
</a>
```

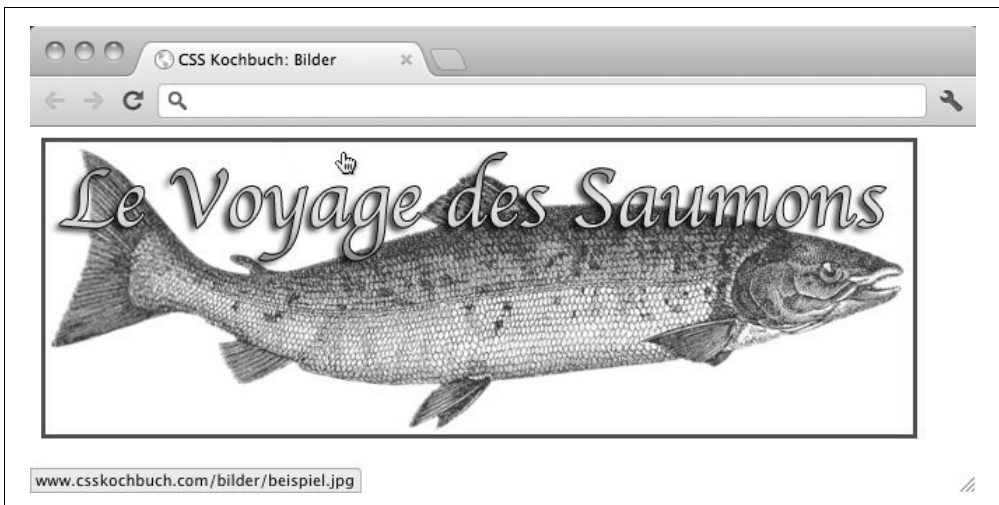


Abbildung 4-2: Die Rahmenfarbe wird geändert, wenn sich der Mauszeiger über dem Bild befindet

Die erste Regel umgibt das Bild mit einem 3 Pixel breiten schwarzen Rahmen. In der zweiten Regel wird festgelegt, dass der Rahmen rot dargestellt werden soll, sobald sich der Mauszeiger über dem Bild befindet. (Da wir die Eigenschaften des Rahmens bereits für das Bild definiert haben, reicht es aus, in der `:hover`-Regel nur die Änderungen anzugeben.)

Diskussion

Gelegentlich sollen Bilder als Links verwendet werden – beispielsweise in Navigationsmenüs. Eine einfache Möglichkeit, dies dem Nutzer zu signalisieren, besteht in der Änderung der Rahmenfarbe, wenn sich der Mauszeiger über dem Bild befindet.

Da diese Regel nur für Bilder gelten soll, die als Link dienen, haben wir den Nachfahrenselektor `a img` gewählt. Die zweite Regel gilt also genau genommen für Links, in denen sich ein Bild befindet, und nicht für das Bild selbst.

Die zweite Regel lässt sich folgendermaßen übersetzen: »Wenn sich der Mauszeiger über einem Bild befindet, das als Link definiert wurde (von einem `a`-Element umgeben ist), soll das Bild von einem roten Rahmen mit 1 Pixel Breite umgeben werden.«

Gelegentlich ist dieser Effekt auch erwünscht, wenn die Bilder nicht als Links dienen. In diesem Fall definieren Sie den Rollover-Effekt direkt für das Bild, wie hier gezeigt:

```
img {  
    border: 3px solid black;  
}  
  
img:hover {  
    border: red;  
}
```

Siehe auch

Das Rezept 1.14 zu den Pseudoklassen `:hover` und `:active`; das Grundrezept 2.7 zur Definition von Rahmen.

4.4 Skalierbare Bilder

Problem

Bilder sollen skaliert werden, wenn sich die Größe des Browserfensters ändert.

Lösung

Geben Sie die Breite der Bilder in Prozent an (siehe Abbildung 4-3).

```
img {  
    width: 100%;  
}
```

Diese Regel besagt, dass das Bild so breit sein soll wie sein umgebender Block (z.B. ein `div`-Element für den Hauptinhalt einer Seite oder der Ansichtsbereich).



Damit diese Lösung funktioniert, muss das Bild entweder ein Kindelement des `body`-Elements sein, oder die Dimensionen der umgebenden Elemente müssen ebenfalls in Prozentwerten angegeben werden.

Diskussion

Bei der Erstellung »flüssiger« oder flexibler Layouts werden Breite und Höhe von Textblöcken durch die Verwendung von Prozentwerten dynamisch an die Dimensionen des

Browserfensters angepasst. Bilder, deren Höhe und Breite in Pixeln angegeben wurde, behalten dagegen ihre Größe.

Um sicherzustellen, dass alle Seitenelemente in flexiblen Layouts im richtigen Verhältnis zueinander dargestellt werden, können Entwickler Höhe und Breite von Bildern ebenfalls in Prozentwerten angeben.

Moderne Browser skalieren die Höhe von Bildern im Verhältnis zu ihrer Breite; eine zusätzliche Angabe der Höhe ist also nicht nötig. Das Seitenverhältnis des Bilds bleibt erhalten.

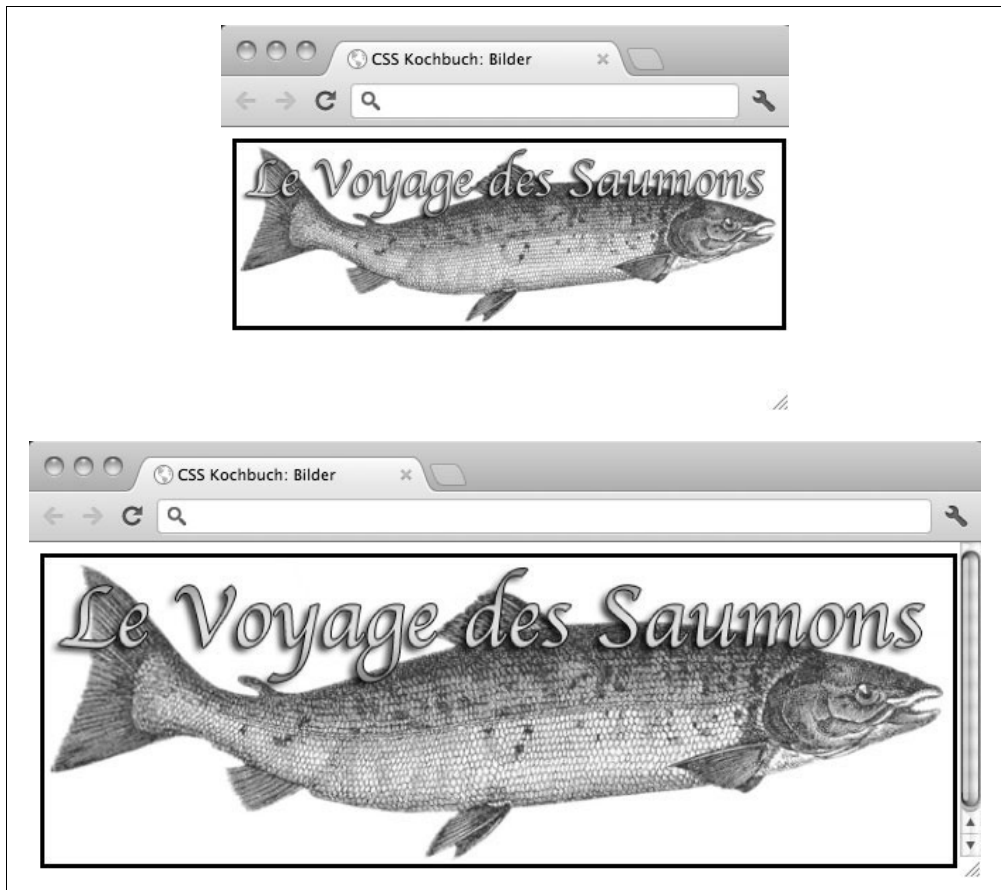


Abbildung 4-3: Das skalierte Bild



In dieser Lösung wird das Bild relativ zur Größe des Browserfensters skaliert. Soll sich die Größe dagegen an der Schriftgröße orientieren, verwenden Sie anstelle von Prozentwerten eine relative Maßeinheit, zum Beispiel `em`.

Die Skalierung beschränken

Bei diesem Vorgehen kann es passieren, dass Bilder durch die Skalierung nicht mehr korrekt dargestellt werden. So können in JPEG-Bildern beispielsweise Artefakte erkennbar sein oder die Pixel in einer GIF-Grafik.

In diesem Fall kann es sinnvoll sein, die Skalierung von Bildern zu beschränken. Dafür verwenden Sie die Eigenschaft `max-width`, wie hier gezeigt:

```
img {  
  width:      60%;  
  max-width: 100%;  
}
```

Auf diese Weise können Sie die Bilder weiterhin skalieren. Gleichzeitig ist sichergestellt, dass sie auf jeden Fall noch in das umgebende Element passen (z.B. eine Seitenleiste).

Der Schlüssel für diese Methode besteht darin, dass sich Prozentwerte nicht auf das Element selbst beziehen, sondern auf die Breite des umgebenden Elements.

Siehe auch

Das Grundrezept 2.1 zur Verwendung von Längeneinheiten in CSS; das Grundrezept 2.2 zur Angabe von Breiten; die CSS 2.1-Spezifikation zur Eigenschaft `max-width` unter <http://www.w3.org/TR/CSS21/visudet.html#min-max-heights>; das Rezept 10.1 zur Verwendung von Media Queries; einen Artikel zum Thema unter <http://www.alistapart.com/articles/responsive-images-how-they-almost-worked-and-what-we-need/>.

4.5 Bilder mit einer Bildunterschrift versehen

Problem

Sie wollen Bilder mit einer Beschriftung versehen. Die Breite soll sich dynamisch an die Bildgröße anpassen.

Lösung

Umgeben Sie die Einzelteile mit einem `div`-Element, das Sie mit einem passenden `class`-Attribut versehen. Darin befindet sich das Bild (`img`) sowie ein Absatz mit Text, der für die Bildunterschrift verwendet werden soll:

```
<div class="abbildung">  
    
  <p>Das CSS-Kochbuch</p>  
</div>
```


Die einzelnen Bestandteile können Sie nun per Klassen- bzw. Nachfahrenselektoren auswählen und mit eigenen Stilen versehen:

```
.abbildung {  
    float: left;  
    border: 1px solid gray;  
    margin: 0 1em 1em 0;  
}  
  
.abbildung p {  
    padding: 3px 5px;  
    margin: 0;  
  
    border-top: 1px solid gray;  
  
    font-style: italic;  
    font-size: smaller;  
}
```

Zu Beginn der Lösung wird festgelegt, dass Elemente mit dem class-Attribut `abbildung` nach links verschoben und rechts vom übrigen Inhalt umflossen werden sollen. Außerdem erhalten sie einen 1 Pixel breiten grauen Rahmen.

Damit der umgebende Text nicht direkt am Behälter »klebt«, schaffen wir per `margin` rechts und unterhalb des Behälters ein wenig Platz.

Es folgen die Angaben für die Bildbeschriftung. Um sie visuell besser vom Bild zu trennen, verwenden wir die Eigenschaft `border-top`, um oberhalb eine 1 Pixel breite Linie zu erstellen.

Zusätzlich gibt es noch etwas Kosmetik: Die Schrift soll kursiv und kleiner als der Standardwert dargestellt werden. Außerdem schaffen wir mit der Eigenschaft `padding` etwas Platz um die Beschriftung. Wir benutzen hier die Zwei-Argumente-Schreibweise. Der erste Wert steht für den oberen und unteren Innenabstand, der zweite gibt die seitlichen Innenabstände an.

Das Ergebnis sehen Sie in Abbildung 4-4.

Diskussion

Oft sollen Bilder mit einer Beschriftung versehen werden, um ihren Inhalt zu erläutern. Damit sie sich wie »normale« Bilder verhalten, verpacken wir Bild (`img`) und Beschriftung (`p`) in ein `div`-Element.

Durch diese Methode ist es beispielsweise möglich, Bild und Beschriftung gemeinsam als Float zu definieren und vom übrigen Text umfließen zu lassen, wie in der Lösung gezeigt.



Abbildung 4-4: Ein Bild mit Beschriftung

Das HTML5-Element figure

In naher Zukunft werden Sie für die Markierung von Abbildungen auf den Einsatz von `div`-Elementen verzichten können. Stattdessen kommt das nach HTML5 semantisch geeignetere `figure`-Element zum Einsatz. Ergänzt wird es durch das Element `figcaption`, mit dem bei Bedarf eine Bildbeschriftung definiert werden kann. Der HTML-Code sähe dann so aus:

```
<figure>
  
  <figcaption>Das CSS-Kochbuch</figcaption>
</figure>
```

Der CSS-Code entspricht weitgehend dem der Lösung. Allerdings kann hier auf die Verwendung von Klassen und Nachfahrenselektoren verzichtet werden, weil die Elemente direkt angesprochen werden:

```
figure {
  float: left;
  border: 1px solid gray;
  margin: 0 1em 1em 0;
}
```

```
figcaption {
    padding: 3px 5px;

    border-top: 1px solid gray;

    font-style: italic;
    font-size: smaller;
}
```

Die Beschriftung vor dem Bild anzeigen

Wollen Sie dieses Beispiel noch etwas weiter treiben, können Sie das Bild auch mit dem Beschriftungstext überlagern, wie in Abbildung 4-5 gezeigt wird.



Abbildung 4-5: Ein Bild wird mit Text überlagert

Der HTML-Code hat dabei die gleiche Struktur wie zuvor:

```
<div class="abbildung">
  
  <>Die neue CD ist da!</p>
</div>
```

Das Stylesheet enthält folgende Regeln:

```
.abbildung {
  position: relative;
  border: 1px solid #000;
  display: inline-block;
}

.abbildung img {
  display: block;
}

.abbildung p {
  position: absolute;
  bottom: 1em;
  width: 100%;

  border: solid #000;
  border-width: 1px 0;

  font-size: 4em;
  text-align: center;
  color: rgb(90%,90%,90%);

  background: gray;
  background: url("bilder/halbtransparent.png");
  background: rgba(0,0,0,0.6);
}
```

In der ersten Regel definieren wir das umgebende `div`-Element (`abbildung`) als Positionierungskontext für die Beschriftung (siehe Rezept 2.19) und versehen es mit einem 1 Pixel breiten schwarzen Rahmen. Zudem erhält es die Darstellungsrolle `inline-block`. Dadurch passt sich die Größe des `div`-Elements automatisch an die Dimensionen des Bilds an.

Die zweite Regel legt fest, dass das Bild als Blockelement dargestellt wird. Standardmäßig ist es als Inline-Element definiert. Es wird also behandelt wie Text, und es erhält eine typografische Grundlinie. Das führt dazu, dass unterhalb des Bilds etwas Platz für Buchstaben wie `p`, `g` oder `y` gelassen wird. Durch die Deklaration `display: block` lässt sich das Problem aber leicht beheben.

Die dritte Regel enthält die wichtigsten Anweisungen: Die Beschriftung wird innerhalb des `div`-Elements absolut positioniert, und die Angabe von `width: 100%` sorgt dafür, dass die Beschriftung die gesamte Breite des Abbildungscontainers einnimmt.

Nun kommt die Regel für die Beschriftung. Da wir den Abbildungscontainer als Positionierungskontext bestimmt haben, können wir die Beschriftung darin nun frei positionieren. Die Eigenschaft `bottom` gibt hierbei den Abstand zur Unterkante des Containers an.

Danach folgt die Definition der oberen und unteren Rahmenkante. Zuerst legen wir mit der Kurzschrifteigenschaft `border` Rahmenstil und -farbe fest. In der nächsten Zeile geben wir der Ober- und Unterkante des Rahmens per `border-width` eine Breite von 1 Pixel, während durch den zweiten Wert (0) links und rechts keine Rahmenkanten dargestellt werden.

Nach der Definition der typografischen Eigenschaften (Schriftgröße, -ausrichtung und -farbe) kommen wir schließlich zum transparenten Hintergrund. Wir benutzen hier die in Grundrezept 2.15 vorgestellte Schreibweise, die aus drei aufeinanderfolgenden `background`-Deklarationen besteht. Browser, die bereits mit der `rgba()`-Notation umgehen können, brauchen die halbtransparente PNG-Grafik nicht zu laden. Ältere Browser verwenden dagegen das Hintergrundbild aus der zweiten Deklaration und ignorieren die dritte Zeile. Kann das Bild nicht geladen werden, greift die erste Deklaration, und es wird ein schlichter grauer Hintergrund angezeigt.

Internet Explorer 7 und inline-block

Da der Internet Explorer 7 eine recht eigene Vorstellung von der Darstellungsrolle `inline-block` hat, müssen Sie hier bei Bedarf ein wenig nachbessern. Dafür muss für das umgebende `div`-Element `hasLayout` aktiviert sein. Durch die Deklaration `display: inline` (ja, richtig gelesen!) wird das Element auch im IE7 wie gewünscht dargestellt. Das geht beispielsweise so:

```
<!--[if lte IE 7]>
<style type="text/css">
.abbildung {
    zoom: 1;
    display: inline;
}
</style>
<![endif]>
```

Falls Sie das Thema `hasLayout` näher interessiert, finden Sie in Rezept 13.15 weitere Details und eine Menge weiterführender Links.

Siehe auch

Das Rezept 1.4 zur Gruppierung von Selektoren; die HTML5-Spezifikation zum `figure`-Element unter <http://www.w3.org/TR/html5/grouping-content.html#the-figure-element>; den Artikel von Richard Clark zu den Elementen `figure` und `figcaption` unter <http://html5doctor.com/the-figure-figcaption-elements/>; das Rezept 2.8 zur Verwendung von Hintergrundbildern; das Rezept 1.17 zu Darstellungsrollen; zwei Artikel zum Thema `inline-block` unter <http://blog.mozilla.com/webdev/2009/02/20/cross-browser-inline-block/> und <http://robertnyman.com/2010/02/24/css-display-inline-block-why-it-rocks-and-why-it-sucks/>.

4.6 Text durch Grafiken ersetzen

Problem

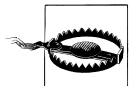
HTML-Text, in unserem Beispiel eine Überschrift, soll durch eine grafische Darstellung ersetzt werden, beispielsweise wenn die typografischen Möglichkeiten von CSS nicht ausreichen oder die gewünschte Schrift ansonsten lizenziert werden muss.

Lösung 1: Phark (Off-Left-Technik)

Verwenden Sie die Phark-Methode zur Textersetzung (gezeigt in Abbildung 4-6). Hierbei wird der Text mittels der CSS-Eigenschaft `text-indent` aus dem sichtbaren Bereich des Browserfensters herausgeschoben. An seiner Stelle wird ein für das Element definiertes Hintergrundbild angezeigt (hier anhand eines `h1`-Elements demonstriert):

```
h1 {  
  text-indent:      -9999em;  
  background-image: url("bilder/logoschrift.png");  
  background-repeat: no-repeat;  
  width:            622px;  
  height:           82px;  
}
```

Zunächst wird der eigentliche Text per `text-indent` aus dem sichtbaren Bereich herausgeschoben. Danach wird das Bild, das das grafische Logo enthält, als Hintergrundbild eingebunden, und es wird dafür gesorgt, dass es nicht wiederholt wird. Die Angaben für `height` und `width` entsprechen der Größe des Logos.



Es sollte sich von selbst verstehen, dass Sie bei der Definition von Bildern im Quelltext prinzipiell eine passende Kurzbeschreibung im `alt`-Attribut des `img`-Tags angeben. Diese wird vom Browser dargestellt, wenn das Bild nicht angezeigt werden kann.

In Webkit-basierten Browsern funktioniert dies allerdings nur, wenn Sie dem Bild entweder im HTML-Code oder per Stylesheet eine ausreichend große Höhe und Breite zugewiesen haben.

Lösung 2: Gilder/Levin

Verwenden Sie die Gilder/Levin-Technik zur Ersetzung von Text durch Bilder. Hierfür muss zunächst ein leeres `span`-Element in die Überschrift eingefügt werden:

```
<h1><span></span>Überschriften mit Stil</h1>
```

Zu Beginn wird per `position: relative` ein neuer Positionierungskontext geschaffen. Danach legen Sie, wie in der ersten Lösung, die Höhe und Breite des Bilds für die Ausmaße der Überschrift fest:

```
h1 {
  position: relative;
  height: 60px;
  width: 200px;
}
```

Indem wir Breite und Höhe mit 100% angeben, erhält das `<span>`-Element die gleichen Maße wie sein Elternelement (die Überschrift). Dadurch liegt es visuell vor der Überschrift. Der letzte Schritt besteht darin, das Ersatzbild mithilfe der Eigenschaft `background` einzubinden:

```
h1 span {
  background: url("bilder/logoschrift.png") no-repeat;
  position: absolute;
  width: 100%;
  height: 100%;
}
```



Abbildung 4-6: Oben: die ursprüngliche Überschrift, unten: die Überschrift wurde durch eine Grafik ersetzt

Lösung 3: FIR

Verwenden Sie das Fahrner Image Replacement (FIR). Im Gegensatz zu Lösung 2 wird hier der gesamte Text der Überschrift mit einem zusätzlichen `span`-Element umgeben:

```
<h1><span>Überschrift mit Stil</span></h1>
```

Dessen Inhalt wird anschließend per `display: none` ausgeblendet. An seiner Stelle wird, wie bei den anderen Lösungen auch, ein Hintergrundbild für das `h1`-Element angezeigt:

```
h1 span {
  display: none;
}

h1 {
  background: url("bilder/logoschrift.jpg") no-repeat;
  width:      100%;
  height:     100%;
}
```

Lösung 4: CSS3-Eigenschaft `content`

Während dieses Buch geschrieben wurde, unterstützten Safari 5, Chrome 11, Opera 11 und deren Nachfolger außerdem bereits eine Technik, die bald alle anderen Methoden ersetzen könnte.

Anstatt zusätzliche `span`-Elemente einzufügen, den Text zu verschieben, auszublenden oder zu überlagern, wird hierbei die CSS-Eigenschaft `content` verwendet.

Der Code sieht etwa so aus:

```
h1 {
  content: url("bilder/logoschrift.jpg");
}
```

Diskussion

Die Ersetzung von Text durch Bilder war lange Zeit die einzige Möglichkeit, Überschriften und andere Textelemente mit eigenen Schriften zu gestalten. Das hat sich aufgrund der weitreichenden Browserunterstützung von Webfonts mittlerweile geändert. Sobald Sie jedoch eine Schrift benutzen müssen, die Sie nicht lizenzfrei einsetzen können, ist der Einsatz von Webfonts entweder rechtlich fragwürdig oder ziemlich teuer. Daher haben die Verfahren zur Textersetzung auch heute noch ihre Gültigkeit.

Mittlerweile existieren unzählige Methoden zur Ersetzung von Text durch Bilder. Alle haben ihre Vor- und Nachteile. Einige verwenden leere (nicht semantische) HTML-Elemente ohne weitere Bedeutung, während bei anderen die Zugänglichkeit leidet. Hier ein kurzer Überblick über die Eigenheiten der einzelnen Methoden:

Phark (Off-Left-Technik)

Diese Methode ist bei Weitem die beliebteste. Sie benötigt keine zusätzlichen Markup-Elemente und ist auch für Menschen zugänglich, die beispielsweise ein Bildschirmlesegerät oder eine Braillezeile verwenden.

Ein Nachteil besteht darin, dass Menschen, die die Darstellung von Bildern im Browser deaktiviert haben, die Textüberschrift nicht sehen können. Ein weiteres Problem tritt auf, wenn der aus dem Ansichtsbereich geschobene Text Links enthält. Dann kann es passieren, dass sich die Gestaltung der Links unschön im sichtbaren Bereich bemerkbar macht.

Gilder/Levin

Diese Lösung nutzt die CSS-Stapelungsreihenfolge (siehe Rezept 2.21) aus. Darin ist definiert, dass Nachfahrenelemente visuell »vor« ihrem Vorfahrenelement dargestellt werden, falls es sonst zu Überschneidungen kommen würde. Das machen wir uns hier zunutze, um das Ersatzbild vor der eigentlichen Überschrift zu platzieren.

FIR (Fahrner Image Replacement)

Todd Fahrner, nach dem diese Methode benannt ist, gehört zu den Ersten, die die Technik der Bildersetzung bekannt gemacht haben. Ihre recht einfache Umsetzung hat diesen Weg bei Webentwicklern sehr beliebt gemacht.

Das Problem besteht darin, dass bei der Verwendung von `display: none` der Text der Überschrift auch dann nicht angezeigt wird, wenn die Webseite beispielsweise mit Screenreadern oder einer Braillezeile dargestellt wird.

CSS3-Eigenschaft content

Im »*Generated and Replaced Content*«-Modul der CSS3-Spezifikation wird beschrieben, wie die Möglichkeiten der Eigenschaft `content` auch außerhalb der Pseudoelemente `:before` und `:after` verwendet werden können. Obwohl dieser Teil von CSS3 seit 2003 nur als Arbeitsentwurf (Working Draft) existiert, stehen einige seiner Merkmale bereits jetzt zur Verfügung.

Alle hier gezeigten Techniken können ein HTML-Element durch ein Bild ersetzen bzw. mit einer Grafik überlagern. Welche Methode am Schluss für Sie die richtige ist, müssen Sie selbst entscheiden.

Siehe auch

Informationen zum Einfügen von Inhalten mit CSS3 unter <http://www.w3.org/TR/css3-content/#inserting3> sowie <http://my.opera.com/ODIN/blog/css-3-image-replacement>; die CSS 2.1-Spezifikation zur Eigenschaft `z-index` unter <http://www.w3.org/TR/CSS21/zindex.html>; eine umfangreiche Liste mit Methoden zur Textersetzung von Jens Meiert unter <http://meiert.com/de/publications/articles/20050513/>; einen Artikel zur Barrierefreiheit von Textersetzungstechniken unter <http://www.einfach-fuer-alle.de/artikel/image-replacement-nicht-barrierefrei/>.

4.7 Bilder als Panorama präsentieren

Problem

Die Breite eines Bilds soll dynamisch an die Größe des Browserfensters angepasst werden, wie in Abbildung 4-7 und Abbildung 4-8 zu sehen ist.



Abbildung 4-7: Ein Foto im Browser

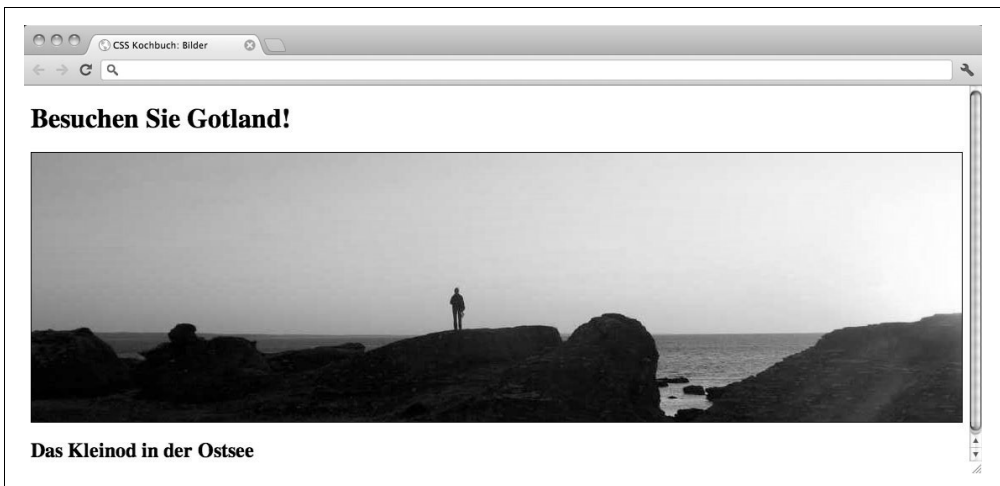


Abbildung 4-8: Im vergrößerten Browserfenster ist mehr von dem Panoramabild zu sehen

Lösung

Umgeben Sie das gewünschte Bild mit einem div-Element:

```
<h1>Besuchen Sie Gotland!</h1>
<div></div>
<h2>Das Kleinod in der Ostsee</h2>
```

Definieren Sie das gleiche Bild als Hintergrund für das div-Element und positionieren Sie es in der rechten oberen Ecke des Blockelements. Danach verbergen Sie das ursprüngliche Bild, wie hier gezeigt.

```
div {
    background-image: url("bilder/gotland_breit.jpg");
    background-repeat: no-repeat;
    background-position: top right;
    height: 320px;
    border: 1px solid black;
    max-width: 1200px;
}

div img {
    display: none;
}
```

Wird das Bild als Hintergrund eines Blockelements benutzt, wird seine Größe basierend auf der Größe des Browserfensters angepasst.

Diskussion

Um ein Panoramabild zu präsentieren, benötigen Sie erst einmal ein breites Foto. Danach müssen Sie das Bild in der rechten oberen Ecke positionieren, damit es an die Größe des Browserfensters angepasst werden kann. Die Verwendung der Eigenschaft `max-width` verhindert, dass das div-Element breiter wird als das Bild selbst.

In dieser Lösung wird das gleiche Foto im HTML- und auch im CSS-Code benutzt. Kann das Bild nicht geladen werden, wird so der im alt-Attribut angegebene Ersatztext angezeigt. Hierdurch wird eine zweite HTTP-Anfrage an den Server gesendet. Da sich das Bild aber bereits im Browsercache befindet, muss nur einmal geladen werden.

Siehe auch

<http://www.creighton.edu/~jaypl/oldpage/panhow.html> für weitere Informationen zum Erstellen von Panoramabildern; die CSS 2.1-Spezifikation zur Eigenschaft `max-width` unter <http://www.w3.org/TR/CSS21/visudet.html#propdef-max-width>.

4.8 Verschiedene Bildformate miteinander kombinieren

Problem

Die drei in Abbildung 4-9 gezeigten Einzelbilder sollen zu der Montage in Abbildung 4-10 zusammengefügt werden. Hierbei soll für die Einzelbilder jeweils das bestmögliche Bildformat verwendet werden.

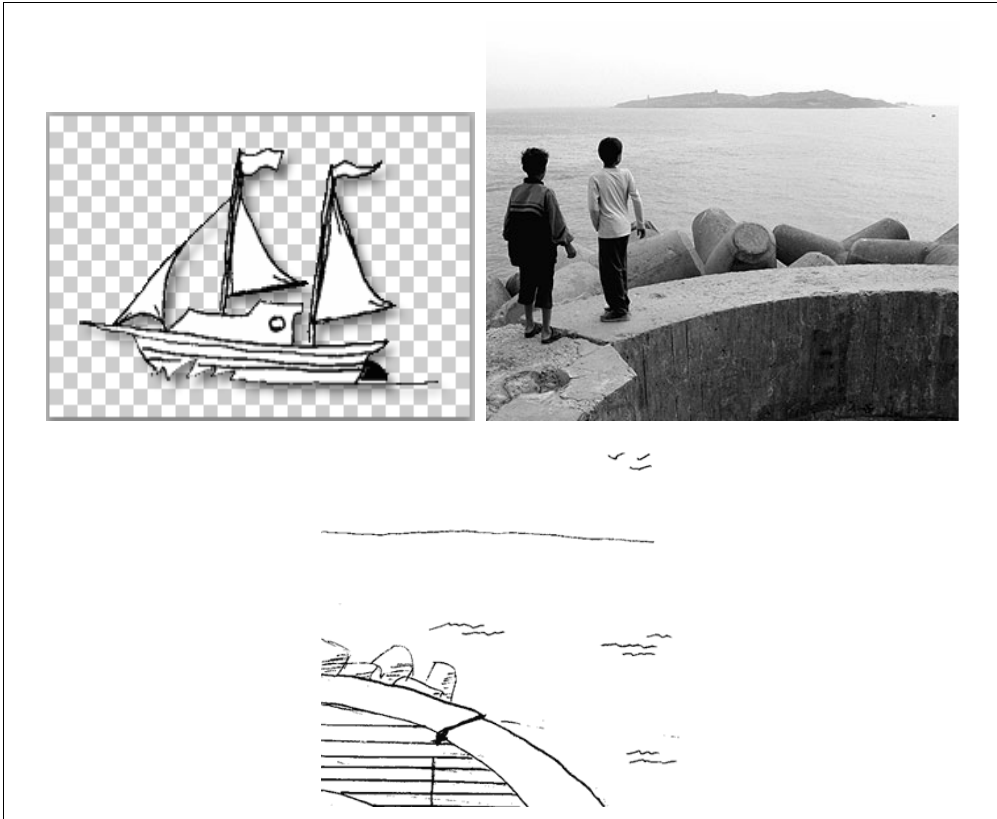


Abbildung 4-9: Drei Einzelbilder sollen zu einer Montage kombiniert werden

Lösung

Umgeben Sie die drei Bilder mit einem div-Container und platzieren Sie sie in der Reihenfolge ihrer Benutzung im HTML-Code der Seite, wie hier gezeigt:

```
<div id="montage">



</div>
```

Danach können Sie den Container gestalten und die einzelnen Bilder anhand ihrer alt-Attribute auswählen, um zusätzliche Stile zu definieren:

```
div#montage {
  position: relative;
  width: 675px;
  height: 326px;
  border: 1px solid black;
}

img#kinder, img#zeichnung {
  float: left;
}

img#kinder {
  margin-top: 22px;
}

img#boot {
  position: absolute;
  left: 260px
}
```

Zur besseren Unterscheidung der verschiedenen Elemente haben wir den ID-Selektoren hier zusätzlich das entsprechende Element vorangestellt. Das ist zwar nicht unbedingt nötig, kann aber die Wartung des Stylesheets erleichtern.

Diskussion

Durch die Kombination verschiedener Grafikformate lassen sich einige interessante Effekte erzielen. Anstatt das Bild als Ganzes in Photoshop zu kombinieren, verwenden wir hier drei Einzelbilder, die für ihren Zweck jeweils am besten geeignet sind.

Daher schaffen wir zunächst per `position: relative` den nötigen Positionierungskontext für das Boot. Es folgen explizite Angaben zur Größe des `div`-Containers, die sich aus den Größen des linken und rechten Bilds (Kinder bzw. Zeichnung) ableiten lässt. Dies ist notwendig, weil wir die Bilder als Floats definieren wollen. Außerdem können wir die gesamte »Komposition« mit einem Rahmen umgeben, der den nötigen visuellen Zusammenhalt für die drei Bilder herstellt.

Die Breite (`width`) erhält man einfach durch Addition der beiden Bildbreiten, während der Wert für `height` etwas höher ausfällt, weil wir die linke Bildhälfte ein Stück nach unten verschieben müssen.



Es gilt als gängige Praxis, Bilder (`img`-Elemente) mit einem `alt`-Attribut zu versehen, damit auch Menschen mit Sehproblemen wissen, was dargestellt wird, und eine Alternative zur Verfügung steht, falls die Bilder nicht geladen werden können.

Zunächst definieren wir die beiden großen Bilder als Floats. Hierdurch werden sie »nahtlos« nebeneinander angeordnet. Das funktioniert allerdings nur, solange der `div`-Container eine entsprechende Breite hat. Wenn Sie zusätzliche Rahmen, Innen- oder Außenabstände verwenden wollen, muss die Containerbreite gegebenenfalls angepasst werden.

Wir arbeiten hier mit der Möglichkeit, mehrere Selektoren, durch Kommata voneinander getrennt, zu gruppieren. Dadurch brauchen wir die Deklaration nur einmal anzugeben.



Anstelle der ID-Selektoren hätten Sie auch Nachfahrenselektoren und Selektoren für benachbarte Geschwisterelemente verwenden können. Für Browser, die bereits die neuen CSS3-Selektoren verstehen, wären auch Dinge wie `nth-child()` möglich.

Weil die Bilder unterschiedlich hoch sind, muss der Abstand zur Oberkante für das Foto angepasst werden. Den Wert für `margin-top` bekommen wir, indem wir die Differenz der Höhen (19 Pixel) und den unteren Leerraum des rechten Bilds (ca. 3 Pixel) addieren. Dadurch entsteht der Eindruck, dass beide Bilder eine Einheit bilden.

Es folgen die Deklarationen, mit denen das Boot an die richtige Stelle gerückt wird. Es soll der Eindruck entstehen, als bewege es sich aus der Zeichnung heraus in das Foto hinein. Weil wir zu Beginn `position: relative` den `div`-Container als Positionierungskontext festgelegt haben, können wir das Boot nun mithilfe absoluter Positionierung frei darin platzieren.

Ohne weitere Angaben werden absolut positionierte Elemente in der linken oberen Ecke des nächsten umgebenden Blocks mit Positionierungskontext (oder des Ansichtsbereichs) platziert. Da der Abstand nach oben bereits ausreicht, brauchen wir hier nichts zu ändern. Um das Boot horizontal an die richtige Stelle zu schieben, verwenden wir die Eigenschaft `left`. Sie gibt an, dass das Boot um 220 Pixel nach rechts verschoben werden soll. Das Ergebnis sehen Sie in Abbildung 4-10.

Bildformate

Die drei vorherrschenden Bildformate für das Web sind JPEG, GIF und PNG. Während das JPEG-Format am besten für Fotos und fotorealistische Abbildungen geeignet ist (das linke Bild mit den Kindern), werden für Grafiken mit wenigen Farben und klaren, einfachen Linienführungen besser GIF-Dateien verwendet (die Zeichnung auf der rechten Seite). Das GIF-Format ermöglicht es außerdem, eine Farbe transparent darzustellen. Mit PNG-Dateien ist es möglich, einen sogenannten Alphakanal zu definieren, der neben den Farbwerten für jeden Bildpunkt außerdem festlegt, wie durchscheinend er ist. Dadurch lassen sich zum Beispiel fließende Farbverläufe mit Transparenzen und Schattenwürfe erzeugen, die unabhängig von einer möglichen Hintergrundfarbe funktionieren (z.B. der Schattenwurf für das Boot).



Abbildung 4-10: Die fertige Bildmontage

Siehe auch

Das Rezept 1.4 zur Gruppierung von Selektoren; die Rezepte zu den verschiedenen Selektoren in Kapitel 1, *Allgemeines*; das Grundrezept 2.23 zur Verwendung von Floats; das Rezept 4.9 zur Verwendung von Sprites.

4.9 Sprites zum Speichern von Icons verwenden

Problem

Um das Laden Ihrer Seiten zu beschleunigen, sollen alle Icons als sogenannte Sprites in der gleichen Grafikdatei gespeichert werden.

Lösung

Nach der Erstellung der Icons werden diese nicht in separaten Dateien gespeichert. Stattdessen ordnen Sie sie in einer gemeinsamen Bilddatei über- oder nebeneinander an (siehe Abbildung 4-11). Diese Datei wird als Hintergrundbild für das gewünschte Element eingebunden. Durch die Verwendung der Eigenschaft `background-position` können Sie steuern, welches Symbol angezeigt wird.

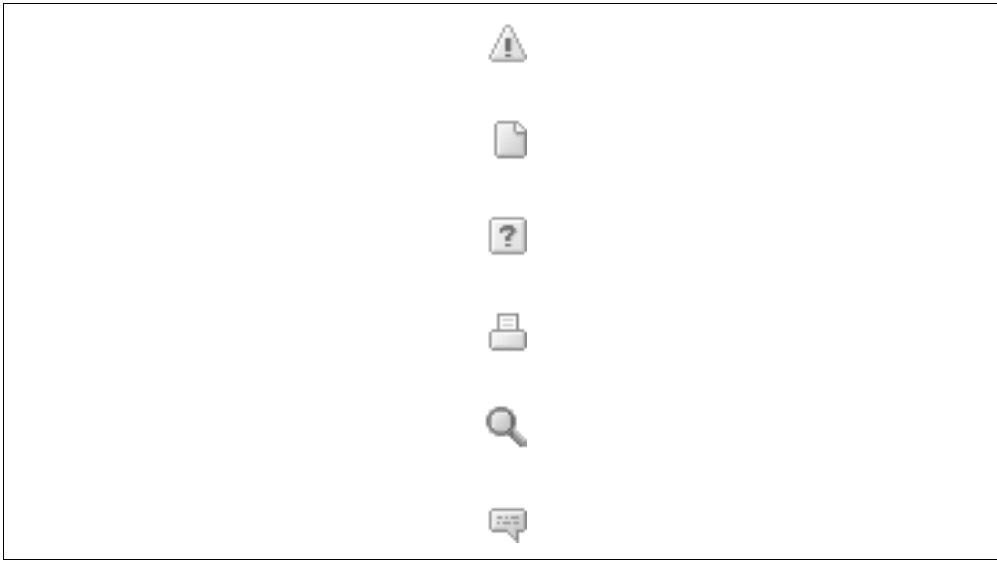


Abbildung 4-11: Die Bilddatei mit Icons



Sorgen Sie dafür, dass die Symbole immer gleich groß sind und den gleichen Abstand zueinander haben (siehe Abbildung 4-11), ansonsten wird die Positionsangabe im Stylesheet schnell zu einem ungewollten Ratespiel.

Danach markieren Sie die gewünschten Elemente mit einem gemeinsamen class-Attribut sowie einem eigenen id-Attribut, wie hier gezeigt:

```
<h2 class="icon" id="warnung">Warnungen</h2>
```

Hierdurch können wir den linken Innenabstand und die Quelldatei für die Sprites für alle Überschriften gemeinsam definieren. Danach brauchen Sie nur noch die Positionen des jeweiligen Icons für die einzelnen Überschriften anzugeben (siehe Abbildung 4-12):

```
h2.icon {
  padding-left: 24px;
  background-image: url("bilder/sprites.gif");
  background-repeat: no-repeat;
}

h2#warnung { background-position: -18px -12px; }
h2#fragen { background-position: -18px -48px; }
h2#kommentar { background-position: -18px -84px; }
h2#dokument { background-position: -18px -120px; }
h2#druck { background-position: -18px -154px; }
h2#suche { background-position: -18px -192px; }
```

Da die Icons in der Grafikdatei untereinanderstehen, müssen wir hier die vertikale Position mit negativen Werten angeben.

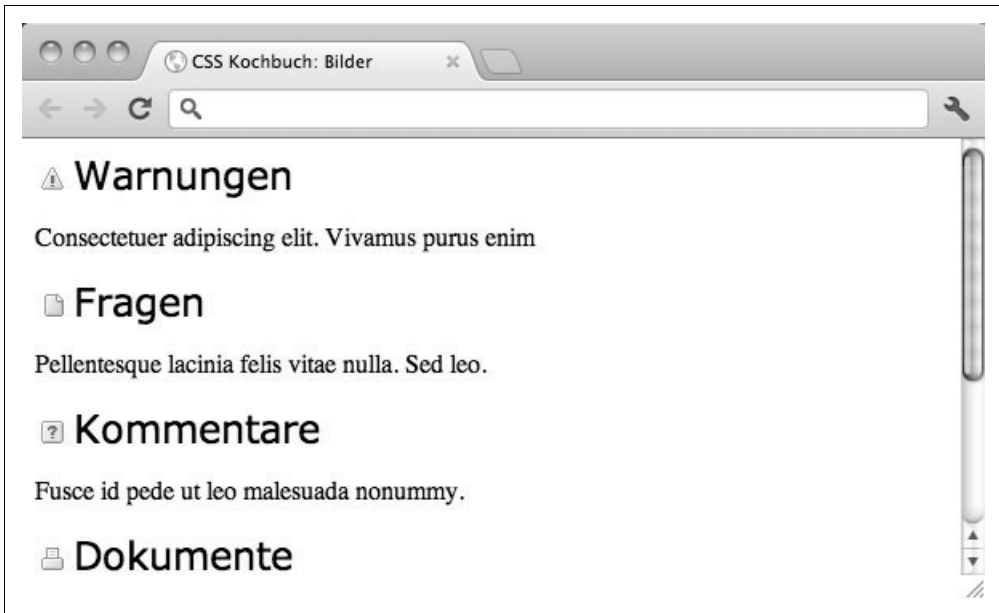


Abbildung 4-12: Durch die Positionierung des Hintergrundbilds mit den Sprites erhält jede Überschrift ihr eigenes Symbol

Diskussion

Einige Entwickler verwenden das gleiche Bild in einer Website immer wieder. Auf diese Weise machen sie sich das lokale Zwischenspeichern der Datei durch den Browser (Caching) zunutze. Der Einsatz von Sprites treibt dieses Konzept noch ein wenig weiter. Durch das Speichern mehrerer Einzelbilder in einer gemeinsamen Datei können Webentwickler die Anfragen an den Webserver gering halten. Diese Lösung eignet sich besonders für Websites mit mittlerem bis hohem Traffic.

Typische Beispiele sind hier die »Platzhirsche« Google und Amazon. Beide Websites (und viele mehr) setzen Sprites ein, um die Ladegeschwindigkeit ihrer Seiten zu erhöhen.

Siehe auch

Den Artikel zu CSS-Sprites unter <http://www.alistapart.com/articles/sprites>; den Artikel von Marc Pentermann zu CSS-Sprites unter <http://www.webanhalter.de/1368-css-sprites.html>; die Original-Sprites von Google und Amazon unter http://www.google.de/images/srpr/nav_logo14.png bzw. <http://tinyurl.com/amazon-sprites>; den Artikel *The Mystery Of CSS Sprites: Techniques, Tools And Tutorials* unter <http://coding.smashingmagazine.com/2009/04/27/the-mystery-of-css-sprites-techniques-tools-and-tutorials/>; das Rezept 13.20 zur Speicherung von Daten direkt im Stylesheet.

Listen

Einführung

Listen können Menschen auf verschiedene Arten dabei behilflich sein, den Überblick zu behalten und sich auf das Wesentliche zu konzentrieren. Das beginnt beim einfachen Einkaufszettel und reicht bis zur Radiostation, die gerade die 100 schlimmsten Songs aller Zeiten vorstellt. Beim Webdesign ist das genauso: HTML-Listen helfen bei der Präsentation der organisierten Inhalte Ihrer Site, indem die Schlüsselemente strukturiert zusammengefasst werden.

HTML-Listen bieten aufgrund ihrer unterschiedlichen Darstellung im Dokument gewisse Vorteile. Die Listeneinträge werden normalerweise eingerückt und mit einem vorangestellten Aufzählungszeichen (Marker, Bullet) dargestellt. Für ungeordnete Listen ist der Marker normalerweise ein gefüllter Kreis (siehe Abbildung 5-1), für geordnete Listen eine Zahl.

Schon mit wenigen Zeilen HTML-Code können Sie auch ohne ein Bildbearbeitungsprogramm eine Liste mitsamt Aufzählungszeichen erstellen. Mit CSS können diese Listen noch ansprechender gestaltet werden.

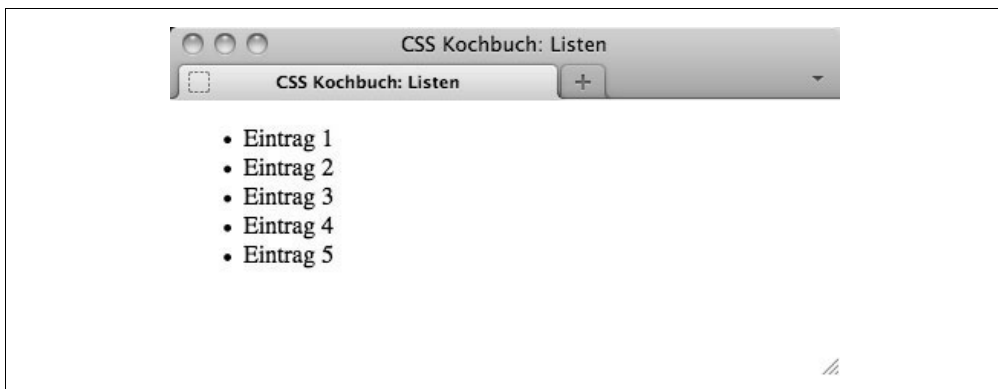


Abbildung 5-1: Standarddarstellung einer Liste

Darüber hinaus können Sie die Darstellung einer Liste an das allgemeine Design einer Webseite anpassen, ohne sich dabei auf die Standarddarstellung des Browsers verlassen zu müssen.

In diesem Kapitel beschäftigen wir uns mit den Möglichkeiten, die Nummerierung einer Liste anzupassen, ein eigenes Bild als Marker zu verwenden, die Einrückungen browserübergreifend zu vereinheitlichen, und vielem mehr.

5.1 Browserübergreifende Einrückungen

Problem

Sie wollen eine browserübergreifende Darstellung der Listeneinrückungen erreichen.

Lösung

Legen Sie für das `ul`-Element Werte für die Eigenschaften `margin-left` und `padding-left` fest:

```
ul {  
    margin-left: 2em;  
    padding-left: 0;  
}
```

Diskussion

Browser setzen zwei verschiedene Methoden ein, um die Einrückung einer Liste zu erreichen. Einige steuern die Einrückung über die Eigenschaft `padding-left` (linker Innenabstand), während andere stattdessen `margin-left` (linker Außenabstand) verwenden.

Damit die Darstellung in allen Browsern einheitlich vorgenommen wird, müssen Sie die Werte für beide Eigenschaften explizit festlegen. Dabei sollte die Einrückung selbst nur mit einer der beiden Eigenschaften definiert werden. Wenn Sie die Einrückung auf beide Eigenschaften aufteilen, wird die Darstellung in verschiedenen Browsern inkonsistent.

Versehen Sie beide Eigenschaften für eine Liste, die direkt vom `body`-Element abstammt, mit dem Wert 0, kann es passieren, dass der Browser die Aufzählungszeichen außerhalb des sichtbaren Bereichs darstellt. Um sicherzugehen, dass der Benutzer die Marker auch sieht, sollten Sie entweder den linken Innenabstand (`padding-left`) oder den linken Außenabstand (`margin-left`) für `ul`-Elemente auf eine Breite von mindestens 1em festlegen. In unserer Lösung haben wir uns für den Außenabstand entschieden.

Siehe auch

Die CSS 2.1-Spezifikation zur Eigenschaft `padding` unter <http://www.w3.org/TR/CSS21/box.html#propdef-padding>; die CSS 2.1-Spezifikation zur Eigenschaft `margin` unter <http://www.w3.org/TR/CSS21/box.html#propdef-margin>; das Rezept 13.16 zur Definition eines Reset-Stylesheets.

5.2 Spezielle Aufzählungszeichen verwenden

Problem

Die Standarddarstellung der Liste soll so verändert werden, dass beispielsweise ein anderes Aufzählungszeichen oder eine spezielle Nummerierung benutzt wird, wie in Abbildung 5-2 gezeigt.

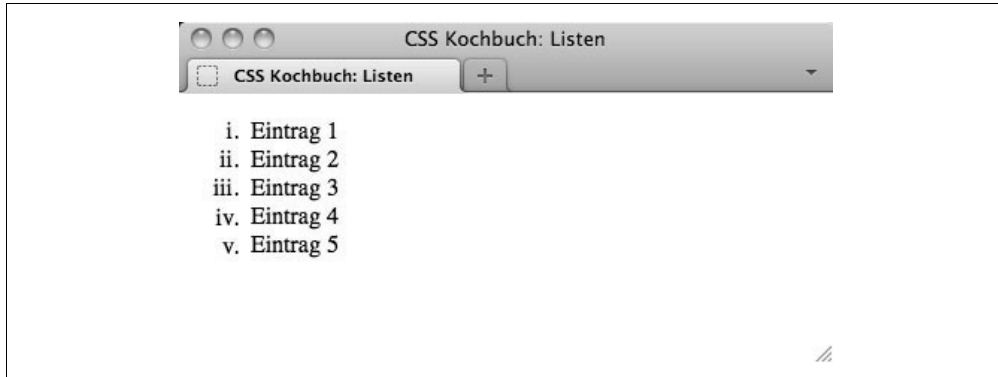


Abbildung 5-2: Als Aufzählungszeichen werden in dieser Liste kleingeschriebene römische Zahlen benutzt

Lösung

Verwenden Sie die Eigenschaft `list-style-type`, um das Aufzählungszeichen oder den Zähler für die Liste anzupassen:

```
ul {  
    list-style-type: lower-roman;  
}
```

Diskussion

Die CSS 2.1-Spezifikation definiert verschiedene Stile für die möglichen Aufzählungszeichen. In der Regel verwenden die Browser für die einzelnen Schachtelungsebenen unterschiedliche Marker. Anstelle dieses Systems können Sie für die Unterlisten mit `list-style-type` auch eigene Marker definieren. Die möglichen Werte für `list-style-type` sehen Sie in Tabelle 5-1.

Tabelle 5-1: Die verschiedenen Stile für Aufzählungszeichen

Stil/Wert	Beschreibung
square	Normalerweise ein ausgefülltes Quadrat, allerdings ist die exakte Darstellung nicht weiter definiert.
disc	Normalerweise ein ausgefüllter Kreis, allerdings ist die exakte Darstellung nicht weiter definiert.
circle	Normalerweise ein nicht ausgefüllter Kreis, allerdings ist die exakte Darstellung nicht weiter definiert.

Tabelle 5-1: Die verschiedenen Stile für Aufzählungszeichen (Fortsetzung)

Stil/Wert	Beschreibung
decimal	Eine fortlaufende Zahlenfolge, die mit 1 beginnt, gefolgt von 2, 3, 4 usw.
decimal-leading-zero	Eine Reihe fortlaufender Zahlen mit einer oder mehreren vorangestellten Nullen (beginnend mit 01, gefolgt von 02, 03, 04 usw.). Die Anzahl der führenden Nullen kann der Anzahl der in der Liste benutzten Ziffern entsprechen. Bei einer Liste mit 5.876 Einträgen wäre die erste Zahl dann 0001.
lower-roman	Beginnt mit einer Reihe kleingeschriebener fortlaufender römischer Zahlen.
upper-roman	Beginnt mit einer Reihe großgeschriebener fortlaufender römischer Zahlen.
lower-alpha	Beginnt mit einer Reihe kleingeschriebener ASCII-Buchstaben.
upper-alpha	Beginnt mit einer Reihe großgeschriebener ASCII-Buchstaben.
lower-latin	Beginnt mit einer Reihe kleingeschriebener ASCII-Buchstaben.
upper-latin	Beginnt mit einer Reihe großgeschriebener ASCII-Buchstaben.
lower-greek	Beginnt mit einer Reihe klassischer griechischer Buchstaben, beginnend mit alpha, gefolgt von beta, gamma usw.
hebrew	Beginnt die Zählung mit traditionellen hebräischen Buchstaben.
hiragana	Beginnt die Zählung mit dem klassischen japanischen Hiragana-System.
katakana	Beginnt die Zählung mit dem klassischen japanischen Katakana-System.
hiragana-iroha	Beginnt die Zählung mit dem klassischen japanischen Hiragana-Iroha-System.
katakana-iroha	Beginnt die Zählung mit dem klassischen japanischen Katakana-Iroha-System.
none	Es wird kein Marker dargestellt.

Wie in der Lösung zu Rezept 5.1 gezeigt, müssen Sie die Eigenschaft `list-style-type` nicht für die Listenelemente (`li`) selbst definieren, sondern können ihren Wert auch für die gesamte Liste (`ul`, `ol`) angeben. Die Vererbung erledigt den Rest.

Aufzählungszeichen verbergen

Gelegentlich sollen die Listeneinträge ohne Aufzählungszeichen dargestellt werden. Genau für diesen Zweck gibt es den Wert `none`. Um den Listenmarker beispielsweise für eine Aufzählungsliste zu verbergen, reicht es daher aus, wenn Sie schreiben:

```
ol li { list-style-type: none; }
```

Siehe auch

Das Rezept 5.5 zur Verwendung eigener Bilder als Aufzählungszeichen; das Kapitel 12 in *CSS – Das umfassende Handbuch* von Eric A. Meyer (O'Reilly 2007); das Rezept 1.18 zur Vererbung von Werten.

5.3 Eigene textbasierte Aufzählungszeichen für Listeneinträge verwenden

Problem

Sie wollen für Listeneinträge Ihre eigenen, textbasierten Aufzählungszeichen benutzen. Verschachtelte Listen sollen hierbei ein eigenes Zeichen erhalten.

Lösung

Verbergen Sie das reguläre Aufzählungszeichen. Fügen Sie dann mithilfe des Pseudoelements `:before` und der Eigenschaft `content` den gewünschten textbasierten Marker (spitze Klammern in unserem Beispiel) vor den Einträgen ein (siehe Abbildung 5-3):

```
ul {
  list-style: none;
  margin: 0;
  padding-left: 1em;
}

li {
  margin-bottom: 0.25em;
  padding: 0;

  border: 1px solid green;
}

li:before {
  content: "\00BB" " ";
}

li > ul > li:before {
  content: "\203A" " ";
}
```

Die erste Regel gilt für die gesamte Liste. Hier wird zunächst mit der Kurzschrifteigenschaft `list-style` festgelegt, dass der Browser keine Aufzählungszeichen erzeugen soll.

Eventuell definierte Außenabstände werden per `margin: 0` zurückgesetzt. Mit der Eigenschaft `padding-left` geben wir zudem ausdrücklich an, wie groß die Einrückung für die Listeneinträge ausfallen soll.

Auf ähnliche Weise definieren wir in der nächsten Regel die Innen- und Außenabstände für die einzelnen Einträge. Um sie visuell voneinander zu trennen, erhalten sie einen unteren Außenabstand von `0.25em`. Möglicherweise vom Browser vordefinierte Innenabstände werden per `padding: 0` überschrieben.

In der folgenden Regel erzeugen wir mithilfe des Pseudoelements `:before` und der Eigenschaft `content` das gewünschte Aufzählungszeichen.

Weil für die zweite Verschachtelungsebene der Liste ein eigenes Zeichen eingesetzt werden soll, verwenden wir in der letzten Regel Kindselektoren, um die Definition mit einer höheren Spezifität zu versorgen. Das Ergebnis sehen Sie in Abbildung 5-3.

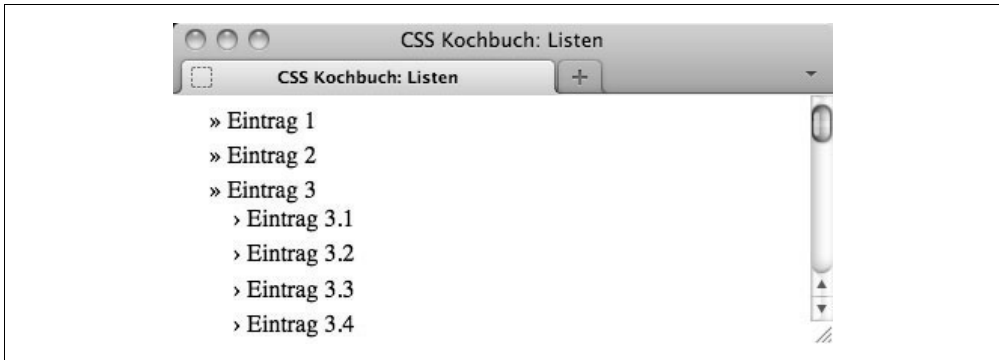


Abbildung 5-3: Unterschiedliche textbasierte Marker für Listeneinträge

Diskussion

Indem Sie der Eigenschaft `list-style` für das `ul`-Element den Wert `none` zuweisen, schalten Sie die Darstellung des Aufzählungszeichens ab, das normalerweise links vor den einzelnen Einträgen steht. Das funktioniert, weil der Wert von `list-style` automatisch an die enthaltenen Elemente vererbt wird.

Stattdessen fügen wir mithilfe des Pseudoelements `:before` unsere eigenen Zeichen ein. Die so erzeugten Zeichen werden innerhalb des Listeneintrags angezeigt. Hierfür werden die Inhalte im HTML-Dokument gerade so weit verschoben, dass genügend Platz ist, um die zusätzlichen Zeichen darzustellen.

Ohne weitere Anpassungen würden das neue Aufzählungszeichen und der eigentliche Listeneintrag ohne Zwischenraum dargestellt. Um das zu verhindern, fügen wir nach dem Marker ein zusätzliches Leerzeichen ein.

Sobald die Vorarbeiten abgeschlossen sind, können wir uns dem Einfügen unseres eigenen Aufzählungszeichens widmen. Wir verwenden hier das Pseudoelement `:before` mit der Eigenschaft `content`. Der Wert besteht aus zwei Zeichen: der Unicode-Schreibweise für das gewünschte Sonderzeichen und einem Leerzeichen.

In der Problembeschreibung wurde verlangt, dass Untereinträge ein eigenes Aufzählungszeichen erhalten. Da der Selektor `li:before` aber für alle Listeneinträge (auch die Untereinträge) gilt, muss dieser überschrieben werden.

Dabei hilft uns die Tatsache, dass Kindselektoren eine höhere Spezifität besitzen als einfache Elementselektoren. Durch die Schreibweise `li > ul > li:before` sprechen wir nur die Listenelemente an, die in einem anderen Element verschachtelt sind. Gleichzeitig hat dieser Selektor mehr Gewicht, wodurch wir für die Unterliste ein eigenes Aufzählungszeichen definieren können.

Die Angaben für content

Sonderzeichen müssen laut CSS 2.1-Spezifikation als Unicode-Werte (ISO 10646) angegeben werden. Anstelle der gebräuchlichen HTML4-Entities (z.B. ») müssen Sie für die Eigenschaft content jedoch die hexadezimale Entsprechung des Unicode-Zeichens verwenden. Dieses wird in CSS mit einem Backslash geschützt:

```
content: "\00BB";
```

Bei Drucklegung des Buchs funktionierte diese Lösung mit allen aktuellen Browsern außer Internet Explorer 7.



Sie können die gewünschten Zeichen unkodiert an die Eigenschaft content übergeben. Hierbei müssen allerdings zwei Dinge gegeben sein: Zum einen muss das Stylesheet selbst in UTF-8 kodiert sein. Zum anderen muss es entweder mit einem passenden HTTP-Header ausgeliefert werden (Content-Type: text/css; charset=UTF-8) oder im Dokument selbst mit einer passenden @charset-Direktive gekennzeichnet werden.

Wollen Sie anstelle eines statischen Aufzählungszeichens eine fortlaufende Nummerierung einsetzen, benötigen Sie die Eigenschaft counter, die in Kapitel 13, *Tipps und Tricks*, im Rezept zu erzeugten Inhalten beschrieben wird.

Fallback-Lösungen

Soll Ihre Website auch für Dinosaurier funktionieren, sollten Sie nicht mit automatisch erzeugten Inhalten arbeiten. Stattdessen können Sie, wie in Rezept 5.5 beschrieben, eine Grafik verwenden, oder Sie fügen das Aufzählungszeichen manuell vor den einzelnen Listeneinträgen ein:

```
<ul>
<li>&#187; Eintrag 1</li>
<li>&#187; Eintrag 2</li>
<li>&#187; Eintrag 3
  <ul>
    <li>&#8250; Eintrag 3.1</li>
    <li>&#8250; Eintrag 3.2</li>
    <!-- weitere Einträge -->
  </ul>
</li>
<!-- weitere Einträge -->
</ul>
```

Der größte Nachteil bei diesem Ansatz besteht darin, dass hier Inhalt und Darstellung übel vermischt werden. Das ist bei kurzen Listen, die sich selten ändern (z.B. in Navigationsmenüs) vermutlich nicht so schlimm. Aber sobald die Listen sehr lang oder dynamisch erzeugt werden, ist diese Methode nicht mehr praktikabel.

Eine weitere Möglichkeit besteht darin, nach dem Prinzip der »Graceful Degradation« einfach die Standardstile wieder hervorzuzaubern.

Siehe auch

Die CSS 2.1-Spezifikation zur Verwendung von Escape-Zeichen unter <http://www.edition-w3c.de/TR/1998/REC-CSS2-19980512/kap04.html#heading-4.1.3>; Informationen zur Verwendung hexadezimaler Werte für ASCII- und Unicode-Zeichen unter <http://www.alan-wood.net/demos/ansi.html>; das Rezept 5.5 zur Verwendung von eigenen Grafiken als Aufzählungszeichen; das Rezept 3.17 zur Definition eigener Anführungszeichen; die Website des W3C zur Angabe der Zeichenkodierung im HTTP-Header unter <http://www.w3.org/International/questions/qa-css-charset.de.php>; das Rezept 13.6 zur Angabe der richtigen Zeichenkodierung von CSS-Dateien; das Rezept 13.7 zu erzeugten Inhalten.

5.4 Aufzählungszeichen mit einer eigenen Farbe versehen

Problem

Das Aufzählungszeichen für Listeneinträge soll in einer anderen Farbe dargestellt werden als die Einträge selbst.

Lösung 1

Entfernen Sie das reguläre Aufzählungszeichen und fügen Sie, wie im vorigen Rezept erklärt, mithilfe des Pseudoelements `:before` einen eigenen Marker ein. Für dieses Beispiel verwenden wir eine simple ungeordnete Liste:

```
<ul>
  <li>Eins</li>
  <li>Zwei</li>
  <li>Drei</li>
</ul>
```

Um das Aufzählungszeichen zu entfernen, benutzen wir diesmal die Kurzschrifteigenschaft `list-style`:

```
ul { list-style: none; }
```

Im nächsten Schritt fügen Sie Ihr eigenes Aufzählungszeichen ein, das Sie nun auch mit einem eigenen Wert für die Eigenschaft `color` versehen können:

```
li:before {
  content: "• ";
  color: red;
}
```

Lösung 2

Umgeben Sie den Text der Listeneinträge mit span-Tags:

```
<ul>
  <li><span>Eintrag 1</span></li>
  <li><span>Eintrag 2</span></li>
  ...
</ul>
```

Im Stylesheet legen Sie zunächst eine allgemeine Schriftfarbe für die Listeneinträge fest:

```
li {
  color: red;
}
```

Danach verwenden Sie einen Nachfahrenselektor, um die Textfarbe innerhalb der span-Elemente anzupassen.

```
li span {
  color: black;
}
```

Das Resultat sind Listeneinträge, deren Schrift schwarz dargestellt wird, während die Aufzählungszeichen rot eingefärbt werden.

Diskussion

Wenn Sie eine Schriftfarbe für Listenelemente definieren, gilt diese für den enthaltenen Text und das dazugehörige Aufzählungszeichen gleichermaßen. Das können wir ändern, indem wir, wie in Rezept 5.3 beschrieben, das vom Browser erzeugte Aufzählungszeichen ausblenden, ein eigenes definieren und diese über die Eigenschaft `color` mit einer eigenen Farbe versehen.

Wenn Ihr Auftraggeber sich nicht auf eine »Graceful Degradation« einlässt und Sie auch den IE7 unterstützen müssen, können Sie auf Lösung 2 zurückgreifen. Durch die Verwendung der `span`-Tags können Sie den Text von den Aufzählungszeichen trennen und mit eigenen Stildefinitionen versehen.

Allerdings sollte das Einfügen zusätzlicher Elemente in den HTML-Code nach Möglichkeit vermieden werden, um den Wartungsaufwand der Seite nicht unnötig zu erhöhen.

Das CSS3-Modul *Lists and Counters* definiert das neue Pseudoelement `::marker`. Die Idee ist gut gemeint, wird aber zum Entstehungszeitpunkt dieses Buches von keinem der wichtigen Browser unterstützt. Wir werden Sie in der folgenden Auflage darüber informieren, wenn sich daran etwas ändert.

Siehe auch

Das Rezept 3.17 zum Einfügen eigener Anführungszeichen, ohne hierfür zusätzlichen Markup-Code zu schreiben; das Rezept 5.3 zur Definition eigener textbasierter Aufzählungszeichen.

5.5 Eigene Grafiken als Aufzählungszeichen verwenden

Problem

Sie wollen eine eigene Grafik als Marker für Listeneinträge verwenden – zum Beispiel das in Abbildung 5-4 gezeigte Bild eines Karos.

Lösung 1

Verwenden Sie die Eigenschaft `list-style-image`, um eine Grafik für den Marker anzugeben:

```
li {  
    list-style-type: disc;  
    list-style-image: url('bilder/bullet.gif');  
}
```



Abbildung 5-4: Eigene grafische Marker für Listeneinträge

In der ersten Deklaration dieser Lösung geben wir an, welches Aufzählungszeichen verwendet werden soll, falls das Bild nicht geladen werden kann. In der zweiten Deklaration wird dann das Bild selbst geladen, das nach Möglichkeit anstelle des Aufzählungszeichens benutzt werden soll.

Lösung 2

Verbergen Sie das automatisch erstellte Aufzählungszeichen. Definieren Sie stattdessen ein passendes Hintergrundbild für die Listeneinträge:

```
ul { list-style: none; }  
  
li {  
    padding-left: 1em;  
    margin-left: -1em;
```

```

background-image: url("bilder/bullet.gif");
background-position: left center;
background-repeat: no-repeat;
}

```

Mit der Deklaration `list-style: none` sorgen wir zunächst dafür, dass die standardmäßigen Aufzählungszeichen nicht angezeigt werden. Stattdessen binden wir per `background-image` unser eigenes Bild ein. Die folgende Deklaration verschiebt das Bild an den linken Rand und zentriert es vertikal. Per `background-repeat: no-repeat` geben wir an, dass es nicht wiederholt (gekachelt) werden soll.

Ohne weiteres Zutun wird unser Aufzählungszeichen vom Text des Listeneintrags verdeckt. Das verhindern wir, indem wir den Inhalt mit der Eigenschaft `padding-left` ein Stück nach rechts verschieben. Um diese Verschiebung wieder auszugleichen, verwenden wir gleich darauf die Eigenschaft `margin-left`, um den Eintrag inklusive Aufzählungszeichen um die gleiche Distanz wieder nach links zurückzubewegen.

Diskussion

Gelegentlich ist es erwünscht, nicht die vom Browser vorgegebenen Aufzählungszeichen zu verwenden, sondern ein eigenes Symbol anzugeben. Wenn auch der Unicode-Zeichensatz das gewünschte Zeichen nicht enthält (siehe Rezept 5.3), müssen Sie auf eine Grafik zurückgreifen. CSS stellt hierfür die Eigenschaft `list-style-image` bereit, die in Lösung 1 beschrieben wird. Hiermit können Sie den Speicherort der gewünschten Grafik angeben.

Da sich die Bildgröße in diesem Fall mit CSS nicht verändern lässt, muss die Grafik bereits die korrekten Ausmaße haben. Zu große Bilder können die Lesbarkeit des Listeneintrags herabsetzen oder dafür sorgen, dass der Marker teilweise außerhalb des sichtbaren Bereichs liegt. Wenn Sie eigene Marker benutzen, sollten Sie sicherstellen, dass die Grafik die für das Design Ihrer Webseite richtige Größe hat.

Der Wert für das grafische Aufzählungszeichen wird vererbt. Das bedeutet, dass verschachtelte Einträge das gleiche Bild verwenden wie ihr Elternelement. Um das zu verhindern, muss für das enthaltene Kindelement der Wert `none` angegeben werden.

```

ul {
  list-style-type: disc;
  list-style-image: url('bilder/bullet.gif');
}
ul ul { list-style-image: none; }

```

Für den Fall, dass das Bild aus irgendeinem Grund nicht angezeigt werden kann, sollten Sie mit der Eigenschaft `list-style-type` immer eine Alternative angeben. In unserer Lösung wird der Markertyp `disc` benutzt, falls die Grafik *bullet.gif* nicht angezeigt werden kann.

Die zweite Lösung erscheint im Vergleich zur oben beschriebenen Methode deutlich komplexer. Schließlich wird mit nur zwei Deklarationen scheinbar das Gleiche erreicht.

Das stimmt aber nur teilweise. Da in Lösung 2 das Aufzählungszeichen als Hintergrundbild definiert wurde, stehen sämtliche Eigenschaften für die Gestaltung von Hintergrün-

den zur Verfügung. Hierzu gehört speziell die Eigenschaft `background-position`, mit der Sie die Grafik bei Bedarf pixelgenau positionieren können – eine Möglichkeit, die bei der Verwendung von `list-style-image` fehlt.



Beide Lösungen lassen sich noch weiter verfeinern, indem Sie das Aufzählungszeichen nicht in einer externen Datei speichern, sondern als Daten-URI direkt in der CSS-Datei speichern. Details dazu finden Sie in Kapitel 13, *Tipps und Tricks*.

Siehe auch

Das Rezept 5.3 zum Erstellen eigener textbasierter Aufzählungszeichen; die CSS 2.1-Spezifikation zur Eigenschaft `list-image-type` unter <http://www.w3.org/TR/CSS21/generate.html#propdef-list-style-image>; das Rezept 13.20 zur Verwendung von Daten-URIs; das Grundrezept 2.4 zur Verwendung der `url()`-Funktion.

5.6 Das Aufzählungszeichen innerhalb des Listeneintrags darstellen

Problem

Der Marker soll innerhalb der Box für den Listeneintrag dargestellt werden, wie in Abbildung 5-5 gezeigt. Dadurch umfließt der Text den Marker.

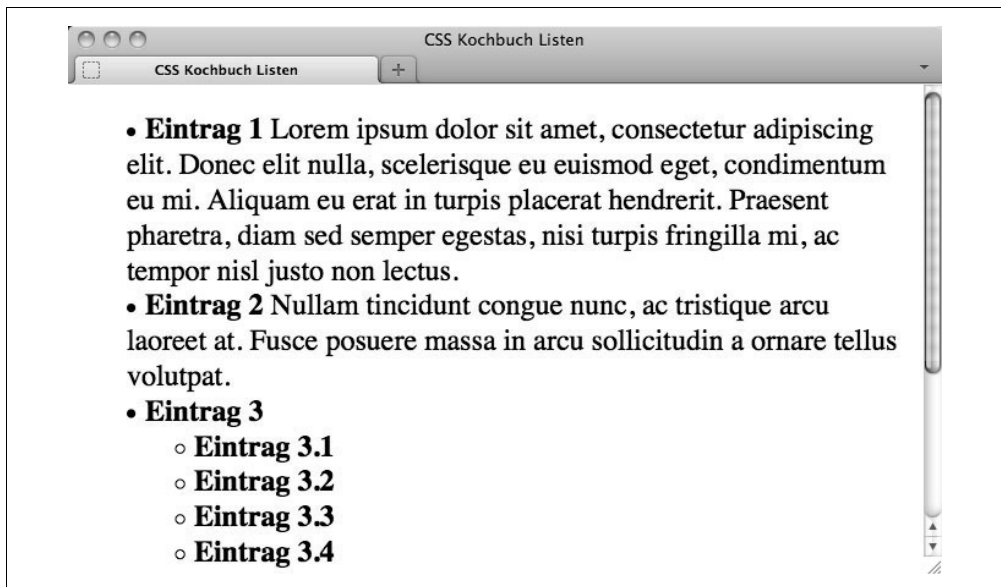


Abbildung 5-5: Den Marker innerhalb des Listeneintrags darstellen

Lösung

Benutzen Sie die Eigenschaft `list-style-position` mit dem Wert `inside`:

```
ul {  
    list-style-position: inside;  
}
```

Durch die erste Deklaration wird das Aufzählungszeichen vom übrigen Inhalt des Listeneintrags umflossen.

Diskussion

Standardmäßig werden die Listenmarker außerhalb der Einträge dargestellt, wodurch das typische Listenlayout zustande kommt. In manchen Designs soll das Aufzählungszeichen aber vielleicht zusammen mit dem Text erscheinen, um beispielsweise auf der linken Seite nicht zu viel Leerraum entstehen zu lassen.

Siehe auch

Die CSS 2.1-Spezifikation zur Eigenschaft `list-style-position` unter <http://www.w3.org/TR/CSS21/generate.html#propdef-list-style-position>; den Artikel zu `list-style-position` im Mozilla Developer Network unter <https://developer.mozilla.org/en/CSS/list-style-position>.

5.7 Die Kurzschrifteigenschaft `list-style` verwenden

Problem

Sie wollen die wichtigsten Werte für die Gestaltung von Listen in einer Deklaration zusammenfassen.

Lösung

Verwenden Sie die Kurzschrifteigenschaft `list-style` mit den entsprechenden Werten. Angenommen, Sie haben folgende Definitionen:

```
ul {  
    list-style-type:    square;  
    list-style-image:   url("bilder/quadrat.gif");  
    list-style-position: outside;  
}
```

Dann können Sie diese Angaben auch wie folgt zusammenfassen:

```
ul {  
    list-style: square url("bilder/quadrat.gif") outside;  
}
```

Beide Regeln sorgen dafür, dass die enthaltenen Listenelemente statt des standardmäßigen Kreises ein Quadrat als Aufzählungszeichen verwenden. Nach Möglichkeit wird hierfür die unter *bilder/quadrat.gif* gespeicherte Grafik verwendet. Außerdem wird ausdrücklich festgelegt, dass der Marker außerhalb der Listeneinträge angezeigt werden soll.

Diskussion

Ähnlich wie andere Kurzschrifteigenschaften, beispielsweise `border` oder `margin`, fasst auch `list-style` die Werte mehrerer Einzeleigenschaften auf einer Zeile zusammen. Namentlich sind das `list-style-type`, `list-style-image` und `list-style-position`.

Auch hier gilt: Nicht angegebene Werte für die Einzeleigenschaften werden auf ihren Standardwert zurückgesetzt. Die Werte können in beliebiger Reihenfolge angegeben werden. Die kürzeste Form, das Aufzählungszeichen zu verbergen, lautet daher:

```
list-style: none;
```

Da das Schlüsselwort `none` nur für die Einzeleigenschaft `list-style-type` eine Bedeutung hat, wird es automatisch korrekt zugewiesen.

Siehe auch

Die CSS 2.1-Spezifikation zur Kurzschrifteigenschaft `list-style` unter <http://www.w3.org/TR/CSS2/generate.html#list-style>.

5.8 Trennlinien für Listeneinträge erstellen

Problem

Die Listeneinträge sollen durch horizontale Linien voneinander getrennt werden, wie in Abbildung 5-6 zu sehen.

Lösung

Schaffen Sie mithilfe der Eigenschaft `padding` etwas Platz zwischen den Listenelementen. Für die Trennlinien verwenden Sie die Eigenschaft `border-bottom`. Mit der Pseudoklasse `:first-child` können Sie die fehlende Linie oberhalb des ersten Listenelements erstellen. Hier die nötigen Regeln:

```
li {  
    padding:      0.5em 0;  
    border-bottom: 1px solid black;  
}  
  
li:first-child {  
    border-top:    1px solid black;  
}
```

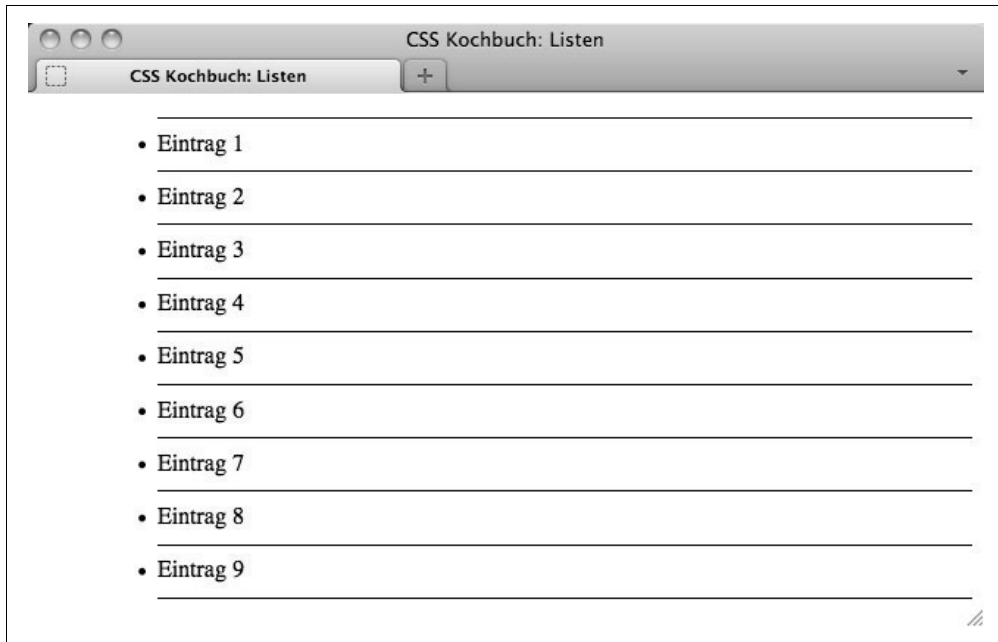


Abbildung 5-6: Die Listeneinträge werden durch eine horizontale Linie visuell voneinander getrennt dargestellt

Diskussion

Neben der häufig verwendeten Kurzschrifteigenschaft `border` stellt CSS auch Eigenschaften bereit, um die Seiten eines Rahmens einzeln zu definieren. Eigentlich sind `border-bottom` und `border-top` ebenfalls Kurzschrifteigenschaften. Mit ihnen können Sie die Werte für Rahmenbreite, -stil und -farbe einer bestimmten Rahmenseite gemeinsam definieren, wie in der Lösung gezeigt.



Der Internet Explorer 7 hat ein kleines Problem mit der Pseudoklasse `:first-child`. Befindet sich vor dem ersten Listeneintrag ein HTML-Kommentar innerhalb der Liste, wird dieser als erstes Element angesehen, wodurch die obere Begrenzungslinie nicht angezeigt wird.

Sollte die Verwendung von `:first-child` einmal gar nicht möglich sein (z.B. weil Sie eine Website für den chinesischen Markt erstellen müssen)¹, können Sie sich damit behelfen, die obere Rahmenseite für das `ul`-Element zu definieren. Hierbei muss allerdings die Einrückung der Liste selbst berücksichtigt werden. Damit das in allen Browsern funktio-

¹ Laut <http://www.ie6countdown.com> hat der Internet Explorer 6 in China immer noch einen Marktanteil von 25%.

niert, sollten Sie eigene Werte für `margin-left` und `padding-left` angeben. Das Rezept aus der Lösung könnte dann beispielsweise so aussehen:

```
li {
    padding:      0.5em 0;
    border-bottom: 1px solid black;
}

ul {
    padding:      0;
    margin:       3em;
    border-top:    1px solid black;
}
```

Siehe auch

Das Rezept 5.1 zur Angabe ausdrücklicher Werte für die Einrückungen von Listen; den Artikel von Robert Nyman zum »first-child-bug« im IE7 unter <http://robertnyman.com/2009/02/04/how-to-solve-first-child-css-bug-in-ie-7/>.

5.9 Listeneinträge nebeneinander darstellen

Problem

Die Elemente einer Liste sollen nebeneinander angeordnet werden (siehe Abbildung 5-7) und nicht, wie sonst üblich, untereinander.

Lösung

Entfernen Sie die Aufzählungszeichen und definieren Sie die Listeneinträge (`li`) als Floats. Damit die Einträge voneinander getrennt erscheinen, benutzen Sie das Pseudoelement `:before`, um für alle Elemente bis auf das erste ein Komma einzufügen:

```
ul, li {
    margin: 0;
    padding: 0;

    list-style: none;
}

li {
    float: left;
}

li:before {
    content: "\002c\0020";
}
```

```
li:first-child:before {
    content: "";
}
```

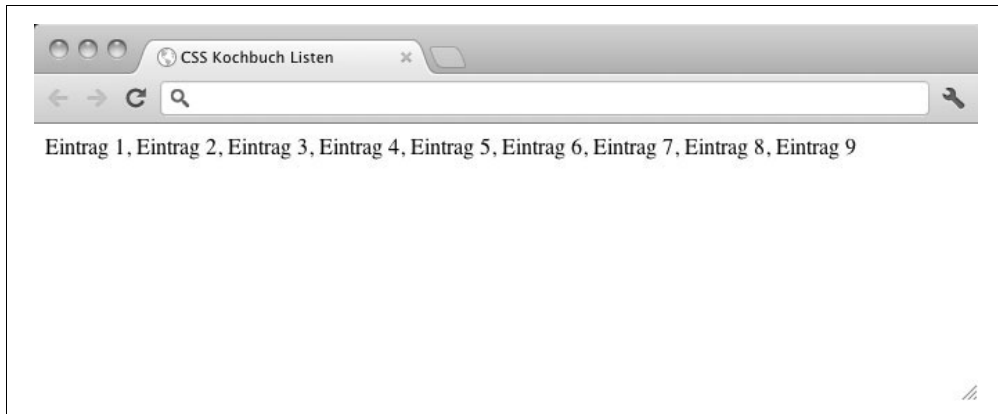


Abbildung 5-7: Durch eine Änderung der Darstellungsrolle werden die Listenelemente nebeneinander angeordnet

Zu Beginn setzen wir die Innen- und Außenabstände sowohl für die Liste selbst wie auch für die Listeneinträge zurück und geben an, dass kein Aufzählungszeichen dargestellt werden soll. Die zweite Regel sorgt dafür, dass die Einträge nebeneinander erscheinen und eventuell dazwischen auftretender Leerraum entfernt wird.

Die dritte Regel fügt vor den Einträgen ein Komma und ein Leerzeichen (jeweils in Unicode-Schreibweise) ein. Um diese Anweisungen für das erste Element wieder rückgängig zu machen, gibt es die vierte Regel.

Diskussion

Dieses Rezept stellt eine wichtige Vorstufe zur Erstellung horizontaler Menüs und Fußbereiche (Footer) dar. Durch die Definition der Listeneinträge als Floats können sie nebeneinander angeordnet werden.

Könnte man das nicht auch dadurch erreichen, dass man die Listeneinträge einfach als Inline-Elemente definierte? Im Prinzip schon, allerdings gibt es hierbei eine Sache zu beachten: Oftmals stehen die einzelnen `li`-Elemente im HTML-Code untereinander. Bei der Änderung der Darstellungsrolle fügen manche Browser für jeden Zeilenumbruch etwas Leerraum zwischen den Einträgen ein.

Daher verwenden wir hier die Eigenschaft `float`, die die Zwischenräume zuverlässig entfernt. Der Grund dafür liegt in der Definition für Floats, nach der sich deren seitliche Außenkanten zwar nicht überschneiden dürfen, aber so dicht beieinanderstehen müssen wie möglich.

Um die Liste wie eine fortlaufende Aufzählung erscheinen zu lassen, fügen wir zwischen den einzelnen Einträgen außerdem ein Komma und ein Leerzeichen ein. Wir benutzen hierfür das Pseudoelement `:before` mit der Eigenschaft `content`.

Als Wert für `content` nehmen wir hier die Unicode-Zeichen `\002c` und `\0020` – ein Komma gefolgt von einem Leerzeichen. Ohne weitere Angaben würde das Komma aber nun auch vor dem ersten Listeneintrag stehen. Um das zu verhindern, verwenden wir eine Kombination aus der Pseudoklasse `:first-child` und dem Pseudoelement `:before`.



Anstelle des Kommas können Sie natürlich auch ein beliebiges anderes Zeichen verwenden. Soll die Liste als horizontales Menü verwendet werden, käme hier beispielsweise auch ein senkrechter Strich (Pipe-Zeichen, `|`) infrage.

Indem wir als Wert für `content` eine leere Zeichenkette (`""`) angeben, wird ausdrücklich festgelegt, dass vor dem ersten Listeneintrag kein Inhalt eingefügt wird.

(Soll auch der Internet Explorer 7 unterstützt werden, ist es am einfachsten, auf die Trennzeichen zu verzichten und die Listeneinträge per `li { padding: 0.2em; }` mit etwas Innenabstand zu versorgen, damit die Elemente nicht direkt »aneinanderkleben«.)

Siehe auch

Das Rezept 1.14 zur Verwendung von Pseudoklassen; das Rezept 1.17 für weitere Informationen zu Darstellungsrollen; die Website <http://www.projectseven.com/csslab/lists/inline-block/horizontal/> für weitere Informationen zur Definition der Listeneinträge als Inline-Block-Elemente.

5.10 Stile für Definitionslisten

Problem

Anstelle der standardmäßigen Browserdarstellung wollen Sie die Teile einer Definitionsliste mit eigenen Stilen versehen, wie in Abbildung 5-8 gezeigt.

Lösung

Verwenden Sie einfache Elementselektoren, um den Definitionsbegriff (»definition term«, `dt`) und die dazugehörige Definition nach Ihren Wünschen zu gestalten, z.B.:

```
dt {  
    font-weight: bold;  
    margin: 0;  
    padding: 0;  
}
```

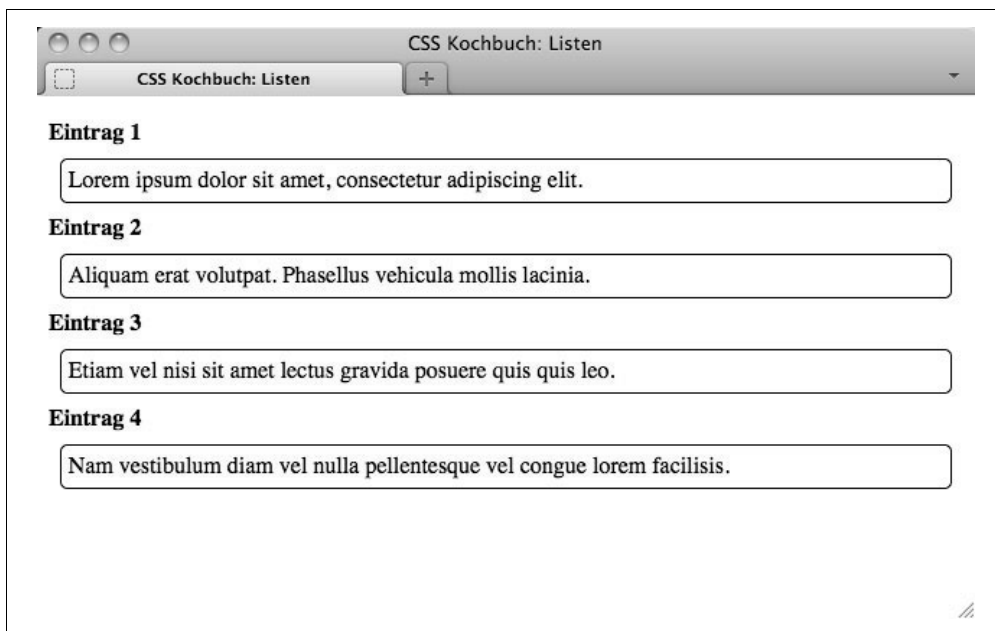


Abbildung 5-8: Eine Definitionsliste mit eigenen Stildefinitionen

```
dd {
  border: 1px solid black;
  margin: 0.5em;
  padding: 0.3em;

  -webkit-border-radius: 5px;
  -moz-border-radius: 5px;
  border-radius: 5px;
}
```

Die erste Regel stellt den Definitionsbegriff fett dar. Zudem werden eventuell vom Browser definierte Innen- oder Außenabstände entfernt.

Die zweite Regel umgibt die Definitionen mit einem 1 Pixel breiten schwarzen Rahmen. Außerdem werden explizite Werte für den Innen- und den Außenabstand vergeben.

Als zusätzliches Bonbon werden die Definitionen in Browsern, die dies bereits verstehen, mit abgerundeten Ecken dargestellt.

Diskussion

Im Gegensatz zu Aufzählungslisten (`ol`) und ungeordneten Listen (`ul`) treten die Einträge einer Definitionsliste üblicherweise als Paar aus Definitionsbegriff (`dt`) und Definition (`dd`) auf. Beide Elemente lassen sich ganz normal im Stylesheet über die entsprechenden Elementselektoren ansprechen und gestalten.

Bei der Gestaltung von Definitionslisten haben Browser oft recht eigene Vorstellungen davon, wie die Darstellung auszusehen hat. So werden die Definitionen z.B. in Safari unverhältnismäßig weit eingerückt.

Durch die Definition eigener Stile können Sie auch Definitionslisten an das Design Ihrer Seiten anpassen.

Siehe auch

Das Rezept 5.1 zur Erzeugung browserübergreifender Einrückungen für Listen; die HTML5-Spezifikation zur Verwendung von Definitionslisten unter <http://www.w3.org/TR/html5/grouping-content.html#the-dl-element>; die HTML 4.01-Spezifikation zu Definitionslisten unter <http://www.w3.org/TR/html401/struct/lists.html#h-10.2>; das Rezept 1.6 zur Verwendung von Elementselektoren.

5.11 Stile für den ersten oder letzten Eintrag einer Liste definieren

Problem

Sie wollen für den ersten bzw. letzten Eintrag einer Liste spezielle Stile definieren.

Lösung

Verwenden Sie die CSS-Pseudoklassen `:first-child` bzw. `:last-child`:

```
li:first-child {  
    border-top: 1px solid black;  
}  
  
li:last-child {  
    margin-bottom: 1em;  
}
```

Hier benutzen wir den Elementselektor für Listeneinträge zusammen mit der Pseudoklasse `:first-child`, um oberhalb des ersten Eintrags einer Liste eine durchgehende schwarze Linie zu erzeugen. Die zweite Regel stützt sich auf den CSS3-Selektor `:last-child`, um unterhalb des letzten Listeneintrags einen 1em breiten Außenabstand einzufügen. Sollen auch Browser unterstützt werden, die `:last-child` noch nicht verstehen, geben Sie dem letzten Element der Liste gegebenenfalls ein passendes `class`-Attribut, das Sie im Stylesheet ansprechen können.

HTML:

```
...  
<li class="last-child">Letzter Eintrag</li>  
</ul>
```

CSS:

```
li.last-child {  
    margin-bottom: 1em;  
}
```

Wir haben hier absichtlich den Namen der Pseudoklasse `last-child` für das `class`-Attribut gewählt, um seine Funktion zu verdeutlichen.

Diskussion

Oft werden für das erste oder das letzte Element einer Liste spezielle Stildefinitionen benötigt, beispielsweise bei der Erstellung von Navigationsmenüs. Hierbei können die Pseudoklassen `:first-child` und `:last-child` sehr hilfreich sein.

Leider versteht der Internet Explorer die Pseudoklasse `:last-child` erst ab Version 9, während Safari, Chrome, Firefox und Opera in allen aktuellen Versionen damit umgehen können. Um auch ältere Versionen des IE zu unterstützen, ist es weiterhin nötig, das letzte Element mit einem eigenen `class`-Attribut zu versehen.

Eine Regel für das letzte Element, die beide Konzepte vereinigt, könnte zum Beispiel so aussehen:

```
li:last-child, li.last-child {  
    margin-bottom: 1em;  
}
```

Eine weitere Möglichkeit ergibt sich, wenn Sie die Anzahl der Listenelemente kennen. In diesem Fall können Sie auch mit Selektoren für benachbarte Geschwisterelemente arbeiten, wie hier am Beispiel einer Liste mit fünf Elementen gezeigt wird:

```
li + li + li + li + li {  
    margin-bottom: 1em;  
}
```

Wenn Ihnen das alles zu mühsam ist, können Sie die Unterstützung für `:last-child` (und viele weitere CSS3-Selektoren) auch mit dem Polyfill *selectivizr* erreichen, das in Kapitel 13, *Tipps und Tricks*, beschrieben wird.

Siehe auch

Die CSS 2.1-Spezifikation zu `:first-child` unter <http://www.w3.org/TR/CSS2/selector.html#first-child>; das Selectors-Modul der CSS3-Spezifikation zur Pseudoklasse `:last-child` unter <http://www.w3.org/TR/css3-selectors/#last-child-pseudo>; Kapitel 6, *Links und Navigation*, zur Gestaltung von Links und Navigationselementen; die Website für die JavaScript-Bibliothek *selectivizr* unter <http://selectivizr.com/>.

Links und Navigation

Einführung

Ohne Links wäre das Web sinnlos. Links ermöglichen es, Informationen über mehrere Websites hinweg zu verfolgen, unabhängig davon, wo auf der Welt die Server für diese Sites stehen.

1996 erstellte Jakob Nielsen, ein Experte für die Benutzerfreundlichkeit des Webs, eine Liste der zehn größten Fehler bei der Gestaltung von Webseiten, die immer wieder auf den neuesten Stand gebracht wird (siehe <http://www.useit.com/alertbox/9605.html>). Einer dieser Fehler ist seiner Meinung nach seit Jahren die Verwendung von nicht standard-konformen Farben für Links. Nielsen empfahl, Links, die bisher nicht genutzt worden waren, prinzipiell blau darzustellen, Lila oder Rot stand für bereits besuchte Links. Diese Überlegungen zielten auf eine größtmögliche Konsistenz ab, während ästhetische Gesichtspunkte weniger eine Rolle spielten.

Durch die Verwendung von CSS schließen sich Konsistenz und Ästhetik aber nicht mehr gegenseitig aus. Mittlerweile können Links, ohne die das Web überhaupt nicht existieren würde, konsistent und gleichzeitig visuell ansprechend dargestellt werden.

In diesem Kapitel zeigen wir Ihnen, wie Sie mehr aus Ihren Links machen können. Hier geht es darum, wie Sie die Links selbst gestalten können, Links in Menüs leichter benutzbar machen, die aktuelle Seite in Navigationsmenüs hervorheben können, und natürlich geht es auch um die Erstellung ansprechender Menüdesigns.

6.1 Die Darstellung der verschiedenen Linkzustände anpassen

Problem

Sie wollen die Darstellung von Links selbst steuern.

Lösung

Verwenden Sie die Pseudoklassen für die Gestaltung von Links wie hier gezeigt:

```
a {
  padding: 0.1em;
  padding-bottom: 0;

  color:          #000;
  text-decoration: none;

  border:         1px solid gray;
  border-left:    0;
  border-right:   0;
}

a:link {
  background-color: #ffb440;
}

a:visited {
  background: #eee;
}

a:hover, a:focus {
  background:    #ff5900;
  outline-color: #ffc873
}

a:active {
  background:    #689ad3;
  border-style: dotted;
}
```

In der ersten Regel legen wir die Stile fest, die für alle Linkzustände gelten sollen: Bis auf die Unterseite erhalten Links etwas Innenabstand. Als allgemeine Textfarbe wählen wir Schwarz. Außerdem entfernen wir die standardmäßige Unterstreichung.

Damit Links trotzdem als solche erkannt werden, versehen wir sie per `border-top` und `border-bottom` mit einer eigenen Über- und Unterstreichung. Die einzelnen Linkzustände stellen wir im Folgenden hauptsächlich über die Hintergrundfarbe dar.

Links, die noch nicht besucht wurden, werden mit einem orangefarbenen Hintergrund dargestellt. Besuchte Links sollen nicht so stark auffallen und werden mit einem hellgrauen Hintergrund versehen.

Befindet sich der Cursor über dem Link (`:hover`) oder wurde er über die Tastatur angesprungen (`:focus`), erhalten die Links einen roten Hintergrund. Da einige Browser Links beim Navigieren mit der Tastatur mit einem speziellen Außenrahmen (`outline`) versehen, passen wir auch dessen Farbe an unser Schema an.

Zum Schluss legen wir noch fest, was passieren soll, wenn der Link aktiviert (angeklickt) wird: Der Hintergrund ist blau, während die Rahmenlinien gepunktet angezeigt werden.

Diskussion

Hyperlinks leisten einen sehr wichtigen Beitrag zur Interaktivität des WWW. Damit diese Interaktivität auch sichtbar wird und zu gebrauchen ist, wird zumindest ein Mechanismus benötigt, Links in ihren verschiedenen Zuständen vom übrigen Text einer Seite zu unterscheiden. In CSS gibt es hierfür die dynamischen Pseudoklassen `:link`, `:visited`, `:hover` und `:active`. Dynamisch sind sie, weil sie direkt in Interaktion mit dem Benutzer wechseln. Je nachdem, ob sich der Cursor gerade über einem Link befindet, ob er angeklickt wird etc., können Sie ihn anhand der Pseudoklassen mit einer eigenen Gestaltung versehen.

In unserer Lösung haben wir uns entschieden, die Schriftfarbe beizubehalten. Die Gestaltung erfolgt stattdessen hauptsächlich durch eine Änderung der Hintergrundfarbe. Links sind Inline-Elemente. Das bedeutet, Sie können im Prinzip alle CSS-Eigenschaften verwenden, die Inline-Elemente »verstehen«. (Sollte Ihnen das nicht ausreichen, können Sie Links auch als Block- oder Inline-Block-Elemente definieren. Dazu mehr in Rezept 6.6.)

Zunächst deklarieren wir die Eigenschaften, die für Links unabhängig von ihrem aktuellen Zustand gelten über einen einfachen Elementselektor (`a`). Es folgen die Regeln für die einzelnen Linkszustände.¹ Hierbei gibt es ein paar Dinge zu beachten: Die Pseudoklassen für Hyperlinks besitzen die gleiche Spezifität (siehe Rezept 1.21). Damit sich die Werte nicht gegenseitig überschreiben, sollten Sie die Selektoren in der folgenden Reihenfolge verwenden: `:link`, `:visited`, `:hover` und `:active`.² Vielleicht hilft Ihnen die im Englischen gern verwendete Eselsbrücke »LoVe/HaTe«.

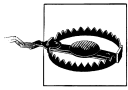
1 Anstelle der Pseudoklasse `:visited` werden die nötigen Stile häufig auch direkt für das `a`-Element definiert (`a {...}`).

2 Eine ausführliche Besprechung dieses Themas finden Sie in Kapitel 3 von Eric A. Meyers *CSS – Das umfassende Handbuch* (O'Reilly 2007).

Besuchte und unbesuchte Links können sowohl den Zustand `:hover` als auch den Zustand `:active` annehmen. Aufgrund der gleichen Spezifität entscheidet die Reihenfolge der Nennung im CSS-Code darüber, welche Regel den Vorrang bekommt.

Wenn Sie sich die Lösung genau ansehen, stellen Sie fest, dass sich hier eine weitere Pseudoklasse eingeschlichen hat: `:focus`. Sie ist zwar hauptsächlich für Formularelemente gedacht, hat aber auch Einfluss auf die Darstellung von Links, wenn die Seite (oder Webapplikation) nicht mit der Maus, sondern über die Tastatur oder ein anderes Eingabegerät bedient wird. Daher verarzten wir sie bei der `:hover`-Regel gleich mit.

Manche Browser zeigen um ein Element herum einen farbigen Außenrahmen an, der mit der Eigenschaft `outline` erzeugt wird. Für den Fall, dass der Browser einen Außenrahmen anzeigt, wird dessen Darstellung über die Eigenschaft `outline-color` an unser Farbschema angepasst.



Mithilfe der JavaScript-Funktion `getComputedStyle()` und anderen Methoden ist es unter Umständen möglich, herauszufinden, welche Websites ein Benutzer besucht hat (History Sniffing). Daher ist es in vielen aktuellen Browsern nicht mehr möglich, der Pseudoklasse `:visited` einen eigenen Stil zuzuweisen. Stattdessen wird meist der Stil für `:link` verwendet.

Steht die Regel für `:hover` vor der für `:active` oder `:visited`, kann es passieren, dass der Benutzer die Definitionen für `:hover` nie zu Gesicht bekommt, weil die folgenden Definitionen die Angaben »maskieren«.

Siehe auch

Das *Selectors*-Modul der CSS3-Spezifikation zu den dynamischen Pseudoklassen `:link`, `:visited`, `:hover`, `:active` und `:focus` unter <http://www.w3.org/TR/selectors/#dynamic-pseudos>; den Artikel von Eric Meyer zur Spezifität der Pseudoklassen für Links unter <http://www.meyerweb.com/eric/css/link-specificity.html>; den Artikel zum »History Sniffing« auf Golem.de unter <http://www.golem.de/1012/79943.html>; eine ausführliche Abhandlung über das »History Sniffing« und die Gegenmaßnahmen in Firefox von David Baron unter <http://dbaron.org/mozilla/visited-privacy>; weitere Informationen zum Thema unter <http://www.impressivewebs.com/styling-visited-links/>; die HTML5-Spezifikation zum `a`-Element unter <http://www.w3.org/TR/html5/text-level-semantic.html#the-a-element>; das Rezept 12.2 zur Erstellung eines eigenen Farbschemas; die Website 24 Ways zur `:focus`-Pseudoklasse unter <http://24ways.org/2009/dont-lose-your-focus>.

6.2 Unterschiedliche Linkfarben für bestimmte Teile einer Seite

Problem

Navigation und Haupttext einer Seite sollen unterschiedliche Stildefinitionen für Links erhalten.

Lösung

Umgeben Sie hierfür die unterschiedlichen Bereiche der Seite mit `div`-Tags, die je nach Funktion unterschiedliche Werte für das `id`-Attribut erhalten:

```
<div id="navigation">
  [...]
</div>

<div id="hauptinhalt">
  [...]
</div>
```

Verwenden Sie die dynamischen Pseudoklassen zusammen mit den ID-Selektoren und der LoVe/HaTe-Methode aus Rezept 6.1, um unterschiedliche Linkstile für die Seitenbereiche zu definieren:

```
#navigation {
  background: #ddd;
  color:      #000;
}

#hauptinhalt {
  background: #555;
  color:      #eee;
}

#navigation a:link { color: blue; }
#navigation a:visited { color: purple; }

#hauptinhalt a:link { color: yellow; }
#hauptinhalt a:visited { color: white; }}
```

Diskussion

Mithilfe von ID- und Klassenselektoren ist es möglich, einzelne Teile einer Webseite zu unterscheiden und mit eigenen Stilen zu versehen, auch wenn diese sich auf die gleichen HTML-Elemente beziehen. Die Verwendung der LoVe/HaTe-Methode stellt sicher, dass Ihre Links erwartungsgemäß funktionieren.

Siehe auch

Das Rezept 1.12 zu ID- und Klassenselektoren; die Sitepoint-Website zur Verwendung von CSS-Pseudoklassen unter <http://reference.sitepoint.com/css/pseudoclasses>; die »Backstage«-Anleitung zur Verwendung mehrerer Linkfarben unter <http://tinyurl.com/multiple-link-colors>.

6.3 Icons am Ende von Links einfügen

Problem

Hinter bestimmten Links soll je nach Funktion (E-Mail-Link, Suche usw.) ein passendes Symbol dargestellt werden, wie in Abbildung 6-1 gezeigt.

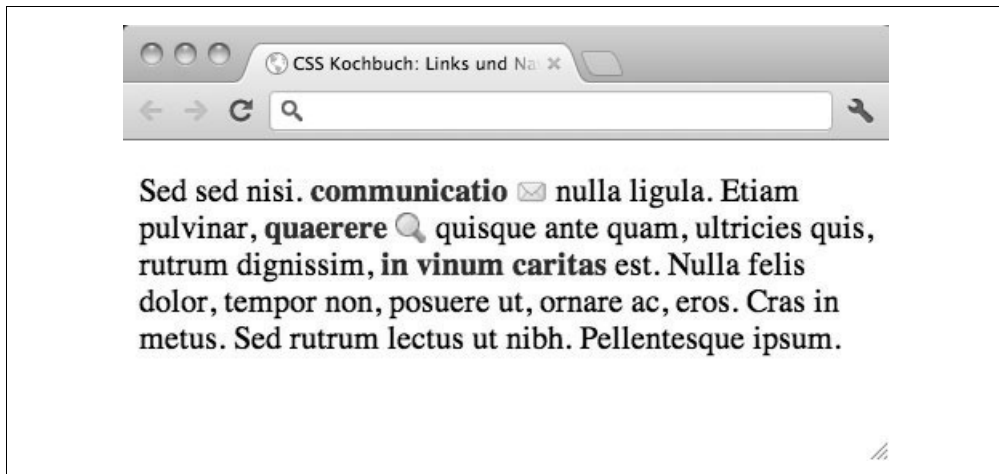


Abbildung 6-1: Die Funktion von Links wird durch nachgestellte Icons verdeutlicht

Lösung

Versehen Sie die Links mit zwei Werten für das Attribut `class`. Der erste Wert, `icon`, besagt, dass dem Link ein Icon nachgestellt werden soll; der zweite Wert (`email` bzw. `suche`) legt fest, welche Art von Icon dargestellt wird. In unserem Beispiel handelt es sich einmal um eine E-Mail-Adresse, das andere Mal um eine Suchmaschine. Danach definieren Sie die Icons als Hintergrundbilder für die Links.

Der HTML-Code könnte beispielsweise so aussehen:

```
...<a href="mailto:bestellungen@csskochbuch.com" class="icon email">communicatio</a> nulla ligula. Etiam pulvinar, <a href="http://www.google.com/" class="icon suche">quaerere</a> quisque...
```

Und hier die nötigen Stildefinitionen:

```
a {
  text-decoration: none;
  font-weight:    bold;
}

a.icon {
  padding-right:    20px;
  background-position: right;
  background-repeat: no-repeat;
}

a.email {
  background-image: url('bilder/links-icons/email.gif');
}

a.suche {
  background-image: url('bilder/links-icons/suche.gif');
}
```

Zunächst wird für alle a-Elemente die Linkunterstreichung entfernt, und es wird festgelegt, dass sie in fetter Schrift dargestellt werden sollen.

Es folgen die Definitionen für alle Links, die mit einem Icon versehen werden sollen. Durch den rechten Außenabstand schaffen wir hinter dem Link ausreichend Platz, um das gewünschte Symbol darzustellen. Außerdem wird festgelegt, dass die Hintergrundbilder rechts positioniert und nicht wiederholt werden sollen.

Die folgenden beiden Regeln legen fest, welches Icon jeweils verwendet werden soll. Da die Werte für Innenabstand, Hintergrundposition und -wiederholung bereits definiert wurden, brauchen wir hier nur noch die URL des benötigten Bilds anzugeben.

Diskussion

Das Konzept dieser Lösung ermöglicht es, das Design ohne großen Aufwand anzupassen und zu erweitern. Wollen Sie Links auf Word-Dokumente speziell hervorheben, bietet sich für das class-Attribut beispielsweise word als zweiter Wert an, wie hier gezeigt:

```
<a href="/berichte/2014.doc" class="icon word">Quartalszahlen 2014 jetzt schon
herunterladen!</a>
```

Danach reicht es auch, das passende Hintergrundbild anzugeben:

```
a.word {
  background-image: url('bilder/links-icons/word.gif');
}
```

Wenn Sie keinen Zugriff auf den HTML-Code der Seite haben, können Sie sich möglicherweise mit CSS3-Attributselektoren behelfen.

```
a[href^="mailto"], a[href$="google.com/"] {
  padding-right:    20px;
  background-position: right;
  background-repeat: no-repeat;
}
```

```

a[href^="mailto"] {
    background-image: url('bilder/links-icons/email.gif');
}

a[href$="google.com/"] {
    background-image: url('bilder/links-icons/suche.gif');
}

```

Hierbei gilt es zu beachten, dass diese Regeln nicht besonders robust sind, da sie vom jeweiligen Inhalt der Seite abhängig sind. Wird die E-Mail-Adresse gegen einen normalen Link ausgetauscht oder die Suchmaschine gewechselt (es reicht schon, den Schrägstrich am Ende der URL wegzulassen), werden möglicherweise keine Icons mehr angezeigt.

Die hier gezeigten Attributselektoren funktionieren im Internet Explorer erst ab Version 9. (Natürlich könnte man die Icons in diesem Fall auch einfach als »Progressive Enhancement« deklarieren, das für die Funktionsweise der Seite nicht unbedingt notwendig ist ...)

Siehe auch

Das Rezept 1.13 zur Verwendung mehrerer Werte im gleichen class-Attribut; den Abschnitt *Neue Attributselektoren in CSS3* in Rezept 1.11; das Rezept 4.9 zur Verwendung von Sprites.

6.4 Tooltips mit CSS gestalten

Problem

Befindet sich der Mauszeiger über einem Link, soll ein kurzer Hilfetext – ein Tooltip – angezeigt werden, der auch Grafiken und Links enthalten soll (siehe Abbildung 6-2).

Lösung

Fügen Sie direkt nach dem Link ein span-Element ein, dessen class-Attribut den Wert `hilfetext` erhält. Zwischen die span-Tags schreiben Sie nun den Text, der angezeigt werden soll, wenn sich der Mauszeiger über dem Link befindet.

Abschließend umgeben Sie Link und Hilfetext mit einem weiteren span-Element (`tooltip`), das den semantischen Zusammenhang zwischen beiden Elementen herstellt und den Positionierungskontext für den Hilfetext bereitstellt. Der HTML-Code könnte etwa so aussehen:

```

<span class="tooltip">
  <a href="#">HTML-Seiten</a>
  <span class="hilfetext">Weitere <a href="/info">Informationen</a></span>
</span>

```

Die beiden span-Elemente werden Sie nun wie folgt mit Stilen versehen:

```

.tooltip {
    position: relative;
    background: rgb(255,218,185);
}

```

```

.tooltip .hilfetext {
    display: none;
}

.tooltip .hilfetext:before {
    content: url('bilder/gluehbirne.png');
    float: right;
}

.tooltip:hover .hilfetext {
    display: block;
    position: absolute;
    z-index: 10;

    top: 1.3em;
    left: 2em;
    width: 17em;
    padding: 0.5em 0.7em;

    font-family: Helvetica, Arial, sans-serif;
    font-size: smaller;
    color: #eee;

    background: rgb(50,50,50);
    background: rgba(30,30,30,0.8);

    border: 1px solid rgb(50,50,50);

    -webkit-border-radius: 5px;
    -moz-border-radius: 5px;
    border-radius: 5px;

    -webkit-box-shadow: 3px 3px 3px rgba(20,20,20,0.6);
    -moz-box-shadow: 3px 3px 3px rgba(20,20,20,0.6);
    box-shadow: 3px 3px 3px rgba(20,20,20,0.6);
}

.hilfetext a:link { color: darkorange; }
.hilfetext a:visited { color: red; }
.hilfetext a:hover { color: gold; }
.hilfetext a:active { color: orangered; }

```

Zu Beginn des Stylesheets definieren wir das äußere span-Element (tooltip) als Positionierungskontext für den Hilfetext. Dadurch erscheint der Tooltip immer in der Nähe des Links, unabhängig davon, wo auf der Seite er schließlich erscheint. Zudem erhält das äußere span-Element eine eigene Hintergrundfarbe.

Die zweite Regel verwendet einen einfachen Nachfahrenselektor und die Eigenschaft `display`, um das innere span-Element zu verbergen, bis der Mauszeiger über den Link bewegt wird.

Um den Hilfetext besser hervorzuheben, soll rechts neben dem Hilfetext ein Glühbirnen-symbol angezeigt werden. Dieses fügen wir mit dem Pseudoelement `:before` in die Darstellung ein und verschieben es dann per `float: right` nach rechts.

Die folgende Regel ist etwas komplexer als der Rest. Wir verwenden hier eine Kombination aus Klassenselektoren, Nachfahrenselektor und der Pseudoklasse `:hover`. Der Selektor wählt `span`-Elemente mit der Klasse `hilfetext` aus. Die Anweisungen werden aktiviert, sobald sich der Mauszeiger über dessen Vorfahrenelement mit der Klasse `tooltip` befindet.

Zunächst ändern wir die Darstellungsrolle von `display: none` in `display: block`. Danach können wir den Tooltip absolut positionieren und per `z-index: 10` vor den übrigen Inhalten platzieren.

Die folgenden vier Deklarationen (`top`, `left`, `width` und `padding`) setzen den Hilfetext unterhalb des äußeren `span`-Elements und verschieben ihn ein wenig nach links. Außerdem beschränken wir die Breite auf 17 em-Einheiten und schaffen per `padding` um den Text etwas Platz.

Der dritte Abschnitt legt fest, welche Schrift für den Hilfetext verwendet werden soll. Er sorgt außerdem dafür, dass die Schrift kleiner als das umgebende `span`-Element erscheint und die Farbe einen ausreichenden Kontrast zum Hintergrund bekommt, den wir im nächsten Schritt definieren.

Der Hintergrund wird in zwei Schritten festgelegt. Browser, die bereits die `rgba()`-Schreibweise verstehen, versehen den Hilfetext mit einem dunkelgrauen, leicht transparenten Hintergrund. Ältere Versionen ignorieren diese Anweisung und verwenden stattdessen den `rgb()`-Wert.

Zum Schluss der Regel noch etwas Kosmetik: Der Hilfetext wird mit einem schlichten Rahmen versehen. Moderne Browser runden zudem die Kanten ein wenig ab und erzeugen etwas räumliche Tiefe, indem sie hinter dem Tooltip einen leichten Schattenwurf erzeugen.

Das Stylesheet endet mit einer Reihe von Farbdefinitionen für Links, die möglicherweise im Tooltip enthalten sind.

Diskussion

Das HTML-Attribut `title` bietet eigentlich eine einfache Möglichkeit, Links und andere Elemente mit erläuternden Texten zu versehen. Leider gibt es hierfür kaum Gestaltungsoptionen, und Sie sind in den meisten Fällen auf die Standarddarstellung des Browsers oder JavaScript-Lösungen angewiesen. Ähnliches gilt für die Verwendung der Eigenschaft `content`, mit der es über die Funktion `attr()` möglich ist, den Inhalt des `title`-Attributs in erzeugten Inhalten zu nutzen.

Die hier gezeigte Lösung bietet dagegen umfangreiche Gestaltungsmöglichkeiten. Der Preis sind zwei ineinander verschachtelte `span`-Tags.

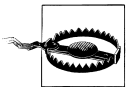


Abbildung 6-2: Der fertige Tooltip

Da der Hilfetext selbst ebenfalls einen Link enthalten soll, müssen wir etwas tiefer in die Trickkiste greifen. Laut HTML-Spezifikation ist es verboten, Links ineinander zu verschachteln. Das End-Tag des inneren `a`-Elements (*Informationen*) würde auf den äußeren Link (*HTML-Seiten*) bezogen.

Ansonsten würde es ausreichen, das `span`-Element mit dem Tooltip einfach innerhalb des Links zu definieren. Stattdessen schreiben wir beide Teile hintereinander und umgeben sie mit einem zusätzlichen `span`-Tag.

Tooltips können für fast alle Elemente einer Webseite definiert werden, z.B. um diese leichter zugänglich zu machen oder ihren Inhalt näher zu erläutern. Probieren Sie diese Technik an Tabellenzellen, Eingabefeldern von Formularen und anhand von Links aus.



Noch ein kurzes Wort zu den Hintergrundfarben: Der Internet Explorer 7 akzeptiert Farbangaben in der `rgb()`-Schreibweise nur, wenn diese für die Kurzschrifteigenschaft `background` definiert werden. Wollen Sie stattdessen die Eigenschaft `background-color` verwenden, müssen Sie z.B. auf hexadezimale Werte zurückgreifen.

Siehe auch

Das Grundrezept 2.19 zum Positionierungskontext; das Rezept 1.17 zur Eigenschaft `display`; die HTML5-Spezifikation zur Verschachtelung von Links unter <http://www.w3.org/TR/html5/text-level-semantic.html#the-a-element>; das Rezept 1.14 zum Pseudoelement

:before und der Pseudoklasse :hover; das Grundrezept 2.21 zur Eigenschaft z-index; die Website *CSS Tricks* zum rgb-Problem in IE7 unter <http://css-tricks.com/ie-background-rgb-bug/>.

6.5 Dokumentteile nur bei Bedarf anzeigen

Problem

Sie möchten ein Element nur dann anzeigen, wenn zuvor ein *Öffnen*-Link angeklickt wurde. Klickt der Benutzer auf den entsprechenden *Schließen*-Link, soll das Element wieder verborgen werden, wie in Abbildung 6-3 gezeigt.



Abbildung 6-3: Die Linkliste wird nach einem Klick auf den Öffnen-Link angezeigt

Lösung

Verwenden Sie die CSS3-Pseudoklassen :target und :not, um das gewünschte Element bei Bedarf ein- und auszublenden. Für diese Lösung verwenden wir den folgenden HTML-Code:

```
<ul>
  <li><a href="#links">Öffnen</a></li>
  <li><a href="#zu">Schließen</a></li>
</ul>

<ul id="links">
  <li><a href="http://www.ora.com/">O'Reilly</a></li>
  <li><a href="http://www.slashdot.org/">Slashdot</a></li>
  <li><a href="http://www.apple.com/">Apple</a></li>
  <li><a href="http://www.microsoft.com/">Microsoft</a></li>
  <li><a href="http://www.mozilla.org/">Mozilla</a></li>
</ul>
```

Das dazugehörige Stylesheet sieht so aus:

```
#links          { display: none; }
#links:target    { display: block;}
#links:not(:target) { display: none; }
```

Die erste Regel sorgt dafür, dass die Liste mit den Links beim Laden der Seite nicht angezeigt wird. Die zweite zeigt die Liste an, sobald der entsprechende Link (*Öffnen*) angeklickt wird. Bei einem Klick auf den *Schließen*-Link sorgt die dritte Regel dafür, dass die Liste wieder verschwindet.

Diskussion

In diesem Rezept verwenden wir die CSS3-Pseudoklassen `:target` und `:not`, um ein Element per Klick auf einen Links anzuzeigen und wieder verschwinden zu lassen.

Im Stylesheet wird per `display: none` festgelegt, dass die Linkliste beim Laden der Seite nicht dargestellt wird. Um sie bei Bedarf anzuzeigen, machen wir uns die Funktionsweise von `:target` zunutze. Hiermit können wir Elemente mit Stildefinitionen versehen, die über einen sogenannten *Fragment-Identifizier* als Linkziel innerhalb des gleichen Dokuments gekennzeichnet sind.

Hierfür sind zwei Dinge nötig: ein Element mit einem `id`-Attribut und ein Link, der auf dieses Element verweist. In unserem Beispiel ist das der *Öffnen*-Link:

```
<a href="#links">Öffnen</a>
```

Er verweist auf das Element mit der ID `links` im gleichen Dokument. Mithilfe der Pseudoklasse `:target` können wir dieses Element nun mit eigenen Stilen versehen. Dadurch ist es möglich, die Darstellungsrolle der Liste von `display: none` in `display: block` zu ändern, wenn der *Öffnen*-Link angeklickt wird.

Um die Liste wieder zu verbergen, bedienen wir uns einer weiteren CSS3-Pseudoklasse mit dem Namen `:not`. Mit ihr können Sie einen Selektor *negieren*. Der Selektor `#links:not(:target)` wählt demnach das Element mit der ID `links` (die Linkliste) aus, wenn es nicht das Ziel des gerade angeklickten Links ist.

Klickt der Benutzer den *Schließen*-Link an (das auf das fiktive Linkziel `#zu` verweist), greift dieser Selektor, und die Eigenschaft `display` für die Linkliste erhält wieder den Wert `none`.

Puristen könnten an dieser Stelle bemängeln, dass durch dieses Vorgehen die Trennung von Darstellung und Verhalten aufgehoben wird. Auf der anderen Seite funktioniert diese Lösung auch ohne JavaScript.

Neben einfachen Anwendungen, wie dem Hervorheben des Linkziels, öffnet `:target` außerdem die Tür zu deutlich ausgefeilteren Anwendungen. Das reicht von »Accordion«-Effekten bis hin zu kompletten Seitenlayouts, bei denen die einzelnen Seiten per `:target` ein- und ausgeblendet werden. Im Abschnitt »Siehe auch« finden Sie eine kleine Auswahl.

Die Pseudoklasse `:target` funktioniert in allen aktuellen Browsern außer Internet Explorer 7 und 8. Bei Bedarf lässt sich die Unterstützung aber mit dem Polyfill *selectivizr* nachrüsten.

Siehe auch

Das Rezept 1.14 unter anderem zur Pseudoklasse `:target`; das Rezept 1.11 zur Gruppierung von Attributselektoren; das *Selectors*-Modul der CSS3-Spezifikation zu den Pseudoklassen `:target` und `:not` unter <http://www.w3.org/TR/selectors/#target-pseudo> bzw. <http://www.w3.org/TR/selectors/#negation>; die Website *The CSS Ninja* zur Realisierung eines Accordion-Effekts per `:target` unter <http://www.thecssninja.com/css/accordion-effect-using-css>; die Opera-Entwickler-Website zur Gestaltung von Benutzerschnittstellen mit `:target` unter <http://dev.opera.com/articles/view/css3-target-based-interfaces/>; den Artikel zur »Yellow Fade Technique« unter <http://37signals.com/svn/archives/000558.php>; das Rezept 13.18 zur Verwendung von Polyfills; einen Artikel von Jens Meiert mit Anwendungsbeispielen für `:target` unter <http://meiert.com/de/publications/articles/20060119/>.

6.6 Den anklickbaren Bereich eines Links vergrößern

Problem

Um die Benutzerfreundlichkeit Ihrer Seiten zu erhöhen, soll der gesamte einen Link umgebende Bereich (z.B. ein `li`-Element) anklickbar sein, wie in Abbildung 6-4 gezeigt.



Abbildung 6-4: Der anklickbare Bereich erstreckt sich über das gesamte Listenelement

Lösung

Definieren Sie die Links als Blockelemente und passen Sie gegebenenfalls die Innenabstände des umgebenden Elements an. Für diese Lösung gehen wir davon aus, dass sich der Link in einem einfachen Listeneintrag befindet, etwa so:

```
<li><a href="http://www.w3.org/Style/CSS/" title="Die offizielle Homepage des World Wide Web Consortium für Cascading Style Sheets">W3C CSS-Seite</a></li>
```

Ohne weitere Angaben wäre der Link nur anklickbar, wenn sich der Mauszeiger direkt darüber befindet. Das lässt sich aber leicht durch die folgenden Regeln beheben:

```
li { padding: 0; }

li a {
    display: block;
    padding: 0.2em;
}
```

Die erste Regel sorgt dafür, dass sämtliche Innenabstände für das `li`-Element auf null gesetzt werden. Dadurch erstreckt sich der Link über die gesamte Fläche des umgebenden Elements.

Der zweite Schritt definiert den Link als Blockelement. Dadurch wird der anklickbare Bereich auf den gesamten umgebenden Bereich (hier das Listenelement) ausgedehnt. Den für das Listenelement entfernten Innenabstand können Sie nun für den Link selbst definieren.

Diskussion

Kennen Sie das? Sie wollen einen Link in einem Navigationsmenü anklicken, aber die Seite reagiert nicht auf Ihre Eingaben. Das liegt oft daran, dass der Mauszeiger sich nicht genau über dem Link befindet, sondern etwas außerhalb.

Mit der hier gezeigten einfachen Lösung können Sie die Benutzerfreundlichkeit Ihrer Seiten deutlich erhöhen, indem der anklickbare Bereich auf den gesamten umgebenden Block ausgedehnt wird.



Unter Umständen ist es sinnvoll, für Links, die als Blockelemente definiert wurden, auch die Eigenschaften `width` und `height` anzugeben.

Das kann auch aus Gründen der Zugänglichkeit sinnvoll sein. Besucher mit motorischen Einschränkungen oder Sehproblemen werden Ihre Seiten leichter bedienen können.

Siehe auch

Das Rezept 1.17 zur Eigenschaft `display`; die Rezepte zur Gestaltung von Navigationsmenüs in diesem Kapitel.

6.7 Listen als Navigationsmenüs gestalten

Problem

Sie haben eine ungeordnete Liste mit Links, die als vertikales Navigationsmenü dargestellt werden soll.

Lösung

Entfernen Sie die Aufzählungszeichen und passen Sie Einrückungen, Hintergründe und Rahmen gemäß Ihren Vorstellungen an. Angenommen, Sie haben folgende Liste mit Links:

```
<ul id="navigation">
  <li><a href="/startseite">Startseite</a></li>
  <li><a href="/produkte">Produkte</a></li>
  <li><a href="/ueber-uns">Über uns</a></li>
  <li><a href="/kontakt">Kontakt</a></li>
  <li><a href="/impressum">Impressum</a></li>
</ul>
```

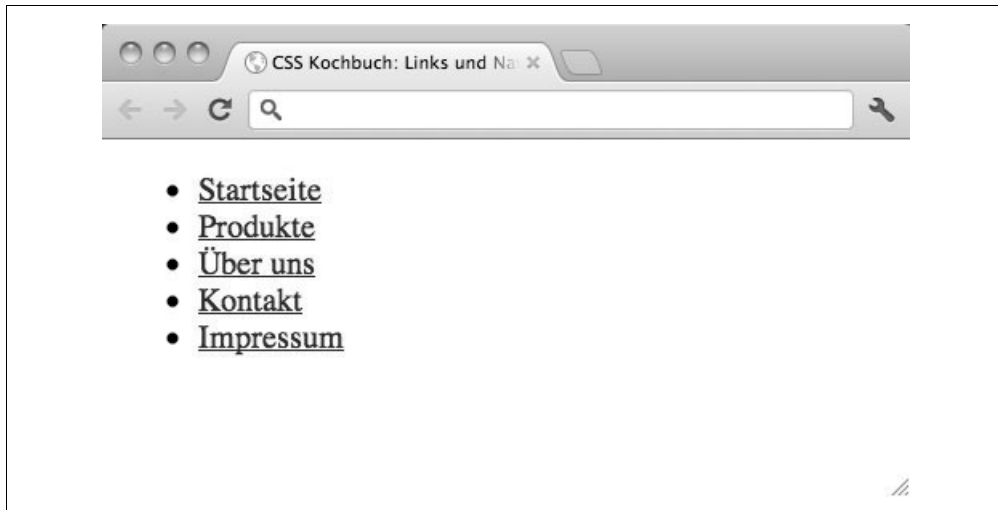


Abbildung 6-5: Eine einfache Liste mit Links

Ohne Stildefinitionen würde die Liste mit den Standardstilen des Webbrowsers dargestellt, wie Abbildung 6-5 zeigt. Mit den folgenden CSS-Regeln können Sie die Darstellung so anpassen, dass aus der schlichten Liste ein »echtes« Navigationsmenü entsteht:

```
#navigation, #navigation li {
  margin: 0;
  padding: 0;
}
```

```

#navigation {
  margin: 1em;
  list-style: none;
  min-width: 15em;

  font-family: Helvetica, Arial, sans-serif;
  color: rgb(10,10,10);
}

#navigation li {
  background: rgb(200,200,200);
  background: rgba(200,200,200,0.6);

  border-bottom: 1px solid black;
}

#navigation li:first-child {
  border-top: 1px solid black;
}

#navigation li a:link,
#navigation li a:visited {
  padding: 0.2em;
  display: block;

  border-left: 5px solid black;
  border-right: 5px solid black;

  text-decoration: none;
  color: inherit;
}

#navigation li a:hover {
  background-color: #f4a460;

  border-left: 5px solid rgb(55,91,126);
  border-right: 5px solid rgb(55,91,126);
}

```

Damit die Regeln nicht auf alle Listen angewendet werden, haben wir die Navigationsliste mit dem id-Attribut `navigation` gekennzeichnet, das wir den einzelnen Selektoren im Stylesheet voranstellen.

Die erste Regel setzt die Innen- und Außenabstände für die Liste (`ul`) und alle enthaltenen `li`-Elemente auf null. Dadurch sind wir von der Standarddarstellung der Browser unabhängig und können zuverlässig unsere eigenen Werte angeben.

Das passiert auch gleich. In der zweiten Regel wird festgelegt, dass die Liste selbst (`ul`) einen Außenabstand von einer em-Einheit erhalten soll. Außerdem entfernen wir hier die Aufzählungszeichen für die `li`-Elemente. Das funktioniert, weil der Wert für `list-style` an die enthaltenen Elemente vererbt wird.

Die Angabe `min-width: 15em` legt fest, dass die Liste mindestens 15em breit sein soll. Hierdurch passt sich die Breite des Menüs dynamisch an den vorhandenen Platz an, ohne zu schmal zu werden. Außerdem legen wir hier die Schriftfamilie und die Textfarbe fest. Während der Wert von `font-family` automatisch an Nachfahren weitergegeben wird, müssen wir den Wert von `color` in den gewünschten Elementen per `inherit` ausdrücklich laden.

Es folgen die Deklarationen für die einzelnen `li`-Elemente. Zunächst legen wir fest, dass die Listeneinträge mit einem hellgrauen Hintergrund hinterlegt werden sollen. Browser, die das verstehen, werden die zweite `background`-Definition verwenden. Sie sorgt dafür, dass der hellgraue Hintergrund teilweise transparent wird. Dadurch scheint der Hintergrund der Seite ein wenig durch, und das Menü passt sich besser an das Farbschema der Seite an.

Um die einzelnen Einträge besser voneinander unterscheiden zu können, definieren wir außerdem per `border-bottom` eine dünne horizontale Trennlinie.

Damit das Ganze geschlossen aussieht, muss außerdem oberhalb des ersten Listeneintrags eine Linie dargestellt werden. Mithilfe der Pseudoklasse `:first-child` wählen wir das erste `li`-Element aus und übergeben per `border-top` die nötigen Werte.

Es folgen die Stile für die enthaltenen Links. Zuerst dehnen wir, wie in Rezept 6.6 beschrieben, den anklickbaren Bereich der Links auf den gesamten umgebenden Block aus, indem wir den Link selbst als Blockelement (`display: block`) definieren. Damit die Links nicht direkt am Rahmen der Listenelemente kleben, wird mit der Kurzschrifteigenschaft `padding` um die Links etwas Platz geschaffen.

Um die Navigationsliste visuell etwas stärker hervorzuheben, soll links und rechts der Links eine 5 Pixel dicke Linie angezeigt werden. Diese soll zudem ihre Farbe wechseln, wenn der Mauszeiger sich über dem entsprechenden Eintrag befindet. Die Eigenschaften `border-left` und `border-right` sind hier genau richtig. Außerdem wird per `text-decoration: none` die für Links übliche Unterstreichung entfernt.

Damit Links, über denen sich der Mauszeiger gerade befindet, trotzdem angemessen hervorgehoben werden, erhalten sie in der folgenden Regel (`li a:hover`) eine eigene Hintergrundfarbe (`#f4a460` für die benannte Farbe »sandybrown«). Außerdem ändern wir beim Überfahren mit der Maus die Farbe des seitlichen Rahmens.

Das fertige Navigationsmenü sehen Sie in Abbildung 6-6.

Diskussion

Eine ungeordnete Liste bietet die perfekte Struktur für ein Linkmenü – sowohl theoretisch wie auch in der praktischen Anwendung. Immerhin ist ja eine Linkliste eine Liste ungeordneter Dinge. Die Verwendung von ungeordneten Listen für die Navigation ergibt demnach eine solide Struktur, die sich sowohl logisch wie auch semantisch korrekt in die Dokumentstruktur einfügt.

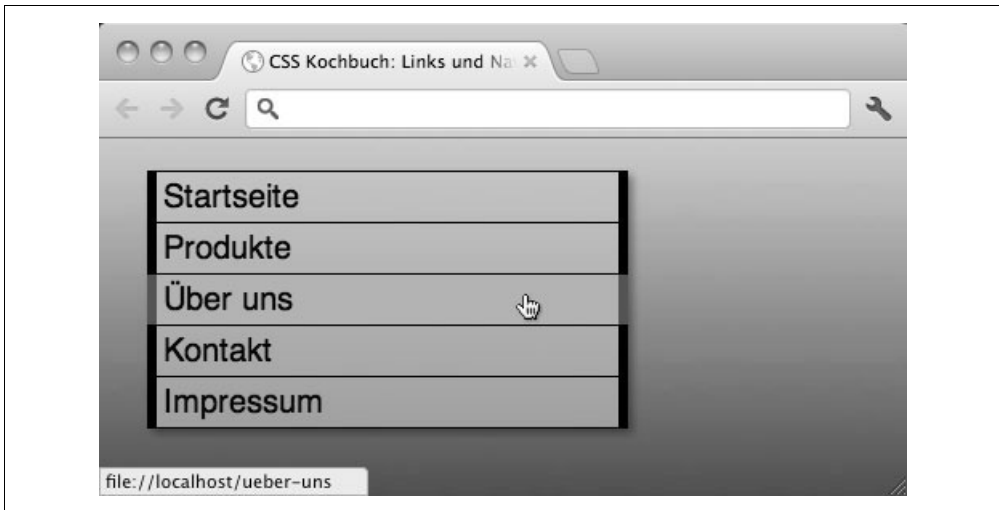


Abbildung 6-6: Die Linkliste wird als Navigationsmenü dargestellt

Gleichzeitig erleichtert die Organisation der Links in einer ungeordneten Liste die Darstellung in Menüform. Bei der Verwendung von mehreren `div`-Elementen wäre das nicht so einfach.

Wenn Sie die standardmäßigen Unterstreichungen der Links entfernen, müssen Sie auf jeden Fall dafür sorgen, dass die Links trotzdem als solche erkennbar sind. In der hier gezeigten Lösung geschieht das, indem sich die Hintergrundfarbe der Links (die sich als Blockelement über den gesamten zur Verfügung stehenden Bereich ausdehnen) sowie die Farbe der seitlichen Rahmenkanten ändern, sobald sich der Mauszeiger darüber befindet.

Navigationslisten mit CSS3-Eigenschaften aufwerten

Wenn Sie sich Abbildung 6-6 ansehen, wird Ihnen auffallen, dass die Liste mit einem dezenten Schlagschatten versehen wurde. Hierfür erweitern Sie die Regel für die gesamte Liste (`ul`) folgendermaßen:

```
ul.navigation {
    margin: 1em;
    list-style: none;

    width: 33%;
    min-width: 15em;

    -webkit-box-shadow: 3px 3px 5px rgba(10,10,10,0.4);
    -moz-box-shadow: 3px 3px 5px rgba(10,10,10,0.4);
    box-shadow: 3px 3px 5px rgba(10,10,10,0.4);
}
```

Browser, die die Eigenschaft `box-shadow` bereits verstehen (in der offiziellen oder der Präfixschreibweise), erzeugen nun den gewünschten Schattenwurf.

Wenn Sie noch einen Schritt weitergehen wollen, können Sie außerdem die Änderung der Hintergrundfarbe für `li a:hover` mit einem sanften Übergang (transition) versehen. Erweitern Sie hierfür die Regel für noch nicht besuchte Links (und besuchte Links) um folgende Deklarationen:

```
ul.navigation li a:link,
ul.navigation li a:visited {
    display: block;
    padding: 0.2em;

    border-left: 5px solid black;
    border-right: 5px solid black;

    color: inherit;
    text-decoration: none;

    -webkit-transition: all 0.2s ease-in-out;
    -moz-transition: all 0.2s ease-in-out;
    -o-transition: all 0.2s ease-in-out;
    transition: all 0.2s ease-in-out;
}
```

Anstatt die Farbe einfach »umzuschalten«, gibt es nun einen fließenden Wechsel der Hintergrundfarbe beim Überfahren eines Links mit der Maus.



Die Verwendung der CSS3-Eigenschaften für Schattenwürfe und Übergänge wird noch nicht von allen Browsern unterstützt. Sie sollten diese Methoden daher nur im Rahmen des »Progressive Enhancement« einsetzen, also an Stellen, die für die Darstellung Ihrer Inhalte nicht absolut notwendig sind.

Siehe auch

Das Rezept 12.1 zur Erstellung von Rollover-Effekten; den Artikel *CSS Design: Taming Lists* von Mark Newhouse, zu finden unter <http://www.alistapart.com/articles/taming-lists/>; das Rezept 12.7 zur Erzeugung von Schattenwürfen mit CSS; das Rezept 12.10 zur Definition von animierten Übergängen (Transitions) zwischen zwei Werten für die gleiche Eigenschaft; das Grundrezept 2.7 zur Gestaltung von Rahmen; das Rezept 6.6 zur Vergrößerung des anklickbaren Bereichs; das Rezept 10.10 zur Gestaltung von Links als Buttons in Mobilgeräten.

6.8 Die aktuelle Seite hervorheben

Problem

Ein mit Rezept 6.7 erzeugtes Navigationsmenü soll so verändert werden, dass die gegenwärtig sichtbare Seite im Menü hervorgehoben wird.

Lösung

Versehen Sie das body-Element der Seite und die einzelnen Listeneinträge mit id-Attributen und entfernen Sie die a-Tags für den Menüeintrag der aktuellen Seite:

```
<body id="produkte">

<ul id="navigation">
  <li id="li-startseite"><a href="/startseite">Startseite</a></li>
  <li id="li-produkte">Produkte</li>
  <li id="li-ueber-uns"><a href="/ueber-uns">Über uns</a></li>
  <li id="li-kontakt"><a href="/kontakt">Kontakt</a></li>
  <li id="li-impressum"><a href="/impressum">Impressum</a></li>
</ul>

</body>
```

Danach können Sie durch die Kombination beider id-Selektoren spezielle Stile für den Menüeintrag der gerade angezeigten Seite definieren, wie hier am Beispiel der *Produkte*-Seite gezeigt:

```
#produkte #li-produkte {
  background-color: #d2691e;

  border-left: 5px solid rgb(55,120,126);
  border-right: 5px solid rgb(55,120,126);
}
```

Durch die Kombination zweier id-Selektoren können Sie die Stile für den Menüeintrag der aktuellen Seite definieren. Da dieser keinen Link enthält, definieren wir die linke und rechte Rahmenseite sowie die Hintergrundfarbe in diesem Fall direkt für das li-Element. Das Ergebnis sehen Sie in Abbildung 6-7.

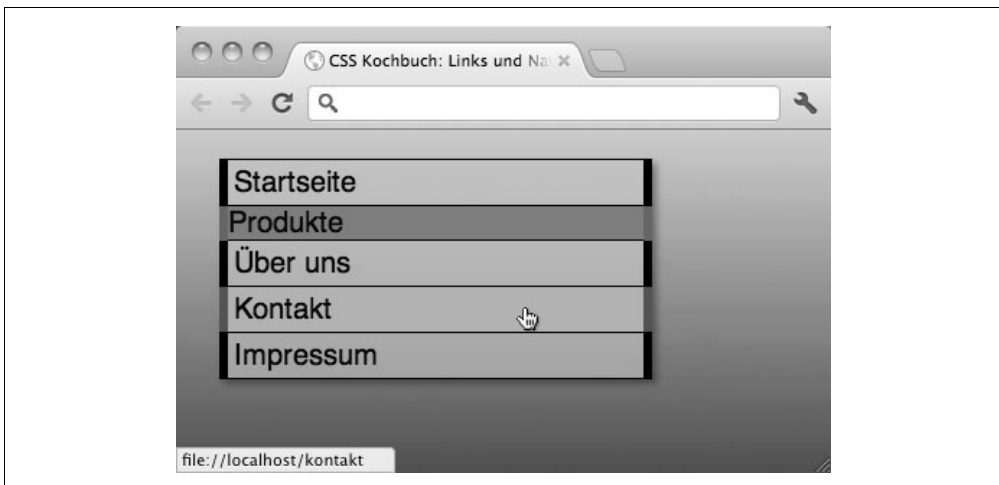


Abbildung 6-7: Die Navigationsliste wurde so verändert, dass die aktuelle Seite hervorgehoben wird

Diskussion

Um den Besuchern Ihrer Seiten die Orientierung zu erleichtern, kann es sinnvoll sein, die aktuelle Seite im Navigationsmenü hervorzuheben. Dafür muss das Stylesheet aber »wissen«, auf welcher Seite sich der Besucher gerade befindet.

Das können Sie erreichen, indem jede Seite im `body`-Tag mit einem `id`-Attribut versehen wird, das diese Seite von allen anderen unterscheidet.

Das gleiche Problem tritt auf, wenn entschieden werden muss, welcher Listeneintrag speziell gestaltet werden soll. Auch hier hilft die Verwendung von `id`-Attributen weiter. Indem jeder Eintrag seine eigene »Identität« erhält, können wir ihn im Stylesheet nach Bedarf gestalten.

Vermutlich wollen Sie die obige Regel nicht für jede Seite komplett neu schreiben. Sie können sich die Arbeit erleichtern, indem Sie die Selektoren, wie in Rezept 1.4 beschrieben, gruppieren. So brauchen Sie die Regel nur einmal anzugeben:

```
#startseite #li-startseite,  
#produkte #li-produkte,  
#ueber-uns #li-ueber-uns,  
#kontakt #li-kontakt,  
#impressum #li-impressum {  
    background-color: #d2691e;  
  
    border-left: 5px solid rgb(55,120,126);  
    border-right: 5px solid rgb(55,120,126);  
}
```

Beachten Sie: Nach dem letzten Selektor (hier `#impressum #li-impressum`) darf kein Komma stehen, ansonsten ist die Regel ungültig und wird vom Browser ignoriert.

Die CSS3-Pseudoklasse `:empty` funktioniert hier leider nicht. Ansonsten hätten Sie sich die Zuweisung der `id`-Attribute komplett sparen können. Leider ist `:empty` aber so definiert, dass das betreffende Element (z.B. `li:empty`) weder Kindelement, noch anonymen Text enthalten darf, also vollkommen leer sein muss.

Siehe auch

Das *Selectors*-Modul der CSS3-Spezifikation unter <http://www.w3.org/TR/selectors/> zur Benutzung von `class` und `id`; das Rezept 1.20 zur Spezifität; das Rezept 6.7 zur Gestaltung von Navigationslisten.

6.9 Eine Breadcrumb-Navigation gestalten

Problem

Anhand einer »Brotkrümelspur« (Bread Crumb Trail) soll dem Besucher verdeutlicht werden, an welcher Stelle in der Seitenhierarchie Ihrer Website er sich gerade befindet. Hierbei soll jeder Eintrag einen Link auf die entsprechende Hierarchieebene enthalten.

Lösung

Umgeben Sie die Links mit einem `div`-Element und fügen Sie im HTML-Code zwischen den Links ein Pfeilsymbol ein:

```
<div id="brotkruemel">
  <a href="/startseite">Startseite</a> &#8594;
  <a href="/produkte">Produkte</a> &#8594;
  <a href="/produkte/instrumente/">Instrumente</a> &#8594;
  <a href="/produkte/instrumente/westerngitarren/">Westerngitarren</a> &#8594;
  <a href="/produkte/instrumente/westerngitarren/cf-martin/">CF Martin</a> &#8594;
  <strong>16-Series</strong>
</div>
```

Und hier die dazugehörigen CSS-Regeln:

```
#brotkruemel {
  margin: 1em;

  background: rgb(170,210,210);
  background: rgba(200,200,200,0.5);

  border: 1px solid black;
  -webkit-border-radius: 5px;
  -moz-border-radius: 5px;
  border-radius: 5px;

  -webkit-box-shadow: 3px 3px 5px rgba(10,10,10,0.4);
  -moz-box-shadow: 3px 3px 5px rgba(10,10,10,0.4);
  box-shadow: 3px 3px 5px rgba(10,10,10,0.4);
}
```

Zu Beginn versehen wir das `div`-Element, das die Links enthalten soll, mit etwas Außenabstand. Für die Hintergrundfarbe verwenden wir eine doppelte `background`-Deklaration. Browser, die bereits mit der `rgba()`-Notation umgehen können, stellen den Hintergrund halb transparent dar. Die erste `background`-Anweisung dient als Fallback für ältere Browser. Außerdem umgeben wir das Element mit einem schlichten schwarzen Rahmen.

Um die Gestaltung abzurunden erhält die Navigation in Browsern, die dies verstehen, abgerundete Ecken und einen Schattenwurf. Die folgende Regel befasst sich mit den Links:

```
#brotkruemel a {
  display: inline-block;
  padding: 0.2em 0.5em;

  text-decoration: none;
  color: black;
}
```

Um die Links leichter bedienbar zu machen, erhalten sie die Darstellungsrolle `inline-block`. Hierdurch sind auch die per `padding` definierten Bereiche noch anklickbar. Außerdem entfernen wir die Unterstreichung und legen fest, dass die Links in schwarzer Schrift dargestellt werden sollen.

```
#brotkruemel a:hover, #brotkruemel a:focus {
    background-color: #f4a460;
    text-decoration: underline;
}
```

Befindet sich der Cursor über einem Link oder hat er den Fokus über die Tastatur erhalten, ändern wir dessen Hintergrundfarbe und zeigen außerdem eine Unterstreichung an. Da sich die Links in der Darstellung »vor« dem `div`-Element befinden, überlagert der erste Link dessen abgerundete Ecken. Dieses ästhetische Problem bekommen wir mit der folgenden Regel in den Griff:

```
#brotkruemel a:first-child {
    -webkit-border-radius: 5px 0 0 5px;
    -moz-border-radius: 5px 0 0 5px;
    border-radius: 5px 0 0 5px;
}
```

Hier verwenden wir die Vier-Werte-Form der Kurzschrittschreibweise für `border-radius`, um nur die beiden linken Ecken abzurunden.

Diskussion

Die Bezeichnung *Brotrümelnavigation* geht auf das Märchen *Hänsel und Gretel* zurück. Eine solche Spur soll den beiden helfen, wieder nach Hause zu finden. Im Web zeigt eine solche Spur dem Benutzer innerhalb der Seitenstruktur den Weg zur gerade betrachteten Seite (siehe Abbildung 6-8).



Abbildung 6-8: Beispiel für eine Brotrümelspur

Bei der Gestaltung von Breadcrumb-Navigationen sollte das letzte Element nicht verlinkt sein. Daher heben wir es in unserem Beispiel hervor, indem wir es mit `strong`-Tags umgeben, die Sie bei Bedarf über den passenden Elementselektor ebenfalls mit eigenen Stilen versorgen können.

Siehe auch

Den deutschsprachigen Wikipedia-Artikel zum Thema unter <http://de.wikipedia.org/wiki/Brotkrümelnavigation>; eine Anleitung zur Suchmaschinenoptimierung für Breadcrumb-Navigationen unter <http://www.mindshape.de/blog/seo/11-regeln-fur-eine-seo-freundliche-breadcrumb.html>; Untersuchungsergebnisse zur Effektivität von Breadcrumb-Navigationen unter <http://psychology.wichita.edu/surl/usabilitynews/51/breadcrumb.asp>; eine Ausgabe von »Hänsel und Gretel« unter <http://gutenberg.spiegel.de/buch/754/254>.

6.10 Karteireiter-Navigationsmenüs

Problem

Aus einer ungeordneten Linkliste soll ein horizontales »Karteireiter«-Navigationsmenü erzeugt werden. Hierbei soll es zwischen Navigation und Inhaltsbereich keinen Zwischenraum geben, wie in Abbildung 6-9 zu sehen.



Abbildung 6-9: Ein schlichtes Karteireiter-Navigationsmenü mit direkt angrenzendem Inhaltsbereich

Lösung

Definieren Sie die Listeneinträge als Inline-Elemente und verwenden Sie die Eigenschaften zur Gestaltung von Rahmen (`border-*`), um die Karteireiter zu erzeugen.

Wie in den anderen Rezepten zur Erstellung von Navigationsmenüs aus Listen werden hier ebenfalls zunächst die Innen- und Außenabstände für die Liste und die enthaltenen Elemente entfernt:

```
#karteireiter, #karteireiter li {  
    margin: 0;  
    padding: 0;  
}
```

Danach kümmern wir uns um die Liste selbst (#karteireiter). Zunächst schaffen wir per margin: 1em um die Liste etwas Platz. Die Unterseite der Liste erhält dagegen ausdrücklich keinen Außenabstand, damit sie ohne Zwischenraum an den Inhaltsbereich anschließt. Außerdem werden in dieser Regel die Aufzählungszeichen verborgen:

```
#karteireiter {  
  margin:      1em;  
  margin-bottom: 0;  
  list-style:   none;  
}
```

Den Kern dieses Rezepts finden Sie in der nächsten Regel (#karteireiter li). Hier legen wir fest, dass die Listeneinträge als Inline-Elemente, also nebeneinander, dargestellt werden sollen:

```
#karteireiter li {  
  display: inline;  
}
```

Die Hauptarbeit findet jedoch in der folgenden Regel statt. (Der zweite Selektor sorgt dafür, dass besuchte Links die gleichen Stildefinitionen erhalten.) Auch hier ändern wir die Darstellungsrolle. Links sind eigentlich Inline-Elemente. Damit der gesamte Karteireiter anklickbar ist, geben wir ihnen die Darstellungsrolle Inline-Block.

Danach können wir die Links nach Belieben mit Innenabständen und Hintergrundfarbe versehen. Auch hier verwenden wir die doppelte background-Schreibweise (siehe Grundrezept 2.15), um moderne Browser gegebenenfalls mit einer halb transparenten Hintergrundfarbe zu versehen, ohne ältere Modelle »außen vor« zu lassen.

Auch die Rahmen werden für die Links (und nicht für die Listenelemente) angelegt. Hierbei bleibt die Unterkante ohne Rahmen; sie wird später vom Inhaltsbereich bereitgestellt:

```
#karteireiter a:link,  
#karteireiter a:visited {  
  display: inline-block;  
  padding: 0.5em;  
  
  background: rgb(170,210,210);  
  background: rgba(200,200,200,0.5);  
  
  text-decoration: none;  
  color:          black;  
  
  border:         1px solid black;  
  border-bottom: none;  
}
```

Damit der Benutzer auch merkt, dass es sich bei den Karteireitern um Links handelt, ändern wir in der folgenden Regel (#karteireiter a:hover) die Hintergrundfarbe und versehen die Links außerdem mit einer Unterstreichung, sobald sich der Mauszeiger darüber befindet:


```
#karteireiter a:hover {
    background-color: #f4a460;
    text-decoration: underline;
}
```

Damit ist die Gestaltung der Liste abgeschlossen, und wir können uns dem Inhaltsbereich (#inhaltsbereich) zuwenden. Er erhält den gleichen seitlichen Außenabstand wie die Liste. Der obere Außenabstand wird ausdrücklich auf null gesetzt, damit kein unerwünschter Zwischenraum entsteht.

Außerdem erhält der Inhaltsbereich rundherum einen Innenabstand von einer em-Einheit und wird mit einem schwarzen Rahmen umgeben. Und hier sehen Sie auch den Grund dafür, dass wir für die Listenelemente bzw. die Links keine Rahmenunterkante definiert haben: Da die Oberkante des Inhaltsbereichs bündig mit der Liste abschließt, stellt sie die untere Rahmenkante für die Karteireiter quasi gratis zur Verfügung.

Hätten wir `border-bottom` nicht auf null gesetzt, wäre der Rahmen unterhalb der Karteireiter jetzt doppelt so dick (2 Pixel).

```
#inhaltsbereich {
    margin: 1em;
    margin-top: 0;
    padding: 1em;

    border: 1px solid black;
}
```

Und hier der HTML-Code dazu:

```
<ul class="karteireiter">
  <li><a href="/startseite">Startseite</a></li>
  <li><a href="/produkte">Produkte</a></li>
  <li><a href="/ueber-uns">Über uns</a></li>
  <li><a href="/kontakt">Kontakt</a></li>
  <li><a href="/impressum">Impressum</a></li>
</ul>
<div id="inhaltsbereich">...</div>
```

Diskussion

Die Verwendung von Karteireitermenüs ist im Web recht beliebt, weil sie das Verhalten vieler Desktopprogramme nachahmt. Suchen Sie im Web einmal nach »CSS tabbed menu« und Sie werden Hunderte, wenn nicht Tausende von Seiten angezeigt bekommen.

Die hier gezeigte Lösung ist daher nicht die einzige Möglichkeit, sondern sie stellt eine solide Ausgangsbasis für eigene Experimente und Erweiterungen dar. Mit ein paar zusätzlichen CSS-Anweisungen können Sie die Karteireiter beispielsweise mit abgerundeten Ecken, Farbverläufen und Schattenwürfen versehen. Wenn Sie außerdem noch die aktuelle Seite hervorheben, wie in Rezept 6.8 beschrieben wird, erhalten Sie in etwa die Darstellung aus Abbildung 6-10.



Abbildung 6-10: Die Karteireiter-Navigation mit zusätzlichen Stildefinitionen

Der Einsatz von Karteireiter-Navigationen (oder Tabmenüs) ist nicht ganz ohne Fallstricke. Im Gegensatz zu Desktopanwendungen gibt es zwei wesentliche Unterschiede: Der Inhalt des Tabs wird beim Anklicken direkt angezeigt (das ließe sich möglicherweise per AJAX oder ähnlichen Technologien nachrüsten). Außerdem können Tabs bei Desktopprogrammen auf vielfältige Weise über die Tastatur bedient werden. Das ist im Web nur bedingt der Fall. So wird bei Benutzung der Tabulatortaste zwar jeder Karteireiter angesprungen, erhält also den Fokus und der Inhalt wird aber nicht angezeigt.

Das soll Sie jetzt nicht davon abhalten, Karteireiter-Menüs in Ihre Website einzubauen. Sie sollten sich über die möglichen Risiken und Nebenwirkungen aber zumindest im Klaren sein.

Siehe auch

Das Rezept 6.7 zur Gestaltung von Listen als Navigationsmenüs; das Grundrezept 2.7 zur Verwendung der Eigenschaft `border`; das Kapitel 12, *Design mit CSS*, unter anderem zur Definition von Schattenwürfen und abgerundeten Ecken; das Rezept 4.5 unter anderem zur Verwendung der Darstellungsrolle `inline-block` im IE7; das Rezept 1.17 zu Darstellungsrollen; den Artikel *Accessible Tabs* von Emily Coward unter <http://www.nomensa.com/blog/2011/accessible-tabs-part-1-the-problem/>; den Artikel *Accessible ARIA Tabs* zur Anwendung von WAI-ARIA-Rollen (Accessible Rich Internet Applications) für Karteireitermenüs unter <http://www.accessibleculture.org/articles/2010/08/aria-tabs/>.

6.11 Ein vertikales Navigationsmenü mit mehreren Ebenen erstellen

Problem

Sie möchten ein Navigationsmenü erstellen, das mehrere Hierarchieebenen enthält (Fly-out-Menü), die nur bei Bedarf angezeigt werden.

Lösung

Verwenden Sie das Navigationsmenü aus Rezept 6.7 als Ausgangspunkt. Erweitern Sie die ungeordneten Listen nach Bedarf um weitere Ebenen. Die Unterlisten werden per `display: none` versteckt und nur angezeigt, wenn sich der Mauszeiger darüber befindet. Hier die wichtigsten CSS-Regeln:

```
#navigation, #navigation * {
    margin: 0;
    padding: 0;
}

#navigation {
    list-style: none;
    width: 10em;
}

#navigation li > ul {
    display: none;
}

#navigation li:hover {
    position: relative;
}

#navigation li:hover > ul {
    display: block;
    position: absolute;
    top: 0.4em;
    left: 8em;
    width: 10em; /* ghostbuster! */
    list-style: none;
}
```

In der ersten Regel werden die Innen- und Außenabstände für die Navigationsliste (`#navigation`) und alle darin enthaltenen Elemente (`#navigation *`) auf null zurückgesetzt. Dadurch werden unter anderem die üblichen Einrückungen für alle verwendeten `ul`- und `li`-Elemente auch in den enthaltenen Unterlisten zuverlässig entfernt.

Direkt danach legen wir die grundsätzliche Darstellung für die Liste der obersten Ebene fest. Die Aufzählungszeichen werden entfernt, die Breite wird mit `10em` angegeben. Dadurch passt sich die Größe der Liste und ihrer Elemente automatisch an die Textgröße an.

Die dritte Regel befasst sich mit den Unterlisten, die nur bei Bedarf angezeigt werden sollen. Das erreichen wir, indem wir sie per `display: none` komplett von Darstellung ausnehmen.

Die Unterlisten sollen entgegen der Standarddarstellung neben dem jeweiligen Listenelement erscheinen, das sie enthält. Das erreichen wir mithilfe der Positionierung in der vierten Regel (`#navigation li:hover`). Sie gilt für alle in `#navigation` enthaltenen `li`-Elemente: Sobald sich der Mauszeiger darüber befindet, werden sie relativ positioniert und schaffen so den nötigen Kontext für die absolute Positionierung gegebenenfalls enthaltener Unterlisten.

Befindet sich der Mauszeiger über einem Listenelement, das eine Unterliste als Kindelement enthält, greift Regel Nummer 5 (`#navigation li:hover > ul`).

Sind die Voraussetzungen erfüllt, wird die Unterliste per `display: block` eingeblendet und absolut positioniert, wobei das umgebende `li`-Element als Positionierungskontext dient.

Mit den Eigenschaften `top` und `left` verschieben wir die Unterliste ein Stückchen (`0.4em`) nach unten und um `8em` nach rechts. Sie überlagert das umgebende `li`-Element also ein wenig.

Außerdem geben wir den Unterlisten erst an dieser Stelle ihre Breite (`10em`). Das hat mit dem gefürchteten »hover ghost-bug« im Internet Explorer 7 zu tun. Wird die Breite für alle `ul`-Listen auf einmal definiert, kann es im IE7 zu Darstellungsproblemen kommen. Zum Schluss werden auch hier noch die Aufzählungszeichen entfernt. Das fertige Ergebnis sehen Sie in Abbildung 6-11.



Abbildung 6-11: Ein horizontales Navigationsmenü mit mehreren Ebenen

Dieses Rezept basiert auf folgendem HTML-Code:

```
<ul id="navigation">
  <li><a href="/startseite">Startseite</a></li>
  <li><a href="/produkte">Produkte</a>
    <ul>
      <li><a href="/produkte/jazz">Jazzgitarren</a></li>
      <li><a href="/produkte/western">Westerngitarren</a>
        <ul>
          <li><a href="/produkte/western/cf-martin">CF Martin</a></li>
          <li><a href="/produkte/western/pendennis">Pendennis</a></li>
          <li><a href="/produkte/western/yairi">H. Yairi</a></li>
        </ul>
      </li>
      <li><a href="/produkte/jazzgitarren">Klassikgitarren</a></li>
      <li><a href="/produkte/saiten">Saiten</a></li>
    </ul>
  </li>
  <li><a href="/ueber-uns">Über uns</a></li>
  <li><a href="/kontakt">Kontakt</a></li>
  <li><a href="/impressum">Impressum</a></li>
</ul>
```

Wir haben in der HTML-Struktur die Navigationsliste mit einem class-Attribut gekennzeichnet, um sie von anderen Listen zu unterscheiden.

Diskussion

Das Grundprinzip dieser Lösung besteht darin, nicht gebrauchte Unterlisten nur bei Bedarf einzublenden. Würde man hier die Positionierung nicht einsetzen, würden diese gemäß ihrer Platzierung im HTML-Code zwischen den Listeneinträgen der ersten bzw. zweiten Ebene erscheinen. Durch die absolute Positionierung werden sie vom normalen Dokumentenfluss ausgenommen und können frei platziert werden.

Bei Bedarf (wenn sich der Mauszeiger über einem passenden li-Element befindet) schaffen wir hierfür per position: relative den nötigen Positionierungskontext.

Um die Lösung nicht unnötig mit gestalterischen Elementen (Farben usw.) zu überfrachten, sind wir im Lösungscode nicht weiter auf das Design eingegangen. Das wird jetzt nachgeholt. Für die Abbildung haben wir zusätzlich zu den Regeln in der Lösung die folgenden Stildefinitionen festgelegt:

```
#navigation li {
  background:    rgb(200,200,200) url('bilder/verlauf.gif') repeat-x;
  border-left:   5px solid black;
  border-right:  5px solid black;
  border-bottom: 1px solid black;
}

#navigation li:first-child {
  border-top: 1px solid black;
}
```

```

#navigation li:hover,
#navigation li:hover li:hover {
    background: orange;

    border-left: 5px solid rgb(55,91,126);
    border-right: 5px solid rgb(55,91,126);
}

#navigation li a:link,
#navigation li a:visited {
    display: inline-block;

    width: 9em; /* Breite 10em - border-left und border-right */
    padding: 0.2em;

    font-family: Helvetica, Arial, sans-serif;
    color: rgb(10,10,10);
    text-decoration: none;
}

```

Zunächst wird für alle Listeneinträge (`#navigation li`) im Navigationsmenü der Hintergrund definiert. Nach Möglichkeit soll das Hintergrundbild *verlauf.gif* verwendet werden. Konnte das nicht geladen werden, wird stattdessen ein schlichter hellgrauer Hintergrund angezeigt.

Es folgen die Definitionen für die einzelnen Rahmenseiten. Links und rechts ist der Rahmen 5 Pixel breit, während unterhalb der Listenelemente nur eine 1 Pixel breite Linie zu sehen sein soll.

Auf die Definition der oberen Rahmenseite wird hier absichtlich verzichtet, damit sie zwischen den einzelnen Listenelementen nicht verdoppelt erscheint. Das führt uns zur nächsten Regel.

Da wir keine Rahmenoberseite definiert haben, fehlt diese beim ersten Listenelement. Das können wir aber ändern, indem wir die Pseudoklasse `:first-child` verwenden, um nur für den jeweils ersten Listeneintrag (`#navigation li:first-child`) die Rahmenoberseite anzulegen.

Die dritte Regel steuert, was passiert, wenn sich der Mauszeiger über einem Listenelement befindet. Um dem Besucher eine visuelle Rückmeldung zu geben, wird in diesem Fall der Hintergrund orange eingefärbt. Außerdem erhalten die Rahmenseiten einen kontrastierenden Blauton.

Bei genauem Betrachten fällt auf, dass wir die Formulierung `li:hover` in der zweiten Zeile gleich zweimal hintereinander (als Nachfahrenselektor) verwendet haben:

```

#navigation li:hover,
#navigation li:hover li:hover {
    color: orange;
}

```

Der zweite Selektor sorgt dafür, dass nicht nur das Element hervorgehoben wird, über dem sich der Mauszeiger gerade befindet, sondern auch die Listeneinträge, die dort hinführen, was die Navigation deutlich erleichtern kann.

Es folgen die Regeln für die Links selbst. Besuchte und noch nicht angeklickte Links (`ul.navigation li a:link` bzw. `ul.navigation li a:visited`) erhalten die gleiche Gestaltung. Wie in Rezept 6.6 erklärt, wird dadurch der anklickbare Bereich auf das gesamte Listenelement ausgedehnt.

Wir geben den Links eine Breite von `9em`, was in etwa der Breite der Listenelemente selbst, abzüglich der Breite der seitlichen Rahmen entspricht. Damit die Links nicht allzu eingengt erscheinen, erhalten sie per `padding: 0.2em` rundherum ein wenig Leerraum.

Es folgen ein paar Angaben zu Schriftart (`font-family`) und -farbe (`color`). Außerdem entfernen wir die standardmäßige Unterstreichung, da die Links bereits durch ihre Funktion im Navigationsmenü als solche erkennbar sind und in der vorigen Regel durch die Änderung der Hintergrundfarbe ausreichend hervorgehoben werden.

Geisterstunde

Der IE7 hat bei der Darstellung ein Problem, das mit dem Konzept des *hasLayout* zusammenhängt. Erhalten die Unterlisten ihre Breite, bevor sie tatsächlich angezeigt werden, kann es passieren, dass neben der Liste der zweiten Ebene ein »Geist« der dritten Unterliste erscheint – allerdings ohne Inhalt!

Das können wir beheben, indem wir die Breite der Unterlisten erst festlegen, wenn sich der Mauszeiger über dem umgebenden `li`-Element befindet, wie es auch in der Lösung beschrieben wird.

Siehe auch

Das Grundrezept 2.17 zur absoluten Positionierung; das Grundrezept 2.19 zum Positionierungskontext; das Rezept 1.17 zur Änderung der Darstellungsrolle von Elementen; das Rezept 1.14 zur Verwendung der Pseudoklasse `:first-child`; das Rezept 1.4 zur Gruppierung von Selektoren; eine ausführliche Betrachtung des »hover ghost-bug« von Vilem Málek unter <http://murphy.cz/ie7-hover-ghosts-bug/>; das Kapitel 13, *Tipps und Tricks*, zur Behebung diverser Probleme mit dem Internet Explorer.

6.12 Ein horizontales Navigationsmenü mit mehreren Ebenen erstellen

Problem

Es soll ein sogenanntes Drop-down-Menü erstellt werden. Hierfür soll das Rezept 6.11 so abgeändert werden, dass die erste Listenebene horizontal dargestellt wird. Die Unterliste

der zweiten Ebene soll direkt unterhalb des umgebenden Listenelements (z.B. *Produkte*) erscheinen. Die Unterliste der dritten Ebene soll die gleiche Darstellung erhalten, wie in Abbildung 6-11 gezeigt.



Um die Lösungsbeschreibung einfach zu halten, haben wir die Regeln für die Gestaltung (Rahmen, Hintergründe usw.) hier nicht aufgeführt. Wir gehen aber in der Diskussion im Abschnitt »Eye Candy« ausführlich darauf ein.

Lösung

Benutzen Sie das Rezept 6.11 als Ausgangsbasis und nehmen Sie die nötigen Änderungen vor. Die Hauptarbeit besteht darin, die `li`-Elemente von `#navigation` als `Inline-Elemente` zu definieren und die Werte für die Positionierung der Unterlisten anzupassen. Zuvor müssen aber noch ein paar Änderungen in den anderen Regeln vorgenommen werden. Wir beginnen mit Regel Nummer 2:

```
#navigation {  
  list-style: none;  
  height:    28px;  
}
```

Hier haben wir die Breitenangabe entfernt. Die Liste soll so breit sein, wie es nötig ist, um alle Einträge aufzunehmen. Stattdessen geben wir `#navigation` eine feste Höhe.

An den folgenden Regeln 3 und 4 (`#navigation li > ul` und `#navigation li:hover`) müssen keine Änderungen vorgenommen werden. Die dritte Regel versteckt die Unterlisten, bis sie gebraucht werden, die vierte schafft bei Bedarf den nötigen Positionierungskontext für die unteren Ebenen.

Es geht weiter mit Regel 5 (`#navigation li:hover > ul`). Hier müssen die Werte für die Eigenschaften `top` und `left` angepasst werden, damit die Liste der zweiten Ebene unterhalb des entsprechenden Listeneintrags erscheint:

```
#navigation li:hover > ul {  
  display:    block;  
  position:   absolute;  
  
  top:        27px;  
  left:       -1px;  
  width:      10em;  
  list-style: none;  
}
```

Die erste Unterliste soll direkt unterhalb des dazugehörigen Listenelements platziert werden. Da als Ursprung für die Angaben von `top` und `left` die linke obere Ecke der »Padding-Box« des umgebenden Elements gilt, müssen wir den Wert von dort aus berechnen. Hierfür müssen wir den Wert für `line-height` der Listeneinträge (20px) mit den Werten für `padding-top` und `padding-bottom` (jeweils 3px) und der Breite des Rahmens (1px) addieren. Als Ergebnis erhalten wir einen Wert von 27 Pixeln.



Bei der Verwendung von em-Einheiten kann in Opera und Internet Explorer 7 zwischen dem Listeneintrag und der Unterliste eine Lücke entstehen. Daher verwenden wir hier Pixel als Maßeinheit.

Mit `left: -1px` gleichen wir auch die Rahmenbreite aus. Ansonsten würde die Unterliste um 1 Pixel nach rechts verschoben erscheinen.

Ohne weiteres Zutun würde diese Regel auch für die Liste der dritten Ebene gelten. Diese soll jedoch so dargestellt werden, wie wir es in Rezept 6.11 bereits gezeigt haben. Also definieren wir eine weitere Regel, die die Werte für `top` und `left` korrigiert:

```
#navigation li li:hover > ul {  
    top:      0.4em;  
    left:     8em;  
}
```

Wir haben uns die Arbeit leicht gemacht und den Selektor einfach um ein weiteres `li` erweitert. Dadurch gilt die Regel (von rechts nach links gelesen) für: »Eine ungeordnete Liste (`ul`), die ein Kindelement eines Listenelements (`li`) ist, über dem sich gerade der Mauszeiger befindet (`:hover`). Dieses `li`-Element soll ein Nachfahre eines weiteren Listenelements (`li`) sein, das seinerseits von `#navigation` abstammt.«, oder auch: »für die Liste der dritten Ebene«. Diese Schreibweise erspart es uns, die übrigen Deklarationen für die dritte Liste zu wiederholen.

Nach diesen Änderungen fügen wir endlich die Regel ein, die dafür sorgt, dass die Listenelemente der ersten Ebene nebeneinander dargestellt werden:

```
#navigation li {  
    display: inline;  
    float: left;  
    line-height: 20px;  
}
```

Sie gilt für alle `li`-Elemente von `#navigation` und ihren Unterlisten. Da die Browser nach einem Listenelement standardmäßig einen Zeilenumbruch einfügen, werden sie normalerweise untereinander dargestellt. Das ändern wir, indem wir die `li`-Elemente als Inline-Elemente definieren.

Häufig stehen die Listeneinträge im HTML-Code untereinander, z.B.:

```
<li><a href="/produkte/jazz">Jazzgitarren</a></li>  
<li><a href="/produkte/western">Westerngitarren</a></li>
```

Dadurch entstehen in manchen Browsern unschöne Zwischenräume zwischen den Einträgen. Durch die Definition als Floats (`float: left`) rücken die Elemente so eng zusammen wie möglich. Die dritte Deklaration (`line-height: 20px`) legt die Zeilenhöhe der Listenelemente fest. Hierdurch können wir die absolute Positionierung der ersten Unterliste genau steuern. Wir verwenden hier Pixelwerte, weil der Einsatz von em-Einheiten zwischen den Browsern zu Rundungsfehlern beim Ausgleich der Rahmenbreiten führen kann.

Diskussion

Bei der Erstellung eines horizontalen Navigationsmenüs aus verschachtelten Listen gibt es einiges mehr zu beachten als bei der vertikalen Variante. Aber mit ein paar Anpassungen sind auch hier ansprechende Ergebnisse möglich, die in allen aktuellen Browsern funktionieren.

Für das Design haben wir die seitlichen Rahmenkanten verringert. Anstatt wie im Ausgangsrezept die Rahmenkanten einzeln zu definieren, legen wir dieses Mal fest, dass alle Listenelemente von einem schwarzen Rahmen umgeben sein sollen. Diese allgemeine Regel wird im Folgenden entsprechend modifiziert:

```
#navigation li {  
    background: rgb(230,230,230) url('bilder/verlauf.gif') repeat-x;  
    border:    1px solid black;  
}
```

Damit die Rahmen zwischen den Listenelementen von #navigation nicht doppelt erscheinen, wird die linke Rahmenkante entfernt:

```
#navigation > li {  
    border-left: none;  
}
```

Da auch der erste Listeneintrag von der vorigen Regel betroffen ist, benutzen wir hier die Pseudoklasse :first-child, um ihn wieder erscheinen zu lassen:

```
#navigation > li:first-child {  
    border-left: 1px solid black;  
}
```

Aus dem gleichen Grund wird die Rahmenoberkante für die Elemente der zweiten und dritten Ebene ausgeblendet:

```
#navigation li li {  
    border-top: none;  
}
```

Das funktioniert, weil wir keinen Kindselektor verwenden. Dadurch gilt die Regel für alle Listenelemente, die ein weiteres Listenelement zum Vorfahren haben. Nur für das erste Listenelement der dritten Ebene müssen wir die Kante wieder zum Vorschein bringen:

```
#navigation li li li:first-child {  
    border-top: 1px solid black;  
}
```

Die folgende Regel wurde nicht verändert. Sie sorgt dafür, dass die Listeneinträge ihre Farbe wechseln, wenn sich der Mauszeiger darüber befindet:

```
#navigation li:hover,  
#navigation li:hover li:hover {  
    background: orange;  
}
```

Anstatt, wie im Rezept für das vertikale Navigationsmenü, die Breite und den Innenabstand für die Links in em-Einheiten anzugeben, verwenden wir hier Pixelwerte. Ansonsten kann es in Opera und Internet Explorer 7 zu Problemen bei der Positionierung der Unterlisten kommen. Davon abgesehen, gibt es auch hier keine Änderungen:

```
#navigation li a:link,
#navigation li a:visited {
    display: inline-block;

    width: 160px;
    padding: 3px;

    font-family: Helvetica, Arial, sans-serif;
    color: rgb(10,10,10);
    text-decoration: none;
}
```

Das Ergebnis unserer Bemühungen sehen Sie in Abbildung 6-12.

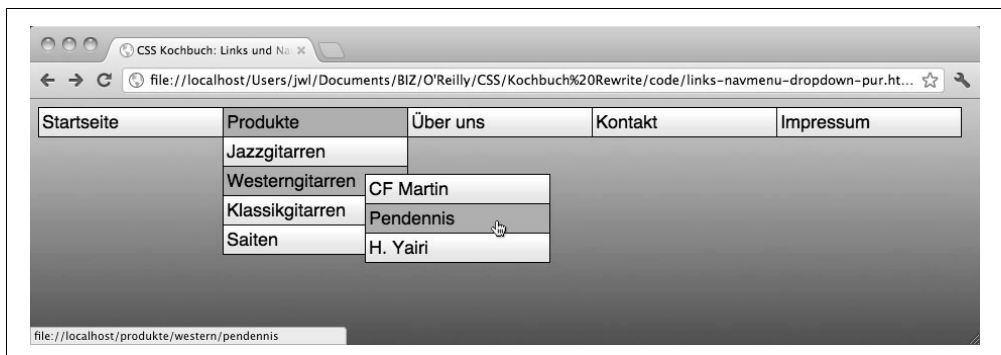


Abbildung 6-12: Ein horizontales Navigationsmenü mit zwei Unterebenen

Eye Candy

Wenn Sie sich die Prinzipien des »Progressive Enhancement« zu eigen machen wollen, können Sie die Darstellung des Menüs noch verfeinern und so die Benutzerfreundlichkeit weiter erhöhen. Diese Regeln funktionieren nur mit Browsern, die bereits mit den CSS3-Eigenschaften `box-shadow`, `transition`, `border-radius` und `box-shadow` umgehen können (entweder in der offiziellen oder in der Präfixschreibweise). Können sie das nicht, ist das kein Beinbruch. Die Listen werden weiterhin korrekt dargestellt – nur eben ohne die Erweiterungen.

Im ersten Schritt versehen wir die Listenelemente und die Unterlisten mit einem Schattentwurf, um etwas räumliche Tiefe zu schaffen:

```
#navigation > li, #navigation ul {
    -moz-box-shadow: 3px 3px 5px rgba(10,10,10,0.4);
    -webkit-box-shadow: 3px 3px 5px rgba(10,10,10,0.4);
    box-shadow: 3px 3px 5px rgba(10,10,10,0.4);
}
```

Für die Liste der ersten Ebene wird der Schatten an den Listeneinträgen bzw. den Unterlisten erzeugt. Der Grund liegt darin, dass sich `#navigation` als Blockelement über die gesamte Breite des Ansichtsbereichs erstreckt. Dadurch wäre der Schatten breiter als nötig.

Im nächsten Schritt erhalten die Beschriftungen ebenfalls einen leichten, kaum wahrnehmbaren Schatten. Außerdem verwenden wir hier die Eigenschaft `transition`, um den Farbwechsel der Listenelemente zu animieren.

```
#navigation li {
    text-shadow: 1px 1px 3px rgba(100,100,100,0.5);

    -webkit-transition: all 0.2s ease-in-out;
    -moz-transition:    all 0.2s ease-in-out;
    -o-transition:     all 0.2s ease-in-out;
    transition:        all 0.2s ease-in-out;
}
```

Um die räumliche Tiefe weiter zu verstärken, ändern wir den Schattenwurf, wenn sich der Mauszeiger über einem Listenelement befindet. Der Schattenwurf wird nun innerhalb des Listenelements angezeigt (durch den Zusatz `inset`) und erweckt so den Eindruck eines heruntergedrückten Buttons.

```
#navigation li:hover {
    -moz-box-shadow: 1px 1px 1px rgba(10,10,10,0.2) inset;
    -webkit-box-shadow: 1px 1px 1px rgba(10,10,10,0.2) inset;
    box-shadow: 1px 1px 1px rgba(10,10,10,0.2) inset;
}
```

Den letzten Schliff erreichen wir, indem wir die Position der Beschriftung beim Überfahren mit der Maus ganz leicht verschieben. Das macht den Eindruck eines gedrückten Buttons etwas realistischer.

```
#navigation a:hover {
    position: relative;
    top: 1px;
    left: 1px;
}
```

Damit diese Verschiebung den Internet Explorer (7 und 8) nicht verwirrt, müssen wir diese Regel in einem Conditional Comment wieder rückgängig machen, indem wir `position` wieder mit dem Standardwert `static` versehen:

```
<!--[if lte IE 8]>
<style type="text/css">
#navigation a:hover {
    position: static;
}
</style>
<![endif]-->
```

Zum Schluss versehen wir das Menü noch mit abgerundeten Ecken. Wir beginnen mit dem linken Listenelement der ersten Ebene:

```
#navigation > li:first-child {
    -webkit-border-top-left-radius: 7px;
    -webkit-border-bottom-left-radius: 7px;
    -moz-border-radius-topleft: 7px;
    -moz-border-radius-bottomleft: 7px;
    border-top-left-radius: 7px;
    border-bottom-left-radius: 7px;
}
```

Es folgt das letzte Listenelement der ersten Ebene. Hierfür setzen wir die in CSS3 neu hinzugekommene Pseudoklasse `:last-child` ein:

```
#navigation > li:last-child {
    -webkit-border-top-right-radius: 7px;
    -webkit-border-bottom-right-radius: 7px;
    -moz-border-radius-topright: 7px;
    -moz-border-radius-bottomright: 7px;
    border-top-right-radius: 7px;
    border-bottom-right-radius: 7px;
}
```

Die folgende Regel definiert die abgerundeten Ecken an der Unterseite für die Listen auf der zweiten und dritten Ebene. Hier müssen wir die Werte für das letzte Element wie auch für die Liste selbst angeben:

```
#navigation > li > ul,
#navigation > li > ul > li:last-child,
#navigation > li > ul > li > ul,
#navigation > li > ul > li > ul > li:last-child {
    -webkit-border-bottom-left-radius: 7px;
    -webkit-border-bottom-right-radius: 7px;
    -moz-border-radius-bottomleft: 7px;
    -moz-border-radius-bottomright: 7px;
    border-bottom-left-radius: 7px;
    border-bottom-right-radius: 7px;
}
```

Das Gleiche gilt für die Oberkante der Liste auf der dritten Ebene:

```
#navigation > li > ul > li > ul,
#navigation > li > ul > li > ul > li:first-child {
    -webkit-border-top-left-radius: 7px;
    -webkit-border-top-right-radius: 7px;
    -moz-border-radius-topleft: 7px;
    -moz-border-radius-topright: 7px;
    border-top-left-radius: 7px;
    border-top-right-radius: 7px;
}
```

Damit sind sämtliche Arbeiten abgeschlossen. Das Ergebnis ist eine Navigationsliste, die in modernen Browsern fast schon vergleichbar ist mit der eines Desktop-Programms (siehe Abbildung 6-13). In älteren Browsern fehlen diese Erweiterungen zwar, aber die

Funktionalität bleibt vollständig erhalten. Ausführliche Informationen zur Verwendung der hier verwendeten CSS3-Eigenschaften finden Sie übrigens in Kapitel 12, *Design mit CSS*.

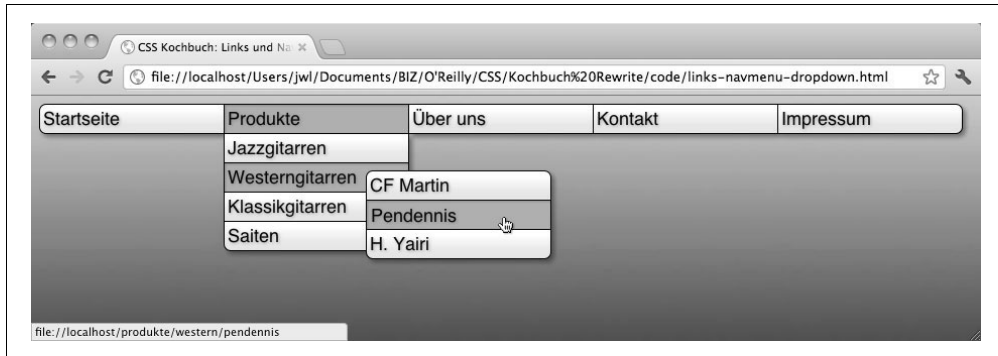


Abbildung 6-13: Ein horizontales Navigationsmenü mit allen Schikanen

Das vollständige Stylesheet inklusive der zusätzlichen Gestaltungsmöglichkeiten finden Sie im Beispielcode zu diesem Buch unter http://examples.oreilly.de/german_examples/cssckbk3ger.

Siehe auch

Das Rezept 6.11 zur Erstellung eines vertikalen Navigationsmenüs; das Rezept 1.16 unter anderem zur Erklärung der »Padding-Box«; das Rezept 1.17 zur Erklärung der Darstellungsrollen; die CSS 2.1-Spezifikation zur Platzierung von Floats unter <http://www.w3.org/TR/CSS2/visuren.html#float-position>; das Kapitel 12, *Design mit CSS*, zur Verwendung der neuen CSS3-Eigenschaften; das Rezept 1.14 unter anderem zur Verwendung der Pseudoklasse :last-child.

Einführung

Ohne HTML-Formulare wäre es nicht möglich, sich über das Web auf seinem E-Mail-Konto einzuloggen, Bücher mit einem Mausklick zu bestellen oder über das Netz mit Aktien zu handeln. Obwohl Formulare heutzutage ein essenzieller Bestandteil des Webs sind, lässt sich an der Standarddarstellung der Browser noch einiges verbessern.

Formulare werden mit der Fähigkeit, Webapplikationen zu programmieren, und der Verbreitung von HTML5 immer wichtiger. Eigentlich schaffen Formulare erst die Möglichkeit, mit den Besuchern einer Website zu interagieren und auf ihre Eingaben zu reagieren.

Leider ist die Darstellung der Formularelemente zwischen den verschiedenen Browsern alles andere als einheitlich. Es wimmelt nur so von Fallstricken, Wechselwirkungen und anderen bösen Dingen. Wir zeigen Ihnen dort, wo es möglich ist, wie Sie die schlimmsten Gefahren am besten umgehen können.

Die Standarddarstellung von Formularen besteht in der Regel aus abgekanteten Eingabe- und textarea-Feldern sowie aus langweiligen Buttons. Diese Art von »Look-and-feel« ist für ein Intranet oder eine kleine Website vielleicht gerade noch akzeptabel. Den Eindruck von Professionalität verleihen Sie Ihrem Projekt damit jedoch nicht.

Selbst Google, das für seinen Minimalismus bekannt ist, verwendet mittlerweile CSS3-Eigenschaften, um die Gestaltung seines viel gelobten Suchformulars aufzuwerten.

Glücklicherweise lassen sich schon mit wenigen CSS-Regeln Formulare gestalten, die einen professionellen Eindruck vermitteln und sich in das Look-and-feel Ihrer Website einfügen. Dieses Kapitel gibt Ihnen die nötigen Techniken an die Hand, um direkt qualitativ hochwertige Formulare zu erstellen.

Eine der größten Schwierigkeiten bei der Gestaltung von Formularelementen ist, dass die verschiedenen Browser hier teilweise recht eigene Wege gehen, um sie darzustellen. Einige Browser lassen beispielsweise die Gestaltung von Checkboxes zu, während andere diese Stildefinitionen schlichtweg ignorieren.

In diesem Kapitel zeigen wir Ihnen, welche Stildefinitionen für HTML-Eingabeelemente wie Buttons, Textbereiche und -felder möglich sind, wie Sie benötigte Formularfelder hervorheben können und wie Sie auch mehrspaltige Formularlayouts umsetzen können. Am Ende des Kapitels finden Sie einen »Serviovorschlag«, der die wesentlichen hier behandelten Themen zusammenfassend behandelt.

7.1 Stile für Formulare definieren

Problem

Sie möchten ein Webformular mit eigenen Stildefinitionen versehen.

Lösung

Verwenden Sie den Elementselektor `form`, um alle Formulare auf einmal im Stylesheet auszuwählen. Um einzelne Formulare individuell zu gestalten, versehen Sie die Formulare mit einem `id`-Attribut, das Sie gegebenenfalls im Stylesheet ansprechen können. Hier ein Beispiel:

```
form {
    margin: 1em;
    padding: 1em;
    width: 30em;

    background: rgb(0,206,209) url('bilder/blauer-verlauf.jpg');
    border: 1px solid rgb(50,50,50);
}
```

In dieser Regel erhalten alle Formulare der Website (bzw. der aktuellen Seiten bei Verwendung eines internen Stylesheets) einen Innen- und Außenabstand von jeweils 1em Breite.

Damit sich die Formulare nicht über die gesamte Breite des Ansichtsbereichs erstrecken, begrenzen wir ihre Breite auf 30em. Als Hintergrund wird die Grafik eines blauen Farbverlaufs benutzt. Konnte die Grafik nicht geladen werden, wird das Formular in einem schlichten Blauton dargestellt.

Zum Abschluss erhalten die Formulare außerdem noch einen 1 Pixel breiten dunkelgrauen Rahmen. Das Ergebnis sehen Sie in Abbildung 7-1.



Auch wenn es in diesem Rezept in erster Linie um die Darstellung des `form`-Elements geht, haben wir uns die Freiheit genommen, ein wenig vorzugreifen. Für die Abbildung haben wir die Beschriftung der Eingabefelder (`label`) und die Texteingabefelder (`input`) als Blockelemente definiert und diese mit einer Breite von 15em versehen.

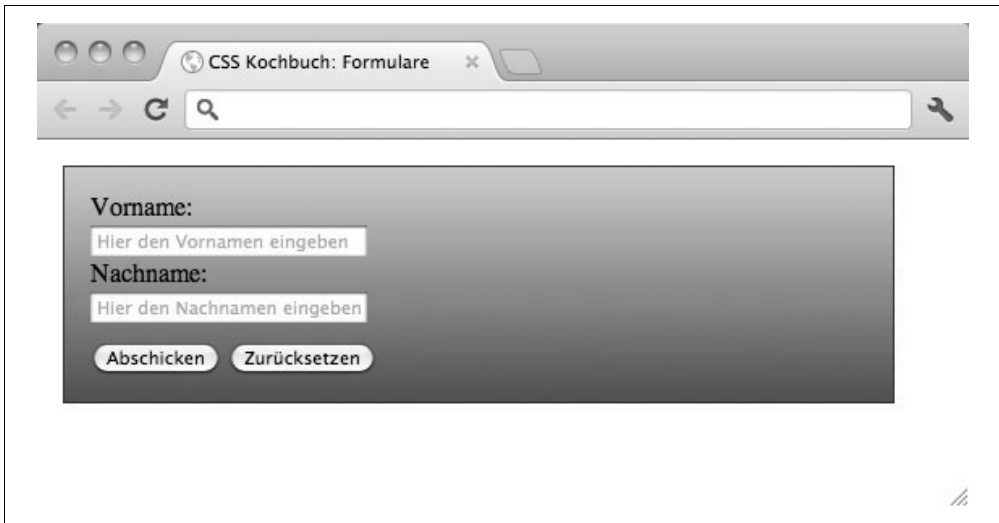


Abbildung 7-1: Stildefinitionen für ein Formular

Für diese Lösung haben wir den folgenden HTML-Code verwendet:

```
<form action="/adressen.php">

  <label for="vorname">Vorname:</label>
  <input type="text" id="vorname" name="vorname"
    placeholder="Hier den Vornamen eingeben" />

  <label for="nachname">Nachname:</label>
  <input type="text" id="nachname" name="nachname"
    placeholder="Hier den Nachnamen eingeben" />

  <button type="submit">Abschicken</button>
  <button type="reset">Zurücksetzen</button>

</form>
```

Diskussion

Formulare sind als Blockelemente definiert. Das heißt, Sie können sie mit dem gesamten Werkzeugkasten an CSS-Eigenschaften nach Belieben gestalten. In Bezug auf Stylesheets dient das `form`-Tag, ähnlich wie ein `div`-Element, hauptsächlich als Behälter für die einzelnen Formularelemente.

Die hier gezeigte Lösung bezieht sich nur auf das `form`-Element selbst. Die verschiedenen Eigenschaften können aber – sofern sie vererbt werden – Auswirkungen auf die enthaltenen Elemente haben (z.B. `font-family`). Seien Sie also vorsichtig.

Siehe auch

Die Rezepte 7.2 und 7.3 zur Gestaltung von gruppierten Formularelementen (fieldset) und zu Beschriftungen von gruppierten Formularelementen (legend); das Rezept 7.5 zur Definition von Stilen für Beschriftungen von einzelnen Formularelementen (label); die Rezepte 7.6 bis 7.12 zur Gestaltung einzelner Formularelemente.

7.2 Stile für fieldset-Elemente definieren

Problem

Sie haben verschiedene Formularelemente anhand des HTML-Elements fieldset gruppiert und wollen diese nun unabhängig vom umgebenden Formular gestalten.

Lösung

Gruppieren Sie die gewünschten Formularelemente, indem Sie sie mit fieldset-Tags umgeben. Danach können Sie den passenden Elementselektor fieldset benutzen, um die verschiedenen Darstellungsmerkmale selbst zu steuern:

```
fieldset {
    margin:      0;
    padding:     0.3em 0.5em 0.3em 0.5em;

    background:  rgb(164,202,218);
    background:  url("bilder/halbtransparent-weiss.png");
    background:  rgba(255,255,255,0.5);

    border:      1px solid rgb(50,50,50);
}

fieldset + fieldset {
    margin-top: 1.5em; /* Abstand zwischen zwei fieldsets */
}
```

Damit die Darstellung in den verschiedenen Browsern möglichst einheitlich aussieht, setzen wir zu Beginn den Außenabstand auf null zurück. Danach schaffen wir innerhalb der fieldset-Elemente ein wenig Platz, indem wir die Innenabstände mithilfe der Kurzschrifteigenschaft padding anpassen.

Es folgt die dreifache background-Deklaration, die Sie vielleicht schon aus Rezept 2.15 kennen. Sie soll die fieldset-Elemente mit einem weißen, halb transparenten Hintergrund versehen. Zum Abschluss umgeben wir die gruppierten Formularfelder noch mit einem dunkelgrauen dünnen Rahmen.

Nachdem wir den Außenabstand für fieldset-Elemente in der obigen Regel zurückgesetzt haben, würden die beiden Elementgruppen ohne Zwischenraum übereinanderste-

hen. Weil das nicht besonders schön aussieht, verwenden wir die zweite Regel (fieldset + fieldset). Sie schafft zwischen den beiden fieldsets einen Zwischenraum von 1.5em Breite (siehe Abbildung 7-2).

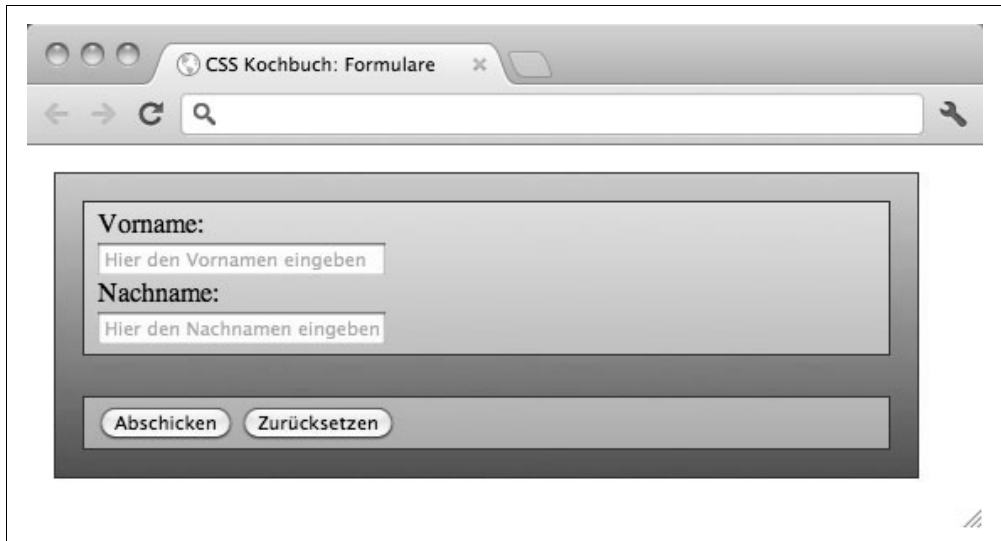


Abbildung 7-2: Spezielle Stile für gruppierte Formularelemente

Für die obige Darstellung haben wir den HTML-Code aus Rezept 7.1 wie folgt erweitert:

```
<form action="adreszen.php">

<fieldset>
<label for="vorname">Vorname:</label>
<input type="text" id="vorname" name="vorname" placeholder="Hier den Vornamen
eingeben" />

<label for="nachname">Nachname:</label>
<input type="text" id="nachname" name="nachname" placeholder="Hier den Nachnamen
eingeben" />
</fieldset>

<fieldset>
<button type="submit">Abschicken</button>
<button type="reset">Zurücksetzen</button>
</fieldset>

</form>
```

Diskussion

In diesem Beispiel haben wir die beiden Textfelder und die Buttons jeweils zu einer Gruppe zusammengefasst, um Benutzern die Orientierung im Formular zu erleichtern.

Um zwischen den beiden `fieldset`-Elementen Platz zu schaffen, haben wir einen Selektor für benachbarte Geschwisterelemente gewählt. Er wählt ein `fieldset`-Element aus, wenn es im HTML-Code direkt auf ein anderes `fieldset`-Element folgt. Durch diese Schreibweise wird ausgeschlossen, dass auch die erste Gruppe von Formularelementen den oberen Außenabstand erhält. (Mit den Beschriftungen für `fieldset`-Elemente beschäftigen wir uns im folgenden Rezept.)

Siehe auch

Die HTML5-Spezifikation zum `fieldset`-Element unter <http://www.w3.org/TR/html5/forms.html#the-fieldset-element>; das Grundrezept 2.6 zur Verwendung der Eigenschaft `padding`; das Rezept 13.13 zu Conditional Comments; das Rezept 7.3 zur Gestaltung von `legend`-Elementen; das Rezept 1.9 zu Selektoren für benachbarte Geschwisterelemente.

7.3 legend-Elemente mit Stildefinitionen versehen

Problem

Für die Beschriftung von Formularelementen, die per `fieldset` gruppiert wurden, gibt es das Element `legend`. Dieses soll mit eigenen Stildefinitionen versehen werden.

Lösung

Umgeben Sie den Beschriftungstext innerhalb der `legend`-Tags mit `span`-Tags. Danach definieren Sie die Stile mithilfe des passenden Elementselektors, wie hier gezeigt:

```
legend span {
  display: block;
  padding: 0.2em;

  background: #efefef;
  border: 1px solid rgb(50,50,50);

  font-family: Helvetica, Arial, sans-serif;
  font-size: smaller;
  white-space: normal; /* für Firefox */
}
```

Am Anfang dieser Regel wird das `span`-Element als Blockelement definiert. Das hilft uns, Abstände und Rahmen besser zu definieren. Per `padding: 0.2em` wird um den Beschriftungstext etwas Platz geschaffen.

Danach versehen wir die Beschriftung mit einem hellgrauen Hintergrund und einem dunkelgrauen Rahmen. Als Schriftfamilie haben wir hier eine serifenlose Variante gewählt, die kleiner dargestellt werden soll als der übrige Inhalt. Die letzte Deklaration (`white-space`) ist für Firefox wichtig, der auch lange Beschriftungstexte auf einer Zeile darzustellen versucht. Details hierzu finden Sie in der Diskussion.

Das fertige Formular inklusive Beschriftungen sehen Sie in Abbildung 7-3.

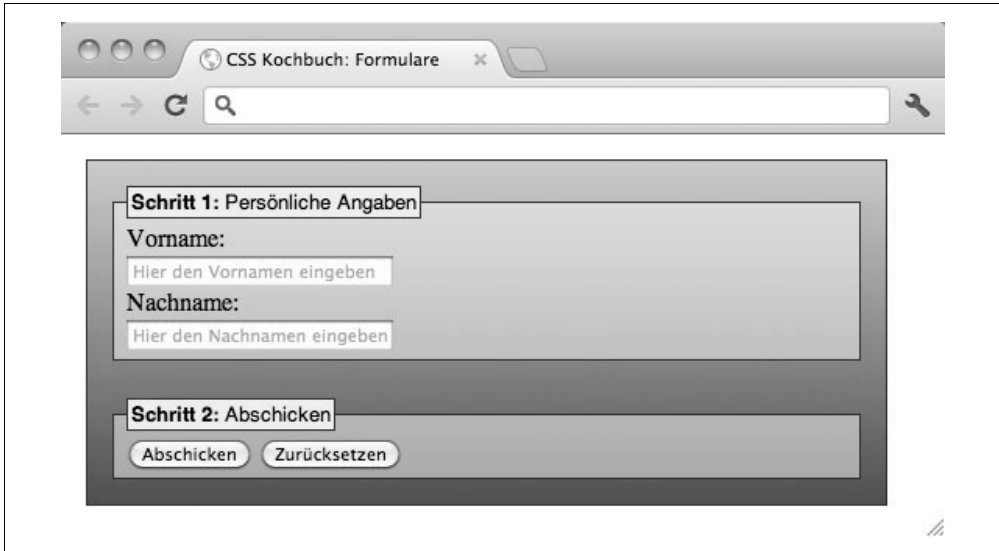


Abbildung 7-3: Die Beschriftung für gruppierte Formularelemente erhält eigene Stildefinitionen

Das gezeigte Formularlayout basiert auf dem folgenden HTML-Code:

```
<form action="adresse.php">

<fieldset>
<legend><span><strong>Schritt 1:</strong> Persönliche Angaben</span></legend>
<label for="vorname">Vorname:</label>
<input type="text" id="vorname" name="vorname" placeholder="Hier den Vornamen
eingeben" />

<label for="nachname">Nachname:</label>
<input type="text" id="nachname" name="nachname" placeholder="Hier den Nachnamen
eingeben" />
</fieldset>

<fieldset>
<legend><span><strong>Schritt 2:</strong> Abschicken</span></legend>
<button type="submit">Abschicken</button>
<button type="reset">Zurücksetzen</button>
</fieldset>

</form>
```

Diskussion

Zwar gilt `legend` offiziell als Blockelement, dennoch wird die Darstellung von den Browsern sehr unterschiedlich umgesetzt. So kann die Eigenschaft `margin` beispielsweise je nach Browser ganz unterschiedliche Auswirkungen haben. Sie können sich eine Menge

Ärger ersparen, wenn Sie sich zur Gewohnheit machen, den Beschriftungstext immer mit span-Tags zu umgeben und die allgemeinen Stile hierfür zu definieren.

Die Verwendung der span-Tags hilft außerdem, wenn Sie die Beschriftungen nicht an ihrer Standardposition darstellen wollen, sondern an einer anderen Stelle, wie in Rezept 7.4 erklärt wird.

Firefox und lange Beschriftungen

Sollte der Text Ihrer Beschriftung zu lang geraten, kann es passieren, dass er in Firefox über den Rand des fieldset hinausragt bzw. es auf unschöne Weise verbreitert, wie in Abbildung 7-4 gezeigt wird. Offenbar hat die Eigenschaft white-space in Firefox vor der Version 6 für legend-Elemente standardmäßig den Wert nowrap.

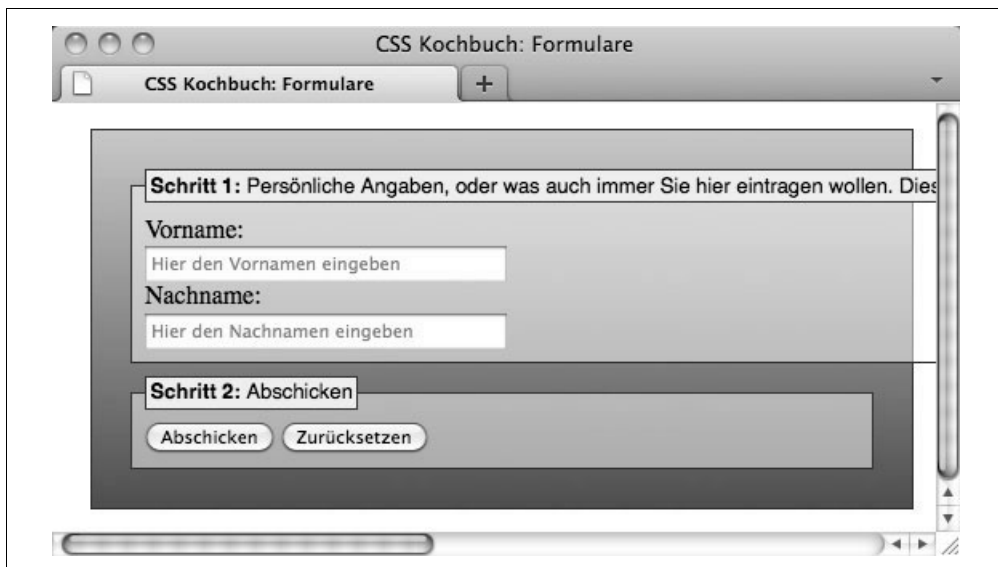


Abbildung 7-4: Ohne zusätzliche Definitionen ragen lange Beschriftungen in Firefox über das fieldset-Element hinaus

Internet Explorer und legend

IE 7 und 8 stellen die Beschriftungen innerhalb des von fieldset umgebenen Bereichs dar. Das fällt aber erst auf, wenn Sie für fieldset eine andere Hintergrundfarbe definieren als für das übrige Formular, wie in Abbildung 7-5 zu sehen ist.

Dieses als »Fieldset Background Color Bleed Bug« bekannte Problem lässt sich beheben, indem Sie die Beschriftungen mit einem negativen oberen Außenabstand versehen. Dadurch wird allerdings auch der Abstand des oberen fieldset zum Rand des Formulars kleiner. Um diese »Nebenwirkungen« auszugleichen, müssen Sie den oberen Außenabstand ebenfalls korrigieren.

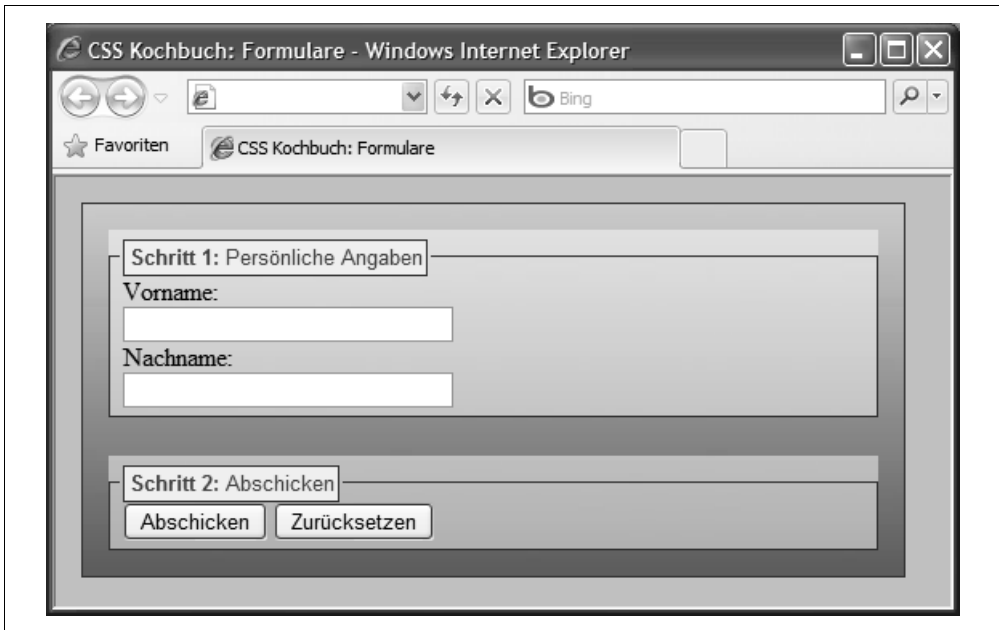


Abbildung 7-5: Der Hintergrund ragt in IE7 und 8 über den oberen Rahmen der fieldset-Elemente hinaus



Im Internet Explorer 9 kann der Fieldset Background Color Bleed Bug ausgelöst werden, wenn Sie versuchen, das fieldset-Element mit abgerundeten Ecken zu versehen.

Außerdem geben wir den Beschriftungen in IE7 und 8 etwas zusätzlichen unteren Außenabstand, um sie besser von den eigentlichen Formularfeldern zu trennen. Vorsichtshalber verpacken wir die Anweisungen in Conditional Comments:

```
<!--[if lte IE 8]>
<style type="text/css">
legend {
    margin-top:    -1.2em;
    margin-bottom: 0.5em;
}

form {
    padding-top: 1.2em;
}

legend span {
    color: black;
}
</style>
<![endif]-->
```

Die dritte Regel legt die Schriftfarbe für den Inhalt des legend-Elements fest, weil IE 7 und 8 den Wert für color nicht automatisch vererben. Der Wert inherit funktioniert hier nicht, daher geben wir die Farbe explizit an.

Internet Explorer 7

Leider hat der Internet Explorer in der Version 7 seine eigenen Vorstellungen davon, wie das Ganze auszusehen hat. Daher sind hier noch weitere Anpassungen notwendig. Die Darstellung entspricht nicht pixelgenau den anderen Browsern, aber für so ein altes Modell ist das Ergebnis ganz passabel:

```
<!--[if IE 7]>
<style type="text/css">
legend {
    margin-top: -1em;
    margin-left: -0.5em;

    overflow: hidden;
}
</style>
<![endif]-->
```

Hier ist der obere Außenabstand der legend-Elemente nicht ganz so hoch (wir haben auf eine eigene Anpassung von padding-top für das Formular verzichtet).

Im oberen Codebeispiel sehen Sie noch eine dritte Deklaration. Sie löst ein Problem, das bei der Verwendung von legend auftritt, wenn der Inhalt des fieldset-Elements (inklusive seitlicher Rahmen, Innen- und Außenabstände) breiter ist als das Formular. Haben Sie für das fieldset einen Rahmen definiert, kann es passieren, dass die obere Rahmenkante nach rechts verlängert wird. Das Problem kann auch auftreten, wenn Sie weder für das fieldset noch für die Inhalte eine feste Breite vergeben haben. Ein zu geringer Wert für width des form-Elements reicht aus.

Durch die Anweisung overflow: hidden wird die Rahmenkante (aber auch andere über das fieldset-Element hinausragende Inhalte) verborgen.

Siehe auch

Die HTML5-Spezifikation zum legend-Element unter <http://www.w3.org/TR/html5/forms.html#the-legend-element>; das Rezept 7.4 zur Positionierung des legend-Elements; den Kommentar von »Ross Angus« zur Verwendung von white-space für Firefox auf der Website von John Faulds unter <http://www.tyssendesign.com.au/articles/css/legends-of-style-revised/>; das Rezept 13.13 zur Verwendung von Conditional Comments; den Artikel *The IE Fieldset Background Color Bleed Bug* unter <http://www.communitymx.com/content/article.cfm?cid=DD9F3>.

7.4 Die Platzierung von legend-Elementen verändern

Problem

Sie möchten die Beschriftung für gruppierte Formularelemente an einer anderen Stelle platzieren, als standardmäßig vom Browser vorgesehen.

Lösung

Umgeben Sie den Beschriftungstext mit einem zusätzlichen `span`-Element und definieren Sie das umgebende `fieldset` als Positionierungskontext. Anstatt das `legend`-Element selbst zu positionieren, verwenden Sie das `span`-Element, um die Beschriftung nach Bedarf an die richtige Stelle zu verschieben. Wir legen hier kein vollkommen neues Stylesheet an, sondern verwenden stattdessen die Stildefinitionen aus Rezept 7.3 und passen sie bei Bedarf einfach an. Erweitern Sie zuerst die `fieldset`-Regel, um den nötigen Kontext für die Positionierung herzustellen:

```
fieldset {  
    position: relative;  
    padding: 0.3em 0.5em 0.3em 0.5em;  
    width: 15em;  
  
    background: rgb(164,202,218);  
    background: url("bilder/halbtransparent-weiss.png");  
    background: rgba(255,255,255,0.5);  
  
    border: 1px solid rgb(50,50,50);  
}
```

Der nächste Schritt besteht darin, das `span`-Element zu positionieren. Dafür passen wir die bereits vorhandene Regel entsprechend an:

```
legend span {  
    display: block;  
    position: absolute;  
  
    top: 0em;  
    left: 20em;  
  
    width: 10em;  
    padding: 0.2em;  
  
    background: #efefef;  
    border: 1px solid rgb(50,50,50);  
  
    font-family: Helvetica, Arial, sans-serif;  
    font-size: smaller;  
    white-space: normal; /* für Firefox */  
}
```

Per position: absolute wird das span-Element aus dem normalen Dokumentenfluss entfernt und kann anschließend mit den Eigenschaften top und left an die gewünschte Stelle gerückt werden. Wir legen ausdrücklich fest, dass die Oberkante der Beschriftung auf gleicher Höhe wie die Oberkante des fieldset-Elements dargestellt werden soll. Durch die Deklaration left: 20em erscheint die Beschriftung nun rechts neben dem fieldset, wie in Abbildung 7-6 gezeigt.

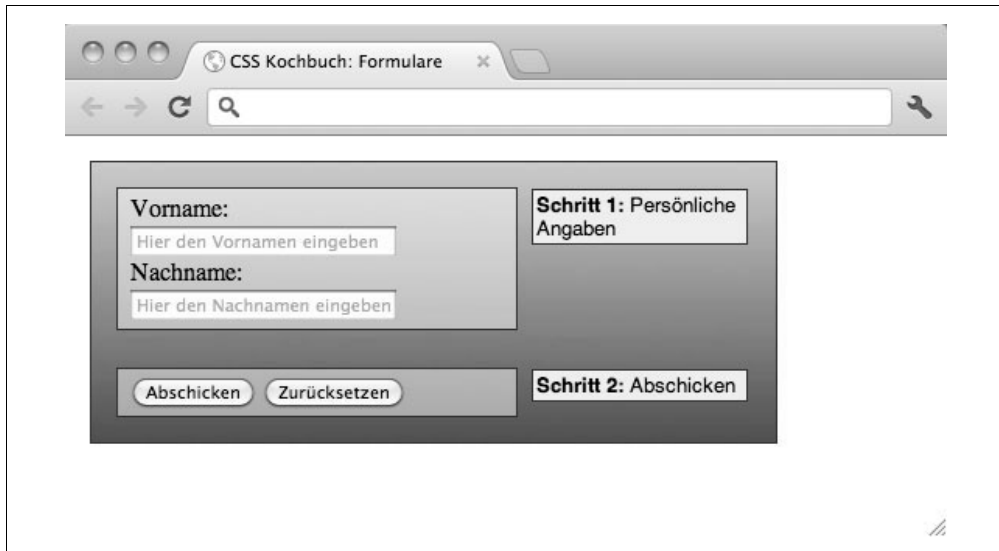


Abbildung 7-6: Die Beschriftungen werden rechts neben den fieldset-Elementen dargestellt



Wir haben das Formular aus ästhetischen Gründen für die Abbildung etwas schmaler gemacht (25.5em) als in den vorigen Rezepten.

Diskussion

Die verschiedenen Browser haben ihre ganz eigenen Vorstellungen davon, wo und wie legend-Elemente platziert werden. Man könnte versuchen, die gewünschten Änderungen mit der Eigenschaft margin umzusetzen. Allerdings sorgt das mehr für Verwirrung als für Ordnung.

Zum Glück hat der australische Webdesigner John Faulds einen Weg gefunden, dieses Problem zu umgehen. Der Grundgedanke besteht darin, das umgebende fieldset als Positionierungskontext für die Beschriftung zu verwenden. Leider kann das legend-Element bei der Annahme von Stildefinitionen ziemlich launisch sein.

Das Problem ist aber elegant zu umschiffen, wenn Sie den Inhalt des legend-Elements entsprechend der Empfehlung aus Rezept 7.3 mit einem zusätzlichen span-Element umgeben. Im Vergleich zu legend ist span geradezu pflegeleicht.

Gemäß der HTML-Spezifikation ist es nicht erlaubt, ein Blockelement innerhalb eines Inline-Elements zu platzieren. Daher kommt hier das `span`-Element zur Anwendung. Des-
sen Darstellungsrolle können wir anschließend per `display: block` ändern, wodurch uns
fast alle Gestaltungsmöglichkeiten von CSS zur Verfügung stehen. Dies ist einer der selte-
nen Fälle, in denen wir die Verwendung von nicht semantischem Markup ausdrücklich
empfehlen.

Internet Explorer vor der Version 9

Im Gegensatz zum Rezept zur Gestaltung des `legend`-Elements sind hier nur wenige
Anpassungen für den Internet Explorer nötig. Wir weisen dem `span`-Element seine eigene
Textfarbe zu und definieren `fieldset` noch einmal ausdrücklich als Blockelement. Die
zweite Anweisung ist eigentlich nur für den IE7 nötig, richtet aber im IE8 keinen Schaden
an. Daher reicht in diesem Fall ein Conditional Comment aus:

```
<!--[if lte IE 8]>
<style type="text/css">

    legend span {
        color: black;
    }

    display: block;
</style>
<![endif]-->
```

Siehe auch

Das Rezept 7.3 zur Gestaltung von `legend`-Elementen; die englische Wikipedia-Seite zu
HTML-Elementen und deren Darstellungsrollen unter http://en.wikipedia.org/wiki/HTML_element; die Website von John Faulds mit weiteren Informationen zur Positionie-
rung von `legend`-Elementen unter <http://www.tyssendesign.com.au/articles/css/legends-of-style-revised/>.

7.5 label-Elemente gestalten

Problem

Sie möchten spezielle Stile für das `label`-Element definieren, das zur Beschriftung von vie-
len Formularelementen dient.

Lösung

Verwenden Sie den Elementselektor `label`, um alle Beschriftungen von Formularfeldern
auf der gesamten Website einheitlich zu gestalten (bzw. auf der gesamten aktuellen Seite,
wenn Sie ein internes Stylesheet verwenden). Wenn Sie die Beschriftungen je nach Kon-

text verschieden gestalten wollen, können Sie den Wert des Attributs `for` (z.B. `<label for="nachname">`) als Unterscheidungsmerkmal nutzen. Alternativ dazu können Sie die Beschriftungen aber auch mit einem `class`-Attribut versehen, das sich auf den Eingabetyp (z.B. Texteingabe, Passwort usw.) bezieht.

```
label {  
    display: inline-block;  
    padding: 0.2em;  
    width: 10em;  
    height: 1.2em;  
  
    background: rgb(0,206,209) url('bilder/verlauf-grau.gif') repeat-x;  
    border: 1px solid rgb(50,50,50);  
  
    font-family: Helvetica, Arial, sans-serif;  
}  
  
label[for="nachname"] {  
    margin-top: 0.2em;  
}
```

In der ersten Regel erhalten die Beschriftungen die Darstellungsrolle `inline-block`. Dadurch ist es möglich, ihnen eine feste Breite (hier 10 em-Einheiten) und Höhe (1.2 em-Einheiten) zuzuweisen, ohne dass nach dem Element ein Zeilenumbruch erzeugt wird. Es folgen die Deklarationen für Hintergrund (ein grauer Farbverlauf), Rahmen und Schrift.

Die zweite Regel verwendet einen einfachen Attributselektor (`label[for="nachname"]`), um für die zweite Beschriftung einen oberen Außenabstand zu definieren.

Die Beschriftung hat eine Breite von 10 em-Einheiten, während das Textfeld in dieser Abbildung 15 em-Einheiten breit ist. Dadurch passt immer ein Paar aus Beschriftung und Eingabefeld in eine Zeile des insgesamt 30em breiten Formulars. Das Ergebnis sehen Sie in Abbildung 7-7.



Abbildung 7-7: Die `label`-Elemente erhalten eigene Stildefinitionen

Diskussion

Die Gestaltung der label-Elemente ist deutlich einfacher zu handhaben als bei legend-Elementen. Durch die Verwendung der richtigen Darstellungsrolle stehen Ihnen fast alle Möglichkeiten offen, die CSS bietet.



Anstatt das label-Element als für sich allein stehende Einheit zu verwenden, können Sie es auch nutzen, um input- und andere Formularelemente damit zu umgeben. Das kann unter Umständen die Definition zusätzlicher Tags unnötig machen.

Siehe auch

Das Rezept 7.6 zur Gestaltung von Texteingabefeldern; das Rezept 1.11 zur Verwendung des Attributselektors; die HTML5-Spezifikation zum label-Element unter <http://www.w3.org/TR/html5/forms.html#the-label-element>.

7.6 Texteingabefelder mit CSS gestalten

Problem

Sie möchten die Standarddarstellung der Browser für Texteingabefelder verändern, wie in Abbildung 7-8 gezeigt.



Abbildung 7-8: Die Textfelder erhalten eigene Stile

Lösung

Verwenden Sie einen passenden Attributselektor, um das Eingabefeld von anderen input-Elementen zu unterscheiden. Achten Sie bei der Gestaltung darauf, dass nicht alle Browser sämtliche CSS-Eigenschaften unterstützen. Hier ein Beispiel:

```
input[type="text"] {
    width: 15em;

    margin-top: 0.2em;
    margin-bottom: 0.2em;
    font-size: 1em;

    background-color: #fffacd;
    outline: 1px solid rgb(50,50,50);
}
```

Hier erhalten alle Texteingabefelder (`input type="text"`) eine Breite von 15 em-Einheiten. Ober- und unterhalb schaffen wir mithilfe von `margin-top` bzw. `margin-bottom` ein wenig Platz.

Damit die Breiten- und Höhenangaben in em-Einheiten für Textfelder und deren Beschriftungen die gleichen Auswirkungen haben, vergeben wir außerdem einen ausdrücklichen Wert für `font-size`. Zum Schluss erhalten die Eingabefelder noch eine eigene Hintergrundfarbe und einen dunkelgrauen Rahmen.

Da die meisten Browser die Eigenschaft `border` verwenden, um den typischen »Textbox-Look« zu erzeugen, den wir hier erhalten wollen, weichen wir auf die eher selten benutzte Eigenschaft `outline` aus. Leider wird `outline` vom Internet Explorer 7 nicht unterstützt. Sie sollten Ihr Design also nicht davon abhängig machen.

Diskussion

Bei der Gestaltung von Textfeldern gibt es ein paar Dinge zu beachten. So beziehen sich die Werte für `padding` und `border` auf das Innere des Textfelds. Eine Anweisung wie `padding: 0.3em` schafft also zusätzlichen Platz um den eingegebenen Text herum. Wenn Sie außerhalb des Eingabefelds Leerraum erzeugen wollen, verwenden Sie daher besser die Eigenschaft `margin`.

Die meisten Browser benutzen die Eigenschaft `border`, um dem Textfeld den Eindruck einer gewissen »Tiefe« zu verleihen. Das können Sie durch eigene Werte für `border` natürlich komplett ändern. Kombiniert mit den CSS3-Eigenschaften für Schattenwürfe, ließe sich der 3-D-Effekt sogar noch verstärken, indem Sie schreiben:

```
input[type="text"] {
    border: 0;
    -moz-box-shadow: 1px 1px 3px black inset;
    -webkit-box-shadow: 1px 1px 3px black inset;
    box-shadow: 1px 1px 3px black inset;
}
```

Hier wird der standardmäßige Rahmen deaktiviert und durch CSS3-Schattenwürfe ersetzt. Sie sollten jedoch bedenken, dass nicht alle aktuellen Browser die Eigenschaft `box-shadow` verstehen. Damit ältere Browser nicht nackt dastehen, sollten Sie auf jeden Fall mit einem Polyfill, z.B. *CSS3 PIE*, arbeiten. Ohne weitere Maßnahmen fehlt der Rahmen sonst komplett.

Aus den genannten Gründen ist es oftmals besser, auf die Eigenschaft `outline` auszuweichen, wenn Sie um das Textfeld herum einen eigenen Rahmen darstellen möchten.



Die Angaben in `em`-Einheiten beziehen sich auf den Wert von `font-size` für das betreffende Element. Da viele Browser den Text in Eingabefeldern kleiner darstellen als die Standardschriftgröße, kann es passieren, dass die Angabe `width: 15em` für Textfelder etwas anderes bedeutet als für deren Beschriftungen. Sie können Abhilfe schaffen, indem Sie, wie in unserer Lösung, die Textgröße für die betroffenen Elemente ausdrücklich mit dem gleichen Wert versehen.

Neue HTML5-Elemente

Die HTML5-Spezifikation definiert eine Reihe neuer `input`-Elemente für bestimmte Datentypen wie URLs, E-Mail-Adressen, Datumsangaben usw. Zur Drucklegung dieses Buchs wurden diese Felder längst nicht von allen aktuellen Browsern (außer Opera 11+) unterstützt. Daher können Sie die hier gezeigte Lösung bis auf Weiteres ohne Probleme auch für diese `input`-Elemente verwenden.

Im Abschnitt »Siehe auch« finden Sie einen Link auf einen Artikel von Zoltan Dulac, in dem erklärt wird, wie Sie mithilfe verschiedener Polyfills auf die Unterstützung für HTML5-Formularelemente testen und gegebenenfalls nachrüsten können.

Safari 5.1+

Safari hat ab der Version 5.1 die Angewohnheit, Textfelder, die vom Browser automatisch ausgefüllt wurden, hellgelb zu hinterlegen. Um das zu verhindern, können Sie für die Safari-eigene Präfixeigenschaft `-webkit-autofill` einen eigenen Farbwert definieren. Beachten Sie aber, dass es sich hierbei nicht um eine »offizielle« CSS-Eigenschaft handelt.

Siehe auch

Die HTML5-Spezifikation zum `input`-Element unter <http://www.w3.org/TR/html5/the-input-element.html>; die Quirksmode-Website zur Browserunterstützung für die neuen `input`-Typen in HTML5 unter <http://www.quirksmode.org/html5/inputs.html>; den Artikel *Creating Cross Browser HTML5 Forms Now, Using modernizr, webforms2 and html5Widgets* von Zoltan Dulac unter <http://www.useragentman.com/blog/2010/07/27/cross-browser-html5-forms-using-modernizr-webforms2-and-html5widgets/>; die Website html5please.us zur Unterstützung und für Ausweichlösungen für HTML5-Formularelemente unter <http://html5please.us/#input>.

7.7 Stildefinitionen für textarea-Elemente zuweisen

Problem

Mithilfe von Stildefinitionen sollen die verschiedenen Darstellungsmerkmale eines text-area-Elements verändert werden (siehe Abbildung 7-9).

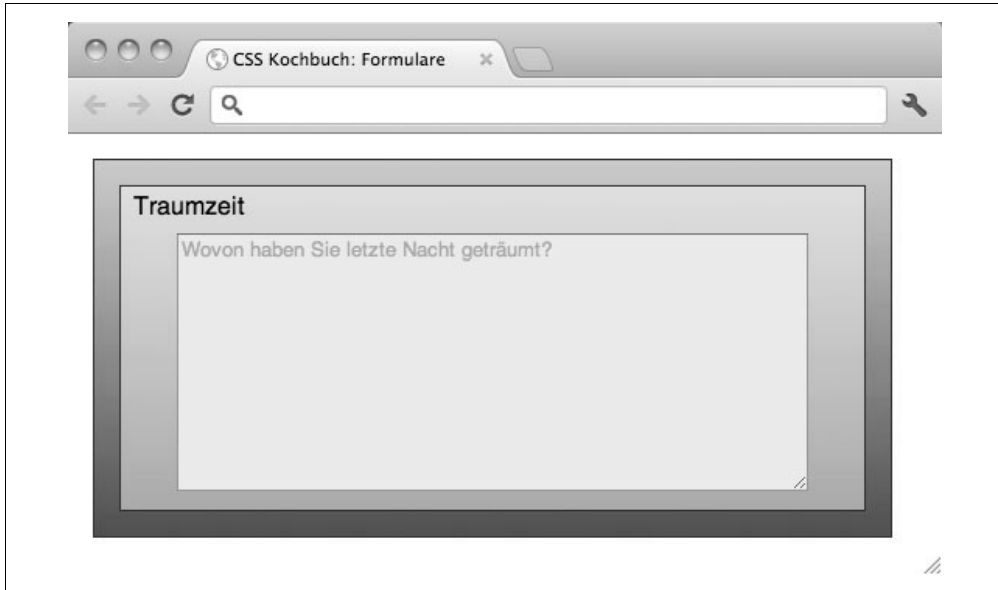


Abbildung 7-9: Ein textarea-Element mit eigenen Stildefinitionen

Lösung

Weisen Sie dem textarea-Element die Stile mithilfe eines Elementselektors zu:

```
textarea {  
  display: block;  
  margin: 0.5em auto;  
  
  width: 25em;  
  height: 10em;  
  
  overflow-y: auto;  
  
  font-size: 1em;  
  font-family: Helvetica, Arial, sans-serif;  
  
  background-color: #eeeecc;  
}
```


Mit der ersten Deklaration wird festgelegt, dass Textbereiche als Blockelemente dargestellt werden sollen. Dadurch können wir die Eigenschaft `margin` benutzen, um das Textfeld horizontal im umgebenden `fieldset` zu zentrieren. Außerdem schaffen wir ober- und unterhalb gleich noch etwas Platz und geben dem Textbereich per `width` und `height` eine feste Höhe und Breite.



Diese Angaben für `width` und `height` überschreiben Werte, die Sie für die Attribute `cols` und `rows` möglicherweise im HTML-Code für das `textarea`-Element angegeben haben.

Die folgenden beiden Deklarationen (`font-size` und `font-family`) legen fest, dass für die Eingaben die Standardschriftgröße des Browsers verwendet werden soll, als Schriftfamilie geben wir Helvetica, Arial oder eine andere serifenlose Schrift an.

Zum Schluss erhält der Textbereich noch eine eigene Hintergrundfarbe.

Diskussion

Wie Sie im Lösungscode gesehen haben, ist es mit Elementselektoren recht einfach, Stile für ein `textarea`-Element zuzuweisen.

Ähnlich wie bei Texteingabefeldern (`input type="text"`) verhalten sich auch bei `textarea` einige CSS-Eigenschaften nicht unbedingt wie erwartet. Die Eigenschaft `padding` wirkt beispielsweise innerhalb des Eingabebereichs und steuert den Abstand zwischen eingegebenem Text und den Rändern des Elements.

Internet Explorer und Scrollbalken

Der Internet Explorer versieht den Textbereich auch dann mit vertikalen Scrollbalken, wenn dies eigentlich nicht nötig ist.

Dieses Verhalten können Sie durch die Deklaration `overflow-y: auto` ändern. Anderen Browsern schadet diese Anweisung nicht, daher steht sie hier zusammen mit den übrigen Anweisungen für das `textarea`-Element.

Gecko und Webkit

In Gecko- und Webkit-basierten Browsern ist es möglich, den mit dem `placeholder`-Attribut definierten Text speziell zu gestalten.



Der Text des Attributs `placeholder` wird übrigens auch in Passwortfeldern im Klartext angezeigt

Verwenden Sie hierfür in Gecko-Browsern (z.B. Firefox) die Präfixpseudoklasse `::-moz-placeholder` bzw. in Webkit (Safari, Chrome) das Präfixpseudoelement `::-webkit-input-placeholder`, z.B.:

```

textarea::-webkit-input-placeholder {
    font-size: smaller;
}

textarea::-moz-placeholder {
    font-size: smaller;
}

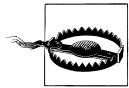
```

Die beiden Selektoren können nicht gruppiert werden. Folgendes funktioniert nicht:

```

textarea::-webkit-input-placeholder, textarea:moz-placeholder {
    color: red;
}

```



Diese browsereigenen Erweiterungen sind nicht Teil einer CSS-Spezifikation. Eine Überprüfung, z.B. durch einen Onlinevalidierungsdienst, wird fehlschlagen.

Diese Präfixselektoren sind nicht nur auf Textfelder beschränkt, sondern Sie können sie auch für andere input-Elemente (z.B. `input:-webkit-placeholder`) benutzen.

Siehe auch

Das Rezept 1.14 zu dynamischen Pseudoklassen; das Rezept 1.6 zu Elementselektoren; das Rezept 1.16 zum Boxmodell; die CSS3-Spezifikation zu dynamischen Pseudoklassen unter <http://www.w3.org/TR/selectors/#dynamic-pseudos>; die CSS3-Spezifikation zum Thema Elementselektoren (bzw. Typselektoren) unter <http://www.w3.org/TR/selectors/#type-selectors>; das Box-Model-Modul der CSS3-Spezifikation zu den Eigenschaften `overflow-x` und `overflow-y` unter <http://www.w3.org/TR/css3-box/#overflow1>.

7.8 Stile für Checkboxes und Radiobuttons definieren

Problem

Sie möchten Checkboxes (`type="checkbox"`) und Radiobuttons (`type="radio"`) mit eigenen Stilen versehen.

Lösung

Versuchen Sie nicht, die Checkboxes und Radiobuttons direkt per CSS zu verändern. Das funktioniert in den meisten Fällen nicht wie erwartet. Wenn es wirklich sein muss, umgeben Sie das input-Element besser mit `span`-Tags, für die Sie dann Ihre Stildefinitionen vornehmen können. Die `span`-Tags erhalten in dieser Lösung zwei Werte für das `class`-Attribut: eines, auf das sich auf den Inhalt bezieht, und eines für das verwendete Element. Zum Beispiel:

```

<label for="zitrone">Zitroneneis</label>
  <span class="zitrone checkbox"><input name="zitrone" id="zitrone" type="checkbox" />
</span>

<label for="erdbeer">Erdbeereis</label>
  <span class="erdbeere checkbox"><input name="erdbeere" id="erdbeere" type="checkbox" />
</span>

```

Im Stylesheet können Sie Folgendes dann schreiben:

```

span.checkbox {
  display: inline-block;
  margin: 0.2em;
  padding: 0.2em;
  border: 1px solid #000;
}

span.zitrone {
  background-color: yellow;
}

span.erdbeere {
  background-color: red;
}

```

Zunächst wird eine allgemeine Regel für die Gestaltung der Checkboxes definiert. Danach verwenden wir den zweiten Wert des `class`-Attributs, um diese Angaben je nach Geschmacksrichtung zu verfeinern.

Diskussion

Checkboxes und Radiobuttons gelten als *ersetzte* Elemente. Das heißt, der Browser selbst hat höchstens eine grobe Vorstellung davon, wie sie dargestellt werden sollen. Oftmals »borgt« sich der Browser diese Elemente vom Betriebssystem aus und fügt sie in die Darstellung ein.

Dadurch ist es nur schwer vorhersehbar, wie Checkboxes oder Radiobuttons am Ende wirklich aussehen. Das macht eine Gestaltung mit CSS nahezu unmöglich.

Deshalb sollten Sie die Elemente selbst nach Möglichkeit in Ruhe lassen. Wenn Sie unbedingt ein Hintergrundbild, einen Rahmen oder andere Dinge verwenden möchten (oder müssen, weil Ihr Auftraggeber darauf besteht), sind Sie meistens besser bedient, wenn Sie das Element mit zusätzlichen `span`-Tags umgeben und diese dann gestalten.¹

¹ Noch besser geht das oft mit einem JavaScript-Framework wie jQuery. Am allerbesten ist es jedoch, wenn Sie es schaffen, Ihrem Auftraggeber diese Idee auszureden.

Siehe auch

Das Rezept 1.11 zur Verwendung von Attributselektoren; das Rezept 1.17 zu ersetzen Elementen; die CSS3-Spezifikation zur Verwendung von Attributselektoren unter <http://www.w3.org/TR/selectors/#attribute-selectors>; die HTML5-Spezifikation zu input-Elementen unter <http://www.w3.org/TR/html5/the-input-element.html>.

7.9 Stildefinitionen für Auswahllisten und Aufklappmenüs

Problem

Farbe und Schrift von Auswahllisten und Aufklappmenüs (select-Elementen) und den enthaltenen Einträgen (option-Elementen) sollen an die eigenen Bedürfnisse angepasst werden (siehe Abbildung 7-10).

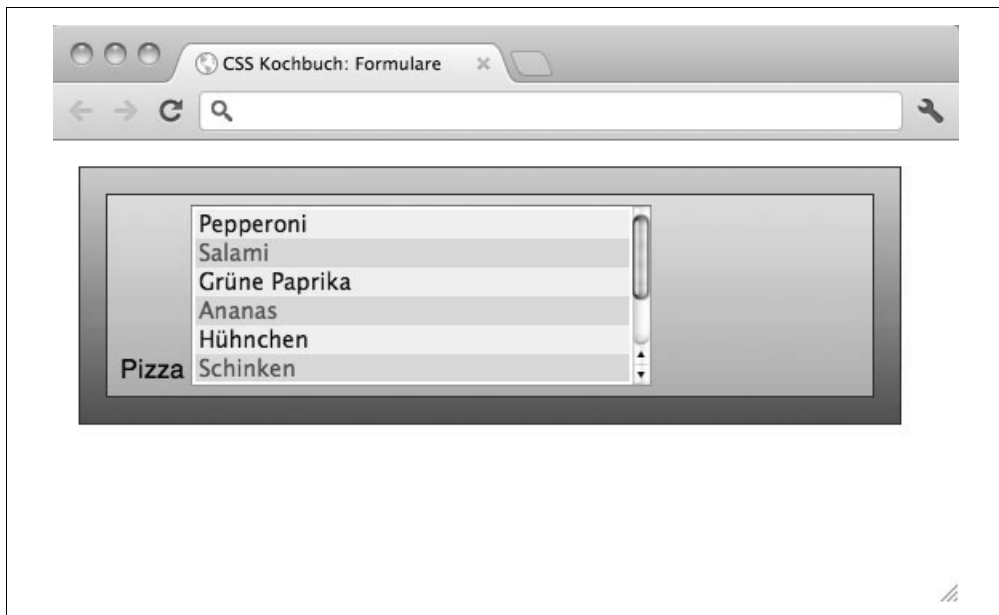


Abbildung 7-10: Auswahlliste mit zugewiesenen Stildefinitionen

Lösung

Verwenden Sie einen Elementsektor, um option- und select-Elementen die gewünschten Stildefinitionen zuzuweisen:

```
select {  
  width:      20em;  
  padding:    0.2em;
```

```

font-size:      0.9em;
color:          #111;
}

select:focus {
  background-color: lightgreen;
}

option {
  background-color: #efefef;
}

```

Die Auswahlliste erhält eine Breite von 20 em-Einheiten und einen Innenabstand von 0.2em. Beide Werte beziehen sich auf den direkt danach definierten Wert für `font-size`, der hier etwas kleiner gewählt wurde als die Standardschriftgröße des Browsers. Als Schriftfarbe wählen wir hier einen dunklen Grauton.

Um die Auswahlliste zusätzlich hervorzuheben, wenn der Benutzer etwas eingibt, erhält sie eine hellgrüne Hintergrundfarbe, sobald sie den Fokus hat.

Zum Schluss erhalten die einzelnen `option`-Elemente noch einen hellen Grauton als Hintergrundfarbe.

Diskussion

Im Gegensatz zu Texteingabefeldern wird bei Aufklappmenüs und Auswahllisten in beiden Fällen das `select`-Element zur Definition verwendet. Daher ist in diesem Fall eine Zuweisung von Stilen nicht so schwer. Das Gleiche gilt für `option`-Elemente.

Wollen Sie den Optionen in einer Auswahlliste abwechselnd unterschiedliche Stile zuweisen, können Sie hierfür in modernen Browsern die CSS3-Pseudoklasse `:nth-child()` mit dem Parameter `even` einsetzen:

```

option:nth-child(even) {
  background-color: #dda;
}

```

Durch diese Regel können Sie jedes zweite `option`-Element mit einem eigenen Hintergrund versehen. Für ältere Browser funktioniert diese Methode leider nicht. Die Pseudoklasse `:nth-child()` sollte daher nur im Sinne des »Progressive Enhancement« eingesetzt werden. Wenn Sie die Auswahllemente unterschiedlich gestalten müssen, ist es immer noch sicherer, jedes zweite `option`-Element mit einem speziellen `class`-Attribut zu versehen:

```

<select name="belag" size="6" multiple>
  <option value="1">Peperoni</option>
  <option value="2" class="even">Salami</option>
  <option value="3">Grüne Paprika</option>
  <option value="4" class="even">Ananas</option>
  <option value="5">Hühnchen</option>
  <option value="6" class="even">Schinken</option>
  <option value="7">Oliven</option>

```

```

<option value="8" class="even">Zwiebeln</option>
<option value="9">Rote Paprika</option>
</select>

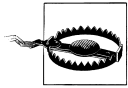
```

Damit das funktioniert, verwenden Sie anstelle von `nth-child` einen schlichten Klassen-selektor:

```

option.even {
    background-color: #dda;
}

```



Die Gruppierung beider Regeln funktioniert in diesem Fall nicht. Unterstützt ein Browser die Pseudoklasse `:nth-child()` nicht, wird die gesamte Regel verworfen. Daher müssen Sie hier zwei separate Regeln verwenden, um auch ältere Browser mit einzubinden.

Die abwechselnde Gestaltung mithilfe von `class`-Attributen funktioniert auch in Internet Explorer 7 und 8.

Siehe auch

Das Rezept 1.14 zur Verwendung der Pseudoklasse `:nth-child()`; das Rezept 1.12 zur Verwendung von Klassenselektoren.

7.10 Stildefinitionen für Buttons

Problem

Für die Submit- und Reset-Buttons eines Formulars sollen Farbe, Innenabstände, Rahmen und Rollover-Effekte definiert werden, wie in Abbildung 7-11 gezeigt wird.

Lösung

Verwenden Sie anstelle von `<input type="submit">` bzw. `<input type="reset">` das spezielle HTML-Element `button`, um eine möglichst konsistente Darstellung zu erreichen.

```

<fieldset>
  <button type="submit">Abschicken</button>
  <button type="reset">Zurücksetzen</button>
</fieldset>

```

Danach können Sie die Buttons mit dem Elementselektor `button` auswählen und fast nach Belieben mit Stilen versehen, wie hier gezeigt:

```

button {
    height: 3em;
    width: 10em;

    background: #fff url("bilder/verlauf-grau.gif") repeat-x;
    border: 1px solid black;
}

```

```

font-size: 1em;
font-weight: bold;
}

button:hover {
border-color: orange;
}

button + button {
margin-left: 0.5em;
}

```

Zuerst erhalten die Buttons, unabhängig davon, ob es sich um Submit- oder Reset-Buttons handelt, eine feste Höhe und Breite. Als Hintergrund verwenden wir hier eine Grafik.

Die letzten beiden Deklarationen in der ersten Regel legen fest, dass die Beschriftung des Buttons in der Standardschriftgröße des Browsers erscheinen soll. Außerdem wird der Text fett formatiert.

Die zweite Regel definiert über die Rahmenfarbe einen Rollover-Effekt. Das Ergebnis sehen Sie in Abbildung 7-11. Mit der dritten Regel legen wir ausdrücklich fest, dass der linke Außenabstand des zweiten Buttons (und damit die horizontale Entfernung zwischen beiden Buttons) ein halbes em betragen soll.



Abbildung 7-11: Zwei Buttons wurden mit Stildefinitionen versehen

Diskussion

Ohne weitere Angaben werden Buttons von den Browsern als ersetzte Elemente behandelt. Das heißt, der HTML-Code `button` wird in der Darstellung der Seite durch einen »echten« Button ausgetauscht. Je nach Browser und Betriebssystem kann es hierbei erhebliche Unterschiede geben.

Sobald Sie das `button`-Element jedoch mit Stildefinitionen versehen, ändert sich die Situation, und es sind erstaunlich konsistente und vielseitige Designs möglich.

Hierbei gilt: Je genauer Sie Ihre Angaben machen, desto einheitlicher ist die Darstellung in den verschiedenen Browsern. Aus diesem Grund wurde in der Lösung die dritte Regel (`button + button`) definiert. Sie sorgt dafür, dass der Abstand zwischen den beiden Buttons immer gleich ist – selbst im Internet Explorer 7.

Das Gleiche gilt für den Leerraum zwischen der Beschriftung und der Kante des Buttons. Ohne ausdrückliche Angaben verwendet jeder Browser hier seinen Standardwert. Der Schlüssel ist hier jedoch nicht die Eigenschaft `padding`, wie man denken könnte. Stattdessen verwendet der Internet Explorer einen anderen Wert für `font-size` als Gecko- oder Webkit-basierte Browser (Firefox bzw. Safari/Chrome).

Sobald Sie den Wert für `font-size` explizit angeben, ist die Welt wieder in Ordnung.

Neben der Pseudoklasse `:hover` funktioniert in den meisten Browsern übrigens auch `:active`. Um den Rahmen des Buttons, während er angeklickt wird, rot einzufärben, könnten Sie beispielsweise schreiben:

```
button:active {  
    border-color: red;  
}
```

Eigene Stile für verschiedene Buttons

Wenn Sie die Buttons abhängig von ihrer Funktion (*Abschicken/Zurücksetzen*) speziell gestalten wollen, können Sie die Unterscheidung beispielsweise anhand des `type`-Attributs vornehmen.

```
button[type="submit"] {  
    background-image: url('bilder/ok-sm.png');  
}  
  
button[type="reset"] {  
    background-image: url('bilder/reset-sm.png');  
}
```

Die übrigen Deklarationen sind für beide Buttons gleich und können daher für das `button`-Element gemeinsam angegeben werden:

```
button {  
    background-position: left center;  
    background-repeat: no-repeat;  
  
    text-align: left;  
    text-indent: 40px;  
}
```

Hier verwenden wir Attributselektoren, um den Buttons jeweils ein eigenes Symbol zuzuweisen. Damit es nicht vom Beschriftungstext überlagert wird, schieben wir ihn per `text-indent` um die Breite des Bilds nach rechts. Das Ergebnis sehen Sie in Abbildung 7-12.

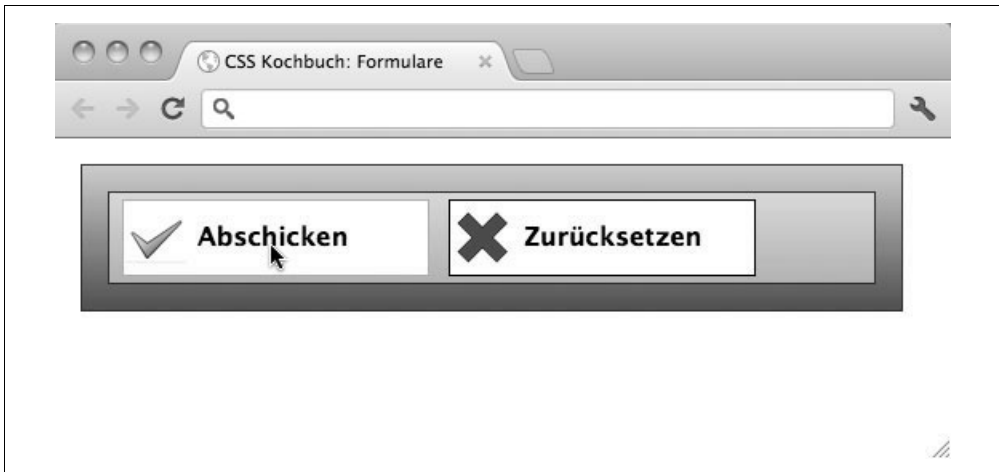


Abbildung 7-12: Mithilfe von Attributselektoren können Sie den Buttons jeweils ein eigenes Symbol zuweisen

Siehe auch

Das Rezept 1.11 zur Verwendung von Attributselektoren; das Rezept 1.17 zu ersetzten Elementen; die CSS3-Spezifikation zur Verwendung von Attributselektoren unter <http://www.w3.org/TR/selectors/#attribute-selectors>; das Rezept 1.9 zu Selektoren für benachbarte Geschwisterelemente; das Rezept 7.11 zur Gestaltung von grafikbasierten Buttons.

7.11 Grafikbasierte Buttons

Problem

Eine Grafik soll als Button verwendet werden, wie in Abbildung 7-13 gezeigt.

Lösung

Umgeben Sie den Beschriftungstext mit einem `span`-Element, das Sie anschließend im Stylesheet mit der Eigenschaft `text-indent` (Off-Left-Technik) aus dem Ansichtsbereich verschieben können. Die gewünschten Grafiken definieren Sie als Hintergrundbilder für die `button`-Elemente, wie hier gezeigt:

```
button {  
    height: 45px;  
    width: 45px;  
  
    background-color: #fff;  
    border: 1px solid black;
```

```

    font-size: 1em;
}

button span {
    display: inline-block;
    text-indent: -9999em;
}

button[type="submit"] {
    background: url('bilder/ok-sm.png') center center no-repeat;
}

button[type="reset"] {
    background-image: url('bilder/reset-sm.png') center center no-repeat;
}

```

In der ersten Regel erhalten die Buttons die gleiche Höhe und Breite. Damit die Grafiken etwas Platz bekommen, haben wir Werte gewählt, die etwas über der Bildgröße von 40 × 40 Pixeln liegt.

Die folgenden Deklarationen sorgen dafür, dass die Grafiken vor einem weißen Hintergrund erscheinen und von einem einfachen schwarzen Rahmen umgeben werden.

Die letzte Deklaration der ersten Regel (`font-size: 1em`) dient als »Maßstab« für die Größenangaben. Hierdurch erreichen wir eine größere Konsistenz zwischen den verschiedenen Browsern.

Die folgenden beiden Regeln verwenden Attributselektoren, um für die Buttons jeweils ein anderes Hintergrundbild zu definieren. In beiden Fällen wird das Bild in der Mitte des umgebenden Blocks (hier das `button`-Element) verankert und nicht wiederholt.

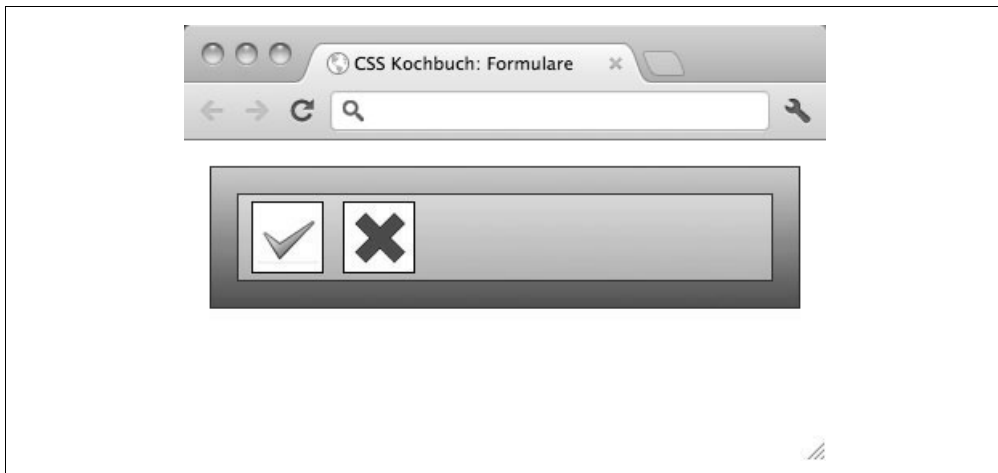


Abbildung 7-13: Die Standard-Buttons wurden durch Grafiken ersetzt

Diskussion

Lange Zeit war es üblich, zur Erzeugung von grafischen Buttons input-Elemente mit dem type-Attribut image zu verwenden. Bei dieser Methode ist es jedoch nicht ohne Weiteres möglich, einen alternativen Beschriftungstext anzugeben, falls das Formular beispielsweise auf einer Braillezeile oder mit einem Screenreader ausgegeben werden soll.

Auch die Möglichkeit, zwischen den Button-Tags ein zusätzliches img-Element mit der gewünschten Grafik einzufügen, ist nicht optimal, weil sie Inhalt und Darstellung unnötig vermischt.



Wenn Sie auf den Internet Explorer keine Rücksicht zu nehmen brauchen, können Sie die span-Elemente auch weglassen und die Eigenschaft text-indent direkt für die button-Elemente definieren.

Siehe auch

Die HTML5-Spezifikation zum button-Element unter <http://www.w3.org/TR/html5/the-button-element.html>; das Rezept 1.11 zur Verwendung von Attributselektoren; das Rezept 4.6 zur Verwendung der »Off-Left«-Methode zur Textersetzung.

7.12 Buttons wie einfachen HTML-Text darstellen

Problem

Ein button-Element soll so dargestellt werden, dass es wie normaler HTML-Text aussieht (siehe Abbildung 7-14).



Abbildung 7-14: Die button-Elemente werden wie normaler HTML-Text dargestellt

Lösung

Durch den Einsatz von verschiedenen CSS-Formatierungseigenschaften kann die gewünschte Darstellung als HTML-Text erreicht werden. Zusätzlich zum Elementselektor `button` können Sie die dynamischen Pseudoklassen `:hover` und `:active` sowie einen passenden Wert für die Eigenschaft `cursor` einsetzen, um das Verhalten der Buttons noch mehr an das von Links anzupassen:

```
button {  
    border: none;  
    background: none;  
  
    font-size: inherit;  
    font-family: inherit;  
    text-decoration: underline;  
}  
  
button:hover {  
    color: blue;  
    cursor: pointer;  
}  
  
button:active {  
    color: red;  
}
```

Die ersten beiden Deklarationen sorgen dafür, dass Rahmen und Hintergrund von den Buttons entfernt werden. Dadurch erscheint die Beschriftung direkt vor dem Hintergrund des Formulars.

Im nächsten Schritt verwenden wir den Wert `inherit`, um anzugeben, dass Schriftgröße (`font-size`) und -familie (`font-family`) vom übergeordneten Element geerbt werden sollen, für das diese Werte definiert wurden. Danach erhalten die Beschriftungen per `text-decoration: underline` die linktypische Unterstreichung.

Die nächste Regel befasst sich mit der Gestaltung der Buttons, wenn sich der Mauszeiger darüber befindet. Wir verwenden hier die dynamische Pseudoklasse `:hover`, um den Text in blauer Farbe darzustellen. Als besonderen Clou passen wir außerdem noch das Cursorsymbol an, wodurch die Illusion fast perfekt wird.

Die dritte Regel (`button:active`) legt schließlich fest, dass die Schrift rot »aufleuchten« soll, wenn der Link-Button angeklickt wird.

Diskussion

Das Anpassen von Submit-Buttons an das Aussehen des übrigen HTML-Texts ist perfekt für Designer, die der Meinung sind, dass die generischen Buttons nicht zu ihrer Gestaltung passen, die aber keine Grafik anstelle des Buttons verwenden möchten.

In der Lösung haben wir zweimal den Wert `inherit` verwendet, um anzugeben, dass die Werte für Schriftfarbe und -familie vom übergeordneten Element geerbt werden sollen.

Hiermit ist das Element gemeint, für das ausdrücklich oder implizit diese Werte festgelegt wurden. Das kann das Formular sein, ein umgebender `div`-Container oder auch das `body`-Element.

In den meisten Fällen müsste der Wert `inherit` das tun, was Sie wollen. Falls nicht, können Sie selbstverständlich auch eigene Werte für `font-size` und `font-family` angeben.

Wir haben die Gestaltung der Links in diesem Rezept an die Standarddarstellung der Browser angelehnt. Bei Bedarf sollten Sie Ihre Buttons mit ähnlichen Stildefinitionen versehen wie die übrigen Links Ihrer Website.



Internet Explorer 7 und 8 lassen sich unseres Wissens nicht davon abhalten, das »Drücken« der Buttons zu simulieren, indem der Beschriftungstext ein Stückchen nach unten recht verschoben wird.

Siehe auch

Die Rezepte 7.10 und 7.11 zur Gestaltung von Buttons; das Rezept 1.14 zur Verwendung von dynamischen Pseudoklassen; das Rezept 1.18 zur Vererbung.

7.13 Eingabefelder hervorheben

Problem

Das vom Benutzer gerade verwendete Formulareingabefeld soll visuell hervorgehoben werden.

Lösung

Nehmen Sie die Pseudoklasse `:focus`, um Elementen, die gerade den Fokus haben (z.B. weil Eingaben vorgenommen werden), mit eigenen Stilen zu versehen.

Mit der folgenden CSS-Regel können Sie die Hintergrundfarbe von gerade benutzten `input`- und `textarea`-Elementen verändern (siehe Abbildung 7-15):

```
input:focus {  
    background-color: #efefef;  
}
```

Durch die hier gezeigte Regel werden Formularefelder, die den Fokus haben, mit einem hellgrauen Hintergrund hinterlegt.

Diskussion

Mithilfe der Pseudoklasse `:focus` können Sie Ihren Besuchern die Benutzung von Formularen erleichtern, indem Sie das Feld hervorheben, in das gerade Eingaben vorgenommen werden.

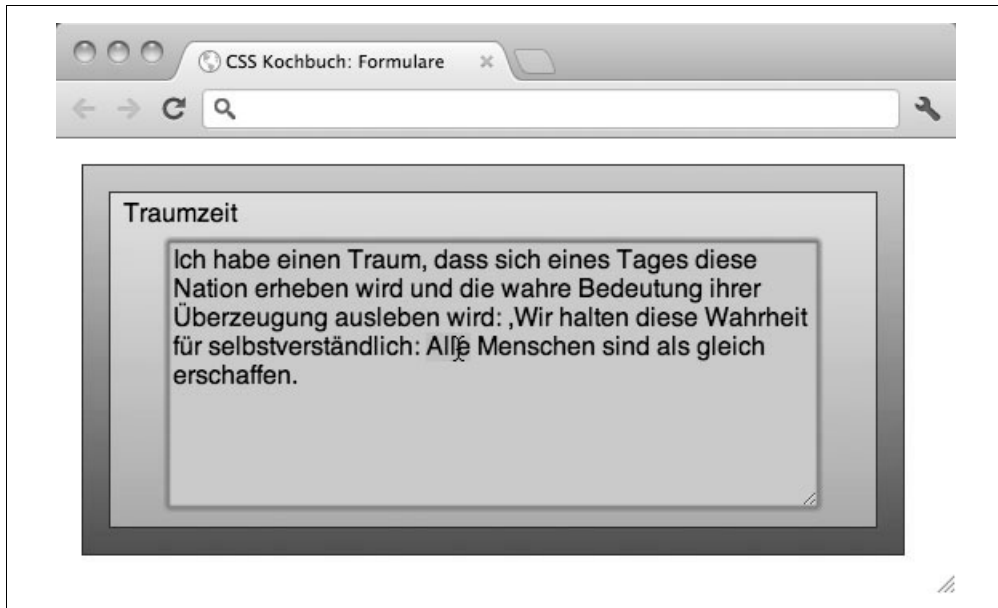


Abbildung 7-15: Die Hintergrundfarbe des gerade ausgewählten Eingabefeldes wird verändert

Das funktioniert übrigens nicht nur für Eingabefelder und Textbereiche. Auch andere Elemente können per `:focus` mit eigenen Stilen versehen werden.

autofocus

Die Spezifikation für HTML5 definiert für Formularelemente das neue Attribut `autofocus`. Ist dieses vorhanden, erhält das so gekennzeichnete Element automatisch den Fokus, und die Stildefinitionen für `:focus` werden aktiviert. Das könnte beispielsweise so aussehen:

```
<label for="vorname">Vorname:</label>
<input type="text" id="vorname" name="vorname" placeholder="Hier den Vornamen eingeben" />
```

Pseudoklassen kombinieren

In den meisten aktuellen Browsern ist es möglich, `:focus` mit weiteren Pseudoklassen zu kombinieren. Sie könnten also beispielsweise schreiben:

```
textarea:focus:hover {
    background-color: lightgreen;
}
```

Hat der Textbereich den Fokus und befindet sich der Mauszeiger darüber, wird der Inhalt des Felds mit einem hellgrünen Hintergrund hinterlegt.

Internet Explorer

Die Versionen 7 und 8 des Internet Explorer unterstützen die Pseudoklasse `:focus` noch nicht. Für Benutzer dieser Browser ist der in diesem Rezept beschriebene Effekt daher nicht sichtbar.

Siehe auch

Das Rezept 7.7 zur Definition von Stilen für `textarea`-Elemente; das *CSS3-Selectors*-Modul zur dynamischen Pseudoklasse `:focus` unter <http://www.w3.org/TR/css3-selectors/#useraction-pseudos>.

7.14 Benötigte Formularfelder hervorheben

Problem

Benutzer sollen sofort sehen, welche Formularfelder auf jeden Fall ausgefüllt werden müssen. Muss das Formular erneut angezeigt werden, weil die Felder leer oder fehlerhaft waren, sollen diese zusätzlich markiert werden.

Lösung 1

Verwenden Sie das neue HTML5-Attribut `required`, um obligatorische Formularfelder von anderen zu unterscheiden. Im Stylesheet können Sie dann die neue CSS3-Pseudoklasse `:required` verwenden, um benötigte Formularfelder speziell zu gestalten:

```
input[type="text"]:required {  
    border: 1px solid red;  
}
```

Mit dieser Regel werden die erforderlichen Textfelder mit einem roten Rahmen umgeben. Folgende Browser unterstützen die neue Pseudoklasse: Internet Explorer 10+, Firefox 4+, Safari 5+ (mit Einschränkungen), Chrome 10+, Opera 10.6+.

Lösung 2

Gehen Sie vor wie in Lösung 1, aber verwenden Sie diesmal einen Attributselektor, damit auch ältere Browser unterstützt werden.

```
input[type="text"][required] {  
    border: 2px solid yellow;  
}
```

(Hier erhalten erforderliche Formularfelder einen 2 Pixel breiten gelben Rahmen.)

Der Vorteil bei der Verwendung reiner Attributselektoren besteht darin, dass diese Lösung selbst mit älteren Browsermodellen wie IE7 noch funktioniert.

Diskussion

Durch die Verwendung des `required`-Attributs können Sie obligatorische Formularfelder hervorheben, und Benutzer erkennen leichter, wo auf jeden Fall Eingaben vorgenommen werden müssen.

Im Stylesheet reicht es aus, auf das Attribut selbst zu prüfen (`[required]`), weil die Angabe eines Werts nur notwendig ist, wenn Sie Ihr Webdokument im XHTML-Format verfassen. Obwohl das `required`-Attribut erst in der Spezifikation von HTML5 definiert wird, funktioniert seine Überprüfung auch im Internet Explorer 7. Der Unterschied besteht in der Methode, mit der geprüft wird. Während in aktuellen Browsern die CSS3-Pseudoklasse `:required` verwendet werden kann, ist es für ältere Modelle notwendig, einen passenden Attributselektor zu verwenden.

Andere Attribute und Attributwerte

Neben `required` definiert die HTML5-Spezifikation eine Reihe weiterer Attribute, die Sie auf die gleiche Weise im Stylesheet auswählen und gestalten können. Einige von ihnen erhalten in CSS3 eigene Pseudoklassen, die aber zur Drucklegung dieses Buchs nur teilweise von den Browsern unterstützt wurden. Eine Übersicht finden Sie in Tabelle 7-1.

Tabelle 7-1: CSS3-Pseudoklassen für Formulare

Pseudoklasse	Funktion
<code>:required</code>	Wählt Formularelemente aus, in denen Angaben gemacht werden müssen.
<code>:optional</code>	Wählt Formularelemente aus, in denen Angaben optional sind.
<code>:enabled</code>	Formularelemente, die benutzt werden können.
<code>:disabled</code>	Deaktivierte Formularelemente.
<code>:checked</code>	Formularelemente, die das <code>checked</code> -Attribut tragen, z.B. ein standardmäßig ausgewählte Checkbox.

Siehe auch

Eine Anleitung zur Integration von Formularüberprüfungen mit PHP unter <http://www.makemtemplate.com/feedback/>; die Website *caniuse.com* zum Ermitteln, welche Browser welche neuen CSS-Eigenschaften unterstützen; die CSS-Spezifikationen zur Gruppierung von Attributselektoren unter <http://www.w3.org/TR/CSS2/selector.html#matching-attrs> und <http://www.w3.org/TR/css3-selectors/#attribute-representation>; das Rezept 1.11 zur Gruppierung von Attributselektoren; die HTML5-Spezifikation zum Attribut `required` unter <http://www.w3.org/TR/html5/common-input-element-attributes.html#the-required-attribute>.

7.15 Formularelemente zeilenweise darstellen

Problem

Formularfelder und deren Beschriftungen sollen zeilenweise dargestellt werden, ohne Tabellen zu verwenden. Dabei soll das Layout allein mithilfe von CSS erstellt werden, ohne dass dabei Präsentations-Markup eingesetzt wird.

Lösung

Definieren Sie `input`- und `label`-Elemente als Blockelemente, um ihnen eine feste Breite zuzuweisen. Mit den Eigenschaften `float` und `clear` steuern Sie, wie die Elemente neben- und untereinander angeordnet werden:

```
input {  
    float: left;  
  
    margin-bottom: 0.5em;  
    width: 15em;  
}  
  
label {  
    float: left;  
    clear: left;  
  
    width: 10em;  
    padding-right: 1em;  
  
    text-align: right;  
}  
  
[type="checkbox"] {  
    width: 1em;  
}
```

Die Deklaration `float: left` sorgt dafür, dass die Elemente so weit wie möglich nach links verschoben werden. Als Nebeneffekt verhält sich das Element jetzt wie ein Blockelement, wodurch es möglich ist, ihm eine feste Breite zuzuweisen. Eine ausdrückliche Änderung der Darstellungsrolle per `display: block` ist hier also nicht nötig.

Um zwischen den Textfeldern etwas Platz schaffen, erhalten diese außerdem einen oberen Außenabstand von `0.5em`. Die Breite wird mit `15em` angegeben.

So ähnlich verfahren wir danach auch mit den Beschriftungen. Neben der Definition als links angeordnetes Float erhalten die `label`-Elemente jedoch zusätzlich die Deklaration `clear: left`. Hiermit wird festgelegt, dass sie andere Elemente nicht umfließen sollen, sondern unterhalb von anderen Floats platziert werden. Die Breite der Beschriftungen wird auf `10em` begrenzt.

Aus ästhetischen Gründen umgeben wir die Beschriftungen außerdem mit einem Innenabstand von 1em und legen fest, dass der Beschriftungstext innerhalb des label-Elements rechts ausgerichtet werden soll.

Die dritte Regel verwendet einen reduzierten Attributselektor, der für die Checkbox eine Breite von 1em festlegt.

Für Abbildung 7-16 haben wir das Basislayout der übrigen Rezepte in diesem Kapitel übernommen.



Abbildung 7-16: Textfelder und Beschriftungen werden auch ohne Tabellen und zusätzliches Markup untereinander angeordnet

Der HTML-Code für diese Lösung sieht so aus:

```
<form action="anmeldung.php" method="post">
<fieldset>
  <label for="benutzer">Benutzername:</label>
  <input type="text" name="benutzer" id="benutzer" placeholder="Benutzernamen
    eingeben" />
  <label for="password">Passwort:</label>
  <input type="text" name="password" id="password" placeholder="Passwort eingeben" />
  <label for="speichern">Daten speichern?</label>
  <input type="checkbox" name="speichern" id="speichern" class="checkbox" />
</fieldset>

<fieldset>
  <button type="submit">Abschicken</button>
  <button type="reset">Zurücksetzen</button>
</fieldset>

</form>
```

Diskussion

Der Schlüssel für diese Lösung liegt in der geschickten Verwendung der Eigenschaften `display`, `float` und `clear`.

Da wir alle Elemente als Blockelemente definiert haben, können wir ihnen unabhängig von ihrem Inhalt eine feste Breite zuweisen. Wenn wir ein solches Element mit der Deklaration `float: left` versehen, wird es aus dem normalen Fluss des umgebenden Elements (hier das Formular oder ein `fieldset`-Element) herausgenommen und so weit wie möglich nach links verschoben. Die folgenden Elemente umfließen das Element auf der rechten Seite. Da sowohl Textfelder wie auch deren Beschriftungen als `Float` definiert wurden, würden sie den vorhandenen horizontalen Platz des umgebenden Elements, soweit es geht, ausfüllen. Dadurch kann es passieren, dass z.B. die zweite Beschriftung rechts neben dem ersten Textfeld erscheint.

Das können Sie verhindern, indem Sie für die Beschriftungen, wie in der Lösung gezeigt, zusätzlich die Eigenschaft `clear` mit dem Wert `left` verwenden. Sie verhindert das Umfließen und verschiebt die zweite Beschriftung unter die erste.

Durch die Zuweisung einer festen Breite für die `label`-Elemente bleibt der Abstand zwischen Beschriftung und `input`-Elementen immer gleich. Er entspricht dem Wert für `width` zuzüglich eventuell definierter Innen- und Außenabstände sowie Rahmen.

Der Internet Explorer 7 weigert sich, die Checkbox neben seiner Beschriftung darzustellen. Durch einen ausreichend großen Wert für `margin-right` können wir das Problem aus der Welt schaffen, und die Darstellung funktioniert auch hier.

(Das Formular hat in unserem Beispiel eine Breite von 30 em-Einheiten. Vermutlich werden Sie den Wert von `margin-right` für Ihre Situation entsprechend anpassen müssen.) Um andere Browser nicht zu stören, steht die zusätzliche Regel in einem Conditional Comment:

```
<!--[if IE 7]><style type="text/css">
[type="checkbox"] { margin-right: 10em; }
</style><![endif]>
```

Siehe auch

Die HTML5-Spezifikation für weitere Informationen zum Element `label` unter <http://www.w3.org/TR/html5/forms.html#the-label-element>; das CSS-Wiki des W3C zur Verwendung der Eigenschaften `float` und `clear` unter <http://www.w3.org/wiki/CSS/Training/floating>.

7.16 Serviervorschlag: ein mehrspaltiges Formular

Problem

Sie möchten ein Formular erstellen, bei dem die gruppierten Formularfelder (fieldset) neben- und untereinanderstehen können. Nach dem Prinzip des Progressive Enhancement soll die Darstellung für moderne Browser aufgewertet werden, während ältere Versionen mindestens das Grundlayout darstellen sollen, das in Abbildung 7-17 zu sehen ist.

The image shows a web browser window with the title "CSS Kochbuch: Formulare". The browser's address bar is empty. The main content area displays a multi-column form. The form is organized into three distinct sections, each with a title in a small box at the top left of the section:

- Schritt 1: Persönliche Angaben**
 - Vorname:
 - Nachname:
 - Straße/Hausnummer:
 - Stadt:
 - PLZ:
 - Land:
- Schritt 2: Kontaktinformationen**
 - Telefon (privat):
 - Telefon (geschäftlich):
 - Telefon (mobil):
 - E-Mail:
 - Website:
- Schritt 3: Abschieken**
 - ☐ **Abschicken**
 - ☒ **Zurücksetzen**

Abbildung 7-17: Ein mehrspaltiges Formular

Lösung

Verwenden Sie die Eigenschaften `float` und `clear` für die verschiedenen `fieldset`-Elemente. Vergeben Sie außerdem passende Breitenangaben für Eingabefelder, Beschriftungen und `fieldset`-Elemente.

Weil die Browser unterschiedliche Innen- und Außenabstände für die Darstellung von Formularelementen verwenden, setzen wir die Werte für `margin` und `padding` zunächst für alle im Formular enthaltenen Elemente zurück:

```
form * {  
    margin: 0;  
    padding: 0;  
}
```

Es folgen die Anweisungen für das Formular selbst. Wir orientieren uns hierbei an der Gestaltung aus den vorigen Rezepten. Hierbei ist es wichtig, die richtige Breite zu ermitteln, damit die Darstellung wie gewünscht funktioniert:

```
form {  
    margin: 1em;  
    padding: 1em;  
    width: 41.5em;  
  
    background-color:    rgb(45,95,135);  
    background-image:    url('bilder/blauer-verlauf-hoch.jpg');  
    background-repeat:   repeat-x;  
  
    border: 1px solid rgb(50,50,50);  
    font-family: Helvetica, Arial, sans-serif;  
}
```

Zunächst erhält das Formular einen Innen- und einen Außenabstand von jeweils 1em. Der Außenabstand schafft Platz zwischen dem Formular und anderen möglichen Elementen auf der gleichen Seite. Der Innenabstand sorgt dafür, dass die gruppierten Felder (fieldset-Elemente) einen gewissen Grundabstand zu den Innenkanten des Formulars einhalten.

Die Breite geben wir hier mit 41.5em an. (Beachten Sie, dass Sie im Stylesheet einen Dezimalpunkt statt des Kommas angeben müssen.) Sie ergibt sich durch Addition der Breite der oberen fieldset-Elemente, den seitlichen Innen- und Außenabständen und den seitlichen Rahmen. Beachten Sie, dass die Innenabstände des Formulars hierbei nicht mitgerechnet werden.

Die folgenden drei Deklarationen definieren den Hintergrund des Formulars. Zunächst definieren wir einen dunklen Blauton als Hintergrundfarbe. Dieser wird verwendet, wenn das in der folgenden Deklaration definierte Bild nicht geladen oder angezeigt werden kann. Mit der Deklaration `background-repeat: repeat-x` wird festgelegt, dass das Hintergrundbild bei Bedarf horizontal wiederholt werden soll. Hierbei sollten Sie darauf achten, dass das Hintergrundbild eine ausreichende Höhe aufweist.



In CSS3-fähigen Browsern können Sie das Hintergrundbild auch mit der Deklaration `background-size: contain` so skalieren, dass die gesamte Fläche des Formulars davon bedeckt wird.

Zum Schluss dieser Regel erhält das Formular noch einen dunkelgrauen dünnen Rahmen. Als Schriftfamilie wählen wir Helvetica, Arial oder eine andere serifenlose Schrift.

Die einzelnen Formularfelder wurden anhand von `fieldset`-Elementen zu drei Gruppen zusammengefasst. Das soll dem Besucher die Bedienung erleichtern (und uns die Gestaltung). Hierbei werden die ersten beiden Gruppen zweispaltig nebeneinander dargestellt, während das dritte `fieldset`-Element sich unterhalb der beiden anderen über die gesamte Breite erstrecken soll:

```
fieldset {  
    float:      left;  
    margin-bottom: 1.5em;  
    padding:    0.5em 1em 0.5em 1em;  
    width:      18em;  
  
    background: rgb(164,202,218);  
    background: url("bilder/halbtransparent-weiss.png");  
    background: rgba(255,255,255,0.5);  
  
    border: 1px solid rgb(50,50,50);  
    text-align: left;  
}
```

Die erste Deklaration legt fest, dass zunächst alle `fieldset`-Elemente als Floats behandelt werden sollen. Dadurch werden sie möglichst weit nach links verschoben und vom folgenden Inhalt umflossen. Um etwas Abstand zu den darunterstehenden Elementen zu schaffen, erhalten sie einen unteren Außenabstand (`margin-bottom`) von 1.5em. Die Innenabstände werden mit 0.5em vertikal und 1em seitlich festgelegt. Außerdem erhalten die `fieldsets` eine feste Breite von 18em. Hierdurch passen genau zwei `fieldsets` nebeneinander in das umgebende Formular.

Für die Definition des Hintergrunds verwenden wir wieder die dreistufige Schreibweise aus Rezept 2.15, um eine möglichst große Rückwärtskompatibilität zu gewährleisten. Browser, die die `rgba()`-Schreibweise bereits verstehen, erzeugen von sich aus einen halb transparenten weißen Hintergrund. Ältere Modelle ignorieren diese Anweisung und verwenden stattdessen eine halb transparente PNG-Grafik. Konnte die Grafik nicht geladen werden, greift die erste `background`-Deklaration, die einen schlichten hellblauen Hintergrund für die `fieldset`-Elemente festlegt.



Durch diese Schreibweise brauchen CSS3-fähige Browser die PNG-Grafik nicht zu laden, weil sie die Transparenz selbst erzeugen können.

Wie bereits das Formular selbst, so erhalten auch die gruppierten Formularfelder einen dunkelgrauen dünnen Rahmen. Damit die Darstellung in möglichst vielen Browsern konsistent bleibt, legen wir schließlich noch fest, dass der Text (und alle anderen Inline-Inhalte) links ausgerichtet werden sollen.

Die hier gemachten Angaben gelten zuerst einmal für alle fieldset-Elemente. Sie werden im Folgenden teilweise wieder überschrieben.

Die zweite Regel verwendet die Pseudoklasse `:first-child`, um das erste fieldset (*Persönliche Angaben*) auf der rechten Seite mit einem zusätzlichen Außenabstand zu versehen. Damit schaffen wir den nötigen Zwischenraum zwischen der ersten und der zweiten Gruppe von Formularfeldern. Dies ist nötig, weil wir alle Innen- und Außenabstände am Anfang des Stylesheets auf null zurückgesetzt haben:

```
fieldset:first-child {  
    margin-right: 1em;  
}
```

Das dritte fieldset soll auf jeden Fall unterhalb der anderen beiden stehen und sich über die gesamte Breite des Formulars erstrecken. Damit die Regel nur für dieses Element gilt, benutzen wir hier eine Kombination der Pseudoklasse `:first-child` und zwei verketteten Selektoren für benachbarte Geschwisterelemente:

```
fieldset:first-child + fieldset + fieldset {  
    width: auto;  
    float: none;  
    clear: left;  
  
    margin-bottom: 0;  
}
```

Der erste Schritt besteht darin, die Definition als Float aus der allgemeinen Regel für fieldset-Elemente wieder aufzuheben. Diese wäre nicht unbedingt nötig. Sie hilft aber einigen älteren Browsern, die Höhe des Formulars und damit seines Hintergrunds richtig einzuschätzen.

Durch die Deklaration `clear: left` sorgen wir dafür, dass das dritte fieldset-Element auf jeden Fall unterhalb der beiden Spalten dargestellt wird. Im Gegensatz zu den ersten beiden fieldset-Elementen soll das dritte keinen eigenen unteren Außenabstand verwenden. Für den unteren Abstand zwischen der dritten Gruppe von Elementen und der Innenkante des Formulars reicht uns dessen Wert für `padding (1em)`.

Weil für die allgemeine fieldset-Regel eine Breite von 18em definiert wurde, müssen wir diese für die dritte Gruppe noch zurücksetzen. Um uns die Berechnung der Breite zu ersparen, verwenden wir einfach den Standardwert `auto`. Den Rest erledigt der Browser.

In der folgenden Regel kümmern wir uns um die Beschriftungen für die gruppierten Formularfelder. Anstatt uns auf die Standarddarstellung der Browser zu verlassen, wollen wir die Gestaltung in die eigenen Hände nehmen. Damit das klappt, haben wir in das legend-Element, wie in Rezept 7.3 beschrieben, ein zusätzliches `span`-Element eingefügt, an dem wir unsere Stildefinitionen vornehmen können:

```
legend span {  
    padding: 0.2em;  
  
    background: #efefef url("bilder/verlauf-grau.gif") repeat-x;  
    border: 1px solid rgb(50,50,50);  
}
```

```

    font-size: smaller;
    white-space: normal;
}

```

Nachdem wir rund um den Beschriftungstext etwas Leerraum geschaffen haben (padding), legen wir fest, dass als Hintergrundbild eine Grafik mit einem grauen Farbverlauf verwendet werden soll, die horizontal wiederholt wird. Kann sie nicht geladen werden, greift der erste Wert für background, und es wird ein einfacher hellgrauer Hintergrund angezeigt. Als Rahmen verwenden wir den gleichen Wert wie bereits für das Formular und die fieldset-Elemente. Die Schrift soll etwas kleiner erscheinen als im übrigen Formular. Daher verwenden wir hier das Schlüsselwort `smaller` für die Eigenschaft `font-size`.

Die letzte Deklaration behebt ein Problem mit Firefox (vor der Version 6). Ältere Versionen stellen den Inhalt von `legend`-Elementen unabhängig von seiner Länge in einer Zeile dar. Das kann bei längeren Beschriftungstexten zu Darstellungsproblemen führen. Durch die Anweisung `white-space: normal` wird diese Eigenschaft wieder auf ihren Standardwert zurückgesetzt, und längere Textpassagen werden automatisch umbrochen.

Sind das Formular und die `fieldset`-Elemente inklusive ihrer Beschriftungen mit den nötigen Stildefinitionen versorgt, können wir uns den eigentlichen Eingabeelement zuwenden. Wir beginnen mit den Beschriftungen für die einzelnen Formularfelder (`label`):

```

label {
    font-size: 0.8em;
    font-weight: bold;
}

```

Die Schriftgröße wird hier mit `0.8em` angegeben. Das heißt, sie soll 80 Prozent der Standardschriftgröße des Browsers (bzw. des umgebenden Elements) entsprechen. Dafür soll die Beschriftung fett erscheinen. Die übrige Darstellung überlassen wir dieses Mal dem Browser. Es folgen die Definitionen für die Eingabefelder (`input`):

```

input {
    display: block;

    padding: 0.4em;
    margin-top: 0.2em;
    margin-bottom: 0.2em;
    width: 17em;

    font-size: 1em;
    border: 1px solid black;
    background-color: #eee;
}

```

In der ersten Deklaration werden die Textfelder als Blockelemente definiert. Dadurch können wir für sie Außenabstände und eine feste Breite festlegen.

Die Außenabstände werden gebraucht, weil die Werte für padding innerhalb des Textfelds verwendet werden, um den Abstand des Eingabetexts zur Innenkante des Felds zu bestimmen (in unserem Fall 0.4em).

Die Textfelder erhalten ober- und unterhalb jeweils einen Außenabstand von 0.2em und sollen 17em breit sein. Die Schriftgröße geben wir hier ausdrücklich mit 1em an, wodurch der Text etwas größer als üblich erscheint.

Standardmäßig verwenden die meisten Browser die Eigenschaft border, um Textfeldern eine Art 3-D-Look zu geben. Wir ersetzen diese Darstellung durch eigene Regeln und umgeben das Feld stattdessen einfach mit einem dünnen schwarzen Rahmen. Als Hintergrundfarbe wählen wir Weiß.



Da alle Längen- und Breitenangaben in em-Einheiten vorgenommen wurden, beziehen sie sich auf die Schriftgröße des aktuellen Elements. Durch einen ausdrücklichen Wert für font-size können Sie die Größenverhältnisse selbst steuern.

Die folgende Regel bestimmt, wie ein Textfeld aussehen soll, wenn es gerade den Fokus hat, also bereit ist, Benutzereingaben entgegenzunehmen. Anstelle der Standardstile des Browsers wollen wir auch hier noch ein wenig Hand anlegen.

```
input:focus {  
    background-color: #ffffad;  
    outline: none;  
}
```

Sobald ein Textfeld den Fokus erhält, wechselt die Hintergrundfarbe (background-color) von Weiß zu einem hellen Blassgelb. Einige Browser verwenden die Eigenschaft outline, um Formularelemente, die gerade den Fokus haben, mit einem zusätzlichen Rahmen hervorzuheben. Das unterbinden wir, indem wir outline hier ausdrücklich den Wert none geben. Damit sind auch die Eingabefelder fertig gestaltet, und wir können uns den Buttons zuwenden.

Die Buttons sollen als Paar aus *Abschicken* und *Zurücksetzen* nebeneinanderstehen, gleichzeitig aber im umgebenden fieldset horizontal zentriert werden. Das erreichen wir, indem die Buttons mit einem zusätzlichen div-Element (mit der Klasse buttons) umgeben werden, für das wir die nötigen Deklarationen vornehmen können.

```
div.buttons {  
    margin-bottom: 0.5em;  
    text-align: center;  
}
```

Nachdem wir unterhalb des div-Elements etwas Leerraum geschaffen haben, können wir die Buttons selbst darin zentrieren. Da div-Elemente standardmäßig als Blockelemente definiert sind, dehnen sie sich über die gesamte Breite des umgebenden Elements (fieldset) aus. Basierend auf dieser Breite, berechnet der Browser dann die Platzierung der Buttons, die üblicherweise als Inline-Block-Elemente dargestellt werden. Deshalb brauchen wir in diesem Fall auch die Eigenschaft text-align, um die Zentrierung zu ermöglichen.

Der nächste Schritt kümmert sich um die Darstellung der Buttons selbst:

```
button {  
  height: 3em;  
  width: 12em;  
  padding-left: 40px;  
  
  border:      1px solid black;  
  
  line-height: 2.8em; /* 08 */  
  font-size:   1em;  
  font-weight: bold;  
}
```

Zu Beginn geben wir den Buttons eine eigene Höhe und Breite. Danach schaffen wir links neben der Beschriftung genügend Platz, um etwas später die »Abgehakt«- bzw. »Stopp«-Symbole einzufügen, die jeweils eine Breite von 40 Pixeln haben.



Durch die Definition eigener Werte für Breite und Höhe kann in den meisten Fällen verhindert werden, dass die Browser sich die Buttons vom jeweiligen Betriebssystem »ausleihen«, und wir können unsere eigene Gestaltung vornehmen.

Die Beschriftung der Buttons erhält eine explizite Zeilenhöhe (line-height) von 2.8em. Als »Maßstab« für die Längen- und Breitenangaben benutzen wir, wie bereits für die Textfelder, die Deklaration font-size: 1em, um eine möglichst konsistente Darstellung zu erreichen.

Die folgenden zwei Regeln steuern die Farbe der Beschriftung, wenn sich der Mauszeiger über einem Button befindet bzw. wenn der Button angeklickt wird:

```
button:hover {  
  color: orange;  
}  
  
button:active {  
  color: red;  
}
```

Im ersten Fall wird die Schrift orange angezeigt, im zweiten rot. Damit ist die Gestaltung der Buttons aber noch nicht abgeschlossen. Wie in Abbildung 7-18 zu sehen ist, haben die Buttons im Idealfall als Hintergrund einen grauen Farbverlauf, vor dem auf der linken Seite ein »Abgehakt«-, bzw. ein »Stopp«-Symbol zu sehen ist. Mithilfe einfacher Attribut-selektoren können wir die Buttons im Stylesheet unterscheiden und ihnen die nötigen Definitionen für die Hintergründe zuweisen:

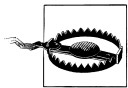
```
button[type="submit"] {  
  background: #efefef url('bilder/ok-sm.png') left center no-repeat;  
  background: url('bilder/ok-sm.png') left center no-repeat,  
  url("bilder/verlauf-grau.gif") repeat-x;  
}
```

```
button[type="reset"] {
    background: #efefef url('bilder/reset-sm.png') left center no-repeat;
    background: url('bilder/reset-sm.png') left center no-repeat,
    url("bilder/verlauf-grau.gif") repeat-x;
}
```



Abbildung 7-18: Durch die Definition mehrerer Hintergrundbilder für die Buttons wird sowohl das Symbol wie auch der graue Farbverlauf angezeigt

Wie in Rezept 2.15 beschrieben, verwenden wir hier mehrere background-Deklarationen. Die zweite gilt für alle Browser, die bereits mit mehreren Hintergrundbildern umgehen können.



Bei der Definition mehrerer Hintergrundbilder steht der zuerst definierte Hintergrund am weitesten »vorne«. Das zuletzt angegebene Hintergrundbild erscheint »hinter« den anderen.

Damit das Symbol vor dem grauen Farbverlauf erscheint, müssen wir es zuerst definieren. Es wird auf der linken Seite vertikal zentriert und nicht wiederholt. Nach dem Komma folgen die Angaben für den Farbverlauf, der entlang der x-Achse so oft wiederholt wird, bis der gesamte Button ausgefüllt ist.

Die erste der beiden background-Regeln dient uns als Sicherheitsnetz für Browser, die nur die einfache Schreibweise verstehen. Sie werden die zweite Deklaration einfach ignorieren. Zunächst wird eine allgemeine Hintergrundfarbe festgelegt (Hellgrau), vor der dann das passende Symbol angezeigt wird.

Für die Buttons ist noch eine Kleinigkeit zu erledigen. Weil die Browser für den Abstand der Buttons unterschiedliche Werte verwenden, geben wir ihn einfach selbst an. Mit dem Selektor für benachbarte Geschwisterelemente wählen wir den *Zurücksetzen*-Button aus und geben ihm einen linken Außenabstand von 0.5em:

```
button + button {
    margin-left: 0.5em;
}
```

Bevor die grundsätzliche Darstellung für das Formular abgeschlossen ist, müssen wir noch ein paar Anpassungen für den Internet Explorer 7 und 8 vornehmen, ansonsten erscheint das Formular, wie Sie es in Abbildung 7-19 sehen.

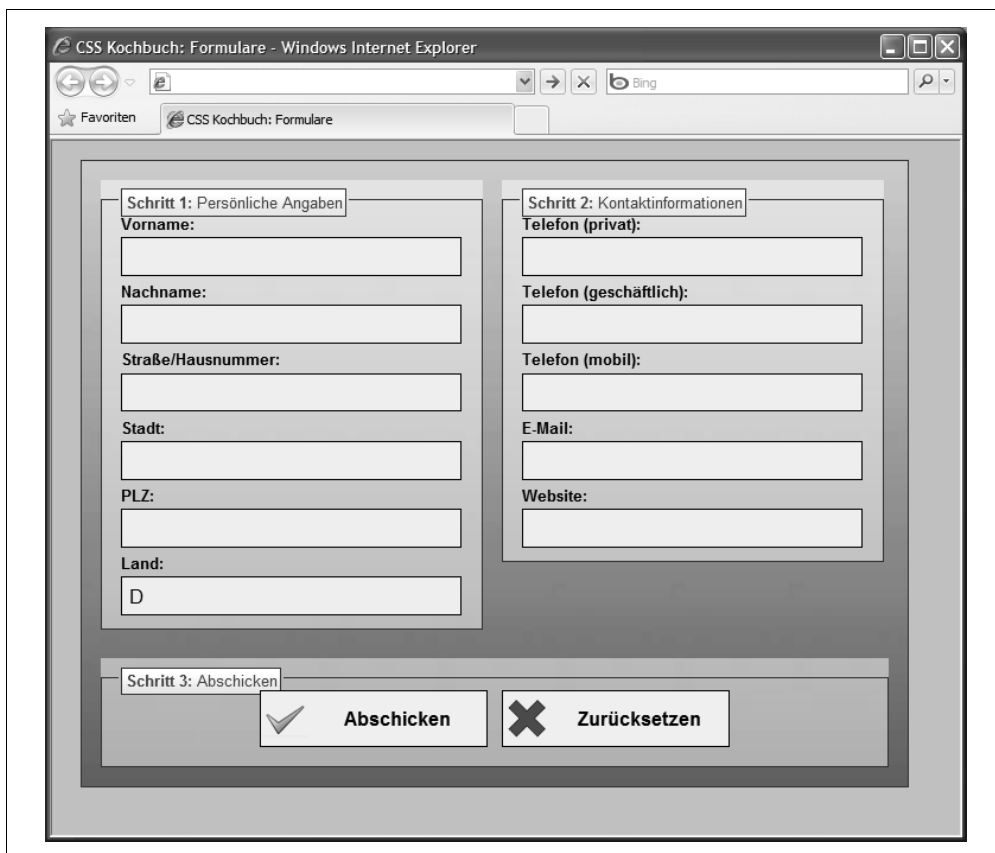


Abbildung 7-19: Ohne zusätzliche Angaben ragt der Hintergrund über die obere Rahmenkante und die Beschriftung der fieldset-Elemente hinaus

Um die Anweisungen vor anderen Browsern zu verbergen, umgeben wir sie mit Conditional Comments. Wir beginnen mit den Definitionen für beide Versionen:

```
<!--[if lte IE 8]>
<style type="text/css">
  legend { margin-top: -1.5em;}
  form { padding-top: 1.5em;}
```

Wie bereits in Rezept 7.3 besprochen, hat der Internet Explorer in den Versionen 7 und 8 die Angewohnheit, den Hintergrund des fieldset-Elements so darzustellen, dass er über die obere Rahmenkante und die Beschriftung (legend) hinausragt. Das gleichen wir aus, indem wir das legend-Element in der ersten Regel mit einem negativen Außenabstand nach oben verschieben.

Dadurch rücken aber sämtliche im Formular enthaltenen Elemente mit nach oben. Dem begegnen wir, indem wir den oberen Innenabstand (padding-top) des Formulars in der zweiten Regel entsprechend erweitern:

```

legend span {
    display:      inline-block;
    margin-top:   0.15em;
    margin-bottom: 0.5em;

    color: black; /* IE vererbt den Wert nicht! */
}

```

Die vertikale Mitte der Beschriftungen soll möglichst auf Höhe der oberen Rahmenlinie liegen. Gleichzeitig soll unterhalb der legend-Elemente genügend Abstand zu den folgenden Formularelementen sein. Um die nötigen Werte für `margin-top` und `margin-bottom` definieren zu können, muss zuvor noch die Darstellungsrolle der `span`-Elemente geändert werden.

Die letzte Deklaration legt für den Beschriftungstext Schwarz als Schriftfarbe fest, weil IE 7 und 8 diesen Wert nicht vererben. Ohne diese Anweisung würde der Beschriftungstext in blauer Schrift erscheinen.

Weil IE 7 und 8 die Pseudoklasse `:focus` noch nicht verstehen, ist der Einsatz unterschiedlicher Hintergrundfarben hier wenig sinnvoll. Die Textfelder erhalten daher einen einfachen weißen Hintergrund:

```

input {
    background-color: white;
    width: 16.2em; /* wegen border-top beim ersten fieldset */
}

```

Die zweite Deklaration macht die Eingabefelder gegenüber dem allgemeinen Stylesheet etwas schmaler, um ein weiteres Darstellungsproblem auszugleichen. Ist das Textfeld nach Meinung von IE 7 und 8 »zu breit«, wird die obere Rahmenlinie des ersten `fieldset`-Elements nach links über das `fieldset` hinaus verlängert. Damit sind die Anpassungen für Internet Explorer 7 und 8 abgeschlossen, und wir können den Conditional Comment wieder schließen.

```

</style>
<![endif]>

```

Es folgt eine letzte Regel, die nur der IE7 »sehen« soll. Daher verwenden wir einen zweiten, spezielleren Conditional Comment:

```

<!--[if IE 7]>
<style type="text/css">
legend {
    margin-left: -0.5em; /* Verschiebung ausgleichen */
}
</style>
<![endif]>

```

Der Internet Explorer 7 stellt legend-Elemente weiter rechts dar als andere Browser. Nachdem wir legend vorhin bereits als Inline-Block definiert haben, können wir hier einen negativen linken Außenabstand verwenden, um die Verschiebung wieder auszugleichen.

Diskussion

In diesem »Serviovorschlag« haben wir die verschiedenen Aspekte der Gestaltung von Formularen aus diesem Kapitel zusammengeführt. Wenn Sie ein paar Dinge beachten, können Sie Formularlayouts erstellen, die in allen Browser zwar nicht absolut gleich, aber zumindest sehr ähnlich aussehen. Zu den nötigen Anpassungen gehört zum einen das Zurücksetzen der Werte von `padding` und `margin` für sämtliche im Formular enthaltenen Elemente, zum anderen die Verwendung von `span`-Elementen um die Beschriftungen der `fieldset`-Elemente und, bei Bedarf, der `label`-Elemente.

Die seltsame obere Erweiterung des Hintergrunds der `fieldset`s im Internet Explorer lässt sich mit den passenden Werten für `margin-top` für das `legend`-Element bzw. `padding-top` für das Formular selbst in den Griff bekommen.

»Touche du Chef«

In guten Restaurants ist es üblich, dass der Chefkoch noch einmal selbst Hand anlegt, bevor das Essen serviert wird. Das kann eine besondere Dekoration oder ein geheimes Gewürz sein, das dem Gericht eine besondere Note verleiht. Ohne diese Definitionen würde das Formular aussehen wie das in Abbildung 7-20.

Wir wollen als »Touche du Chef« ein paar zusätzliche CSS3-Eigenschaften verwenden, die dem Formular den letzten Schliff geben. Durch Schattenwürfe am Formular selbst, den Buttons und in den Textfeldern erhält unser Werk zusätzliche räumliche Tiefe. Außerdem werden das Formular selbst, die drei `fieldset`-Elemente und die Buttons mit abgerundeten Ecken verschiedener Stärke versehen. Wir haben die folgenden Regeln erstellt:

```
input {
  -moz-border-radius: 0.3em;
  -webkit-border-radius: 0.3em;
  border-radius: 0.3em;

  -moz-box-shadow: 1px 1px 3px rgba(0,0,0,0.7) inset;
  -webkit-box-shadow: 1px 1px 3px rgba(0,0,0,0.7) inset;
  box-shadow: 1px 1px 3px rgba(0,0,0,0.7) inset;
}
```

Die oben stehende Regel versieht die Textfelder zunächst mit leicht abgerundeten Ecken und mit einem speziellen Schattenwurf. Dieser soll innerhalb des Elements erzeugt werden, damit der Eingabebereich etwas »tiefer« erscheint. Hierfür verwenden wir das Schlüsselwort `inset`.

Wie auch bei den folgenden Regeln geben wir die Eigenschaften zusätzlich zur (vorläufig) offiziellen Version auch in der Präfixschreibweise an, damit möglichst viele Benutzer unsere Erweiterungen sehen können².

² Es gibt immer noch genügend Browser, die diese Eigenschaften nicht unterstützen. Daher sollten Sie Ihre Gestaltung nicht von diesen Eigenschaften abhängig machen.

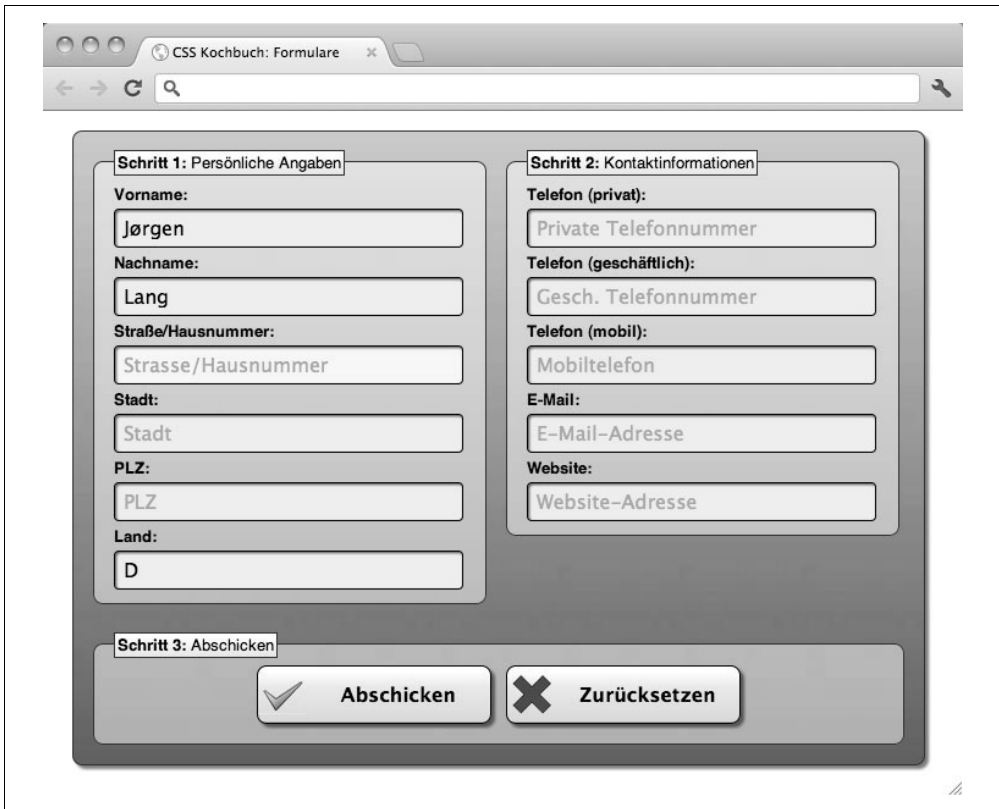


Abbildung 7-20: Das Formular mit zusätzlichen Stildefinitionen

```
form, fieldset, button {
  -moz-border-radius: 0.5em;
  -webkit-border-radius: 0.5em;
  border-radius: 0.5em;
}
```

In der obigen Regel definieren wir die abgerundeten Ecken für das Formular, die fieldset-Elemente und die Buttons. Da die Werte für alle drei Elemente gleich sind, können wir die Selektoren hier, wie in Rezept 1.4) beschrieben, zusammenfassen.

```
form, button {
  -moz-box-shadow: 3px 3px 3px rgba(0,0,0,0.5);
  -webkit-box-shadow: 3px 3px 3px rgba(0,0,0,0.5);
  box-shadow: 3px 3px 3px rgba(0,0,0,0.5);
}
```

Zum Abschluss erhalten Formular und Buttons noch einen Schattenwurf. Wir geben die Schattenfarbe hier in der `rgba()`-Schreibweise an. Dadurch passt sie sich bei Farbänderungen am Hintergrund besser an die neue Situation an.

Siehe auch

Das Rezept 1.16 zum Boxmodell; das Grundrezept 2.23 zur Verwendung von Floats; die anderen Rezepte in diesem Kapitel zur Gestaltung der verschiedenen Formularelemente; das Rezept 12.7 zu Schattenwürfen mit der CSS3-Eigenschaft `box-shadow`; das Rezept 12.6 zur Gestaltung von abgerundeten Ecken per `border-radius`; das Rezept 13.18 zur Verwendung von Polyfills, die den Gebrauch von CSS3-Eigenschaften und -Selektoren auch für ältere Versionen des Internet Explorer ermöglichen.

Einführung

Die Praxis, Tabellen für die Erstellung von Seitenlayouts zu verwenden, ist glücklicherweise schon seit längerer Zeit nicht mehr üblich (außer vielleicht in E-Mails). Auch das Verteilen eines Bilds auf mehrere Tabellenzellen (*Slicing*) oder verschachtelte Tabellen gelten mittlerweile als überholt. Dennoch hat der Einsatz von Tabellen auch weiterhin ihre Berechtigung.

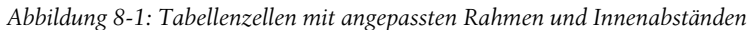
Heutzutage werden HTML-Tabellen eingesetzt, um *tabellarische* Informationen darzustellen. Das können die letzten Fußballergebnisse sein oder auch das Periodensystem der Elemente.

In diesem Kapitel lernen Sie, wie Sie bessere Designs erstellen können, indem Sie Tabellen mit Stildefinitionen für Tabellenüberschriften versehen, Rahmen für Tabellen und die enthaltenen Zellen definieren und unerwünschte Lücken bei der Platzierung von Bildern in Tabellenzellen vermeiden.

8.1 Rahmen und Innenabstände für Zellen einstellen

Problem

Um einen stärkeren visuellen Eindruck als bei der standardmäßigen Tabellendarstellung zu erreichen, sollen die Rahmen und die Innenabstände von Tabellenzellen angepasst werden (siehe Abbildung 8-1).



Verwenden Sie die Eigenschaft `padding`, um den Abstand zwischen dem Zelleneinhalt und den Rändern der Zelle festzulegen. Zur Definition der Rahmen für die Zellen und die Tabelle selbst gibt es die Eigenschaft `border`:

302 | Kapitel 8: Tabellen

Die erste Regel definiert die Darstellung der Tabelle selbst. Durch die Eigenschaft `border-collapse` wird festgelegt, dass die Rahmen der einzelnen Zellen zusammengefasst werden sollen, wie in der Diskussion erklärt wird. Die Eigenschaft `border` gilt nur für die Außenkanten der Tabelle und wirkt sich nicht auf die einzelnen Zellen aus.

Die zweite Regel (`td`) versieht alle Tabellenzellen mit einem Innenabstand. Für die Tabellenüberschriften (`th`) wurde kein eigener Wert für `padding` angegeben. Es wird hier also die Standarddarstellung der Browser verwendet.

Zur besseren Unterscheidung von den anderen Zellen sollen die Tabellenüberschriften einen schwarzen Hintergrund erhalten. Oft wird bei der Eigenschaft `color` davon gesprochen, dass sie die Schriftfarbe steuert. Tatsächlich kann sie sich aber auch auf die Rahmenfarben auswirken. Sofern Sie dies nicht ausdrücklich angeben (z.B. per `border-color`), wird der Wert von `color` benutzt, was gelegentlich zu unerwarteten Ergebnissen führen kann.¹

In Regel Nummer vier (`td, th + th`) wird für alle Tabellenzellen und alle Tabellenüberschriften, die ein benachbartes Geschwisterelement einer anderen Tabellenüberschrift sind, ein durchgehender roter Rahmen definiert. Durch den Selektor `th + th` ist die erste Überschrift davon allerdings ausgenommen. Statt eines eigenen Rahmens ist hier links und oberhalb der Rahmen der Tabelle sichtbar.

Die fünfte Regel (`td + td`) überschreibt die in der vorigen Regel für alle Zellen definierten Werte für `border` wieder – allerdings nicht für die erste Zelle jeder Zeile.

Die letzte Regel verwendet einen ID-Selektor, der hier der Lesbarkeit halber an das Element `td` gebunden ist, um eine Tabellenzelle mit besonderem Inhalt separat zu gestalten.

Diskussion

Für HTML-Tabellen gibt es zwei Modelle für die Darstellung von Rahmen: zusammengefasst (`collapse`) und getrennt (`separate`). Beim Modell der zusammengefassten Rahmen teilen sich aneinandergrenzende Zellen die Rahmenseiten. Im Modell der getrennten Rahmen besitzt jede Tabellenzelle ihren eigenen Rahmen. Beim Schreiben dieses Buchs ist das Modell der zusammengefassten Rahmen am weitesten verbreitet und wird daher von Designern auch häufiger verwendet.

Heutzutage verwenden alle Browser standardmäßig das Modell der zusammengefassten Rahmen. Einzige Ausnahme ist Firefox, der Tabellen mit getrennten Tabellenzellen darstellt. Da die CSS-Spezifikation hier kein bestimmtes Verhalten vorschreibt, empfiehlt es sich, das gewünschte Modell im CSS-Code explizit anzugeben. Um die Zellenrahmen zusammengefasst darzustellen, geben Sie der Eigenschaft `border-collapse` den Wert `collapse`:

```
table {  
    border-collapse: collapse;  
}
```

¹ Dieses Verhalten ist übrigens in der CSS 2.1-Spezifikation definiert. Unter <http://www.w3.org/TR/CSS2/box.html#border-properties> heißt es: »If an element's border color is not specified with a border property, user agents must use the value of the element's 'color' property as the computed value for the border color.«

In HTML können Sie anhand des Attributs `border` für das `table`-Element die Rahmenbreite für eine Tabelle und die enthaltenen Zellen festlegen. Für separate Rahmenbreiten können Sie die CSS-Eigenschaft `border` verwenden.



Die CSS-Eigenschaften für Rahmen haben übrigens Vorrang vor den HTML-Attributen `border` und `bordercolor`. Soll eine Tabelle im Rahmen eines Redesigns neu gestaltet werden, ist es also nicht unbedingt nötig, Änderungen am HTML-Code vorzunehmen.

Beim Modell der zusammengefassten Rahmen kann es passieren, dass sich die Stildefinitionen für benachbarte Zellen widersprechen. In diesem Fall kommen die folgenden Regeln zum Einsatz, um die Konflikte aufzulösen:

- Hat `border-style` für eine Zelle den Wert `hidden`, werden auch alle anderen Rahmenseiten verborgen.
- Hat `border-style` den Wert `none`, haben andere Werte für `border-style` Vorrang.
- Außer in Fällen, in denen `border-style` für eine Zelle den Wert `hidden` oder `none` hat, bekommt ein breiterer Rahmen Vorrang gegenüber schmaleren. Haben zwei Rahmen die gleiche Breite, wird der Rahmenstil anhand der folgenden (absteigenden) Reihenfolge festgelegt: `double`, `solid`, `dashed`, `dotted`, `ridge`, `outset`, `groove`, `inset`.
- Haben zwei Zellen die gleiche Breite und den gleichen Rahmenstil, aber unterschiedliche Farbdefinitionen, wird die Farbe anhand der folgenden Rangfolge von Elementen festgelegt: Zelle (`td`), Zeile (`tr`), Zeilengruppe (`rowgroup`), Spalte (`col`), Spaltengruppe (`colgroup`) und die Tabelle (`table`) selbst.

Das zweite Rahmenmodell wird als »separate« (getrennt) bezeichnet. Dabei besitzt jede Zelle ihren eigenen Rahmen und kann daher unabhängig von den Rahmen anderer Zellen gestaltet werden. Beim Modell der getrennten Rahmen können die horizontalen und vertikalen Abstände zwischen den Zellen anhand der Eigenschaft `border-spacing` festgelegt werden:

```
table {  
    border-collapse: separate;  
    border-spacing: 4px 4px;  
}
```

Hat die Eigenschaft `border-collapse` den Wert `separate`, werden eventuelle Stildefinitionen für Zeilen, Spalten oder Gruppen von Tabellenzellen nicht berücksichtigt. Außerdem kann anhand der Eigenschaft `empty-cells` festgelegt werden, ob Zellen ohne Inhalt dargestellt werden sollen oder nicht.

Zwar haben Webentwickler mit dem Modell der getrennten Rahmen eine größere Kontrolle über die Darstellung von Tabellen, allerdings wurde es zu dem Zeitpunkt, als dieses Buch geschrieben wurde, von Internet Explorer nur unvollständig unterstützt. Aus diesem Grund halten sich konservative Webdesigner an das Modell der zusammengefassten Rahmen.

Siehe auch

Die CSS 2.1-Spezifikation zu den verschiedenen Rahmenmodellen unter <http://www.w3.org/TR/CSS21/tables.html#propdef-border-collapse>; die ausführliche Diskussion der Darstellung von HTML-Tabellen mit CSS von Kapitel 11 in *CSS – Das umfassende Handbuch* von Eric A. Meyer (O'Reilly 2007); das Rezept 8.2 zur Definition des Zellzwischenraums per `border-spacing`; das Rezept 8.5 zur Eigenschaft `empty-cells`.

8.2 Den Zellzwischenraum einstellen

Problem

Beim Modell der separaten Rahmen für Tabellenzellen soll, ähnlich dem (in HTML5 veralteten) HTML-Attribut `cellspacing`, festgelegt werden, wie groß der Abstand der Zellen zueinander bzw. zum Rahmen der Tabelle selbst sein soll.

Lösung

Verwenden Sie die Eigenschaft `border-spacing`, wie hier gezeigt:

```
table {  
    border-collapse: separate;  
    border-spacing: 15px;  
}
```

Nachdem explizit festgelegt wurde, dass die Tabellenzellen separat angezeigt werden sollen, wird oberhalb, unterhalb und zwischen den Zellen ein Abstand von 15 Pixeln definiert.

Diskussion

In der CSS 2.1-Spezifikation ist ein Standardmechanismus beschrieben, um das Tabellenattribut `cellspacing` zu manipulieren. Hat die Eigenschaft `border-collapse` den Wert `separate`, kann man hierfür auch die Eigenschaft `border-spacing` verwenden, wie in der Lösung gezeigt.

Nach der CSS 2.1-Spezifikation können Sie für `border-spacing` zwei Werte angeben:

```
border-spacing: 0.5 1em;
```

Dabei gilt der erste Wert für die seitlichen Abstände, der zweite definiert den oberen und unteren Abstand. Auch wenn die Werte von `border-spacing` eigentlich die Zellen betreffen, wird diese Eigenschaft direkt für das `table`-Element definiert.

Internet Explorer 7

Inzwischen interpretieren alle aktuellen Browser außer Internet Explorer 7 die Werte für `border-spacing` richtig. In diesem Fall können Sie die »dynamische Eigenschaft« `expression` verwenden, um den IE7 mit Werten zu versorgen, die er versteht:

```

<!--[if IE 7]>
<style type="text/css">
table {
  border-collapse: expression('separate', cellSpacing = '15px');
}
</style>
<![endif]>

```

Der einzige Nachteil dieser Schreibweise liegt darin, dass nur ein Wert angegeben werden kann, der für den horizontalen und vertikalen Zellzwischenraum gleichermaßen gilt.

Zellzwischenräume entfernen: eine Alternative

Wenn Sie die Zwischenräume zwischen den Zellen entfernen wollen, ohne den Wert für `border-collapse` auf `collapse` zu ändern, können Sie sich auch damit behelfen, die Eigenschaft `border-spacing` auf null zu setzen:

```

border-collapse: separate;
border-spacing: 0;

```

Siehe auch

Das Rezept 8.1 zur Definition von Rahmen und Innenabständen für Zellen; die CSS 2.1-Spezifikation zur Eigenschaft `border-collapse` unter <http://www.w3.org/TR/CSS21/tables.html#propdef-border-collapse>; die CSS 2.1-Spezifikation zur Eigenschaft `border-spacing` unter <http://www.w3.org/TR/CSS21/tables.html#propdef-border-spacing>; das Blog des ungarischen Webentwicklers »Vacskamati« zum `expression`-Workaround für IE7 unter <http://vacskamati.blogspot.com/2007/03/eliminating-cellspacing-and-cellpadding.html>; die Microsoft-Website zur Verwendung von »dynamischen Eigenschaften« unter [http://msdn.microsoft.com/en-us/library/ms537634\(v=vs.85\).aspx](http://msdn.microsoft.com/en-us/library/ms537634(v=vs.85).aspx).

8.3 Leerraum zwischen Bildern und dem unteren Zellenrand entfernen

Problem

Ohne Stildefinitionen werden Bilder in Tabellenzellen (und anderen Elementen) auf der Grundlinie dargestellt. Dadurch entsteht unterhalb des Bilds etwas Leerraum, wie in Abbildung 8-2 gezeigt wird. Dieser Zwischenraum soll per CSS-Regel entfernt werden.

Lösung

Ändern Sie die Darstellungsrolle des Bilds, indem Sie es als Blockelement definieren:

```

td img {
  display: block;
}

```



Abbildung 8-2: Der Zwischenraum unterhalb eines Bilds in einer Tabellenzelle

Das Ergebnis sehen Sie in Abbildung 8-3.



Abbildung 8-3: Durch die Definition des Bilds als Blockelement wird der Leerraum unterhalb des Bilds entfernt

Diskussion

Das Bild wird als Blockelement dargestellt, damit darunter kein Leerraum angezeigt wird. Dieser Leerraum entsteht, weil Bilder eigentlich als Inline-Inhalte (wie z.B. Text) angesehen werden. Der Browser platziert das Bild daher auf der Grundlinie, die eigentlich für Text verwendet wird – selbst wenn die Zelle keinen Text enthält.

Die Grundlinie befindet sich nicht ganz an der Unterkante der Zelle (oder eines anderen umgebenden Elements), da einige Buchstaben (z.B. y, g, p und q) über die Grundlinie hinaus nach unten ragen.

Diese sogenannte *Unterlänge* kann nicht entfernt werden, weil die Grundlinie prozentual zur Gesamtgröße der verwendeten Schrift berechnet wird. Daher besteht die beste Möglichkeit, Bilder ohne Berücksichtigung der Grundlinie zu platzieren, darin, die Darstellungsrolle des Bilds anhand der Eigenschaft `display` auf den Wert `block` zu setzen.

Es kann sein, dass die Definition des Bilds als Blockelement nicht die beste Lösung darstellt, um den Leerraum unterhalb eines Bilds zu entfernen. Eine weitere Möglichkeit, das Grundlinienproblem zu lösen, besteht darin, der Eigenschaft `vertical-align` für das Bild den Wert `bottom` zuzuweisen. Hierbei muss das Bild allerdings höher sein als die Zeilenbox der Zelle.

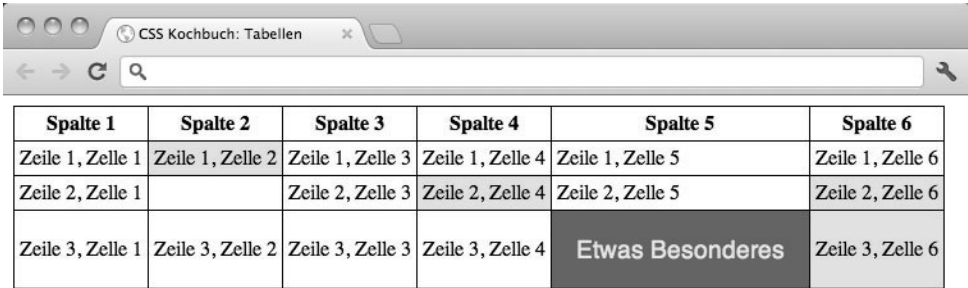
Siehe auch

Die CSS 2.1-Spezifikation zur Eigenschaft `display` unter <http://www.w3.org/TR/CSS21/visuren.html#propdef-display>; das Rezept 1.17 zu den verschiedenen Darstellungsrollen.

8.4 Tabellenzellen individuell gestalten

Problem

Sie möchten bestimmte Zellen einer Tabelle individuell gestalten, wie in Abbildung 8-4 dargestellt.



Spalte 1	Spalte 2	Spalte 3	Spalte 4	Spalte 5	Spalte 6
Zeile 1, Zelle 1	Zeile 1, Zelle 2	Zeile 1, Zelle 3	Zeile 1, Zelle 4	Zeile 1, Zelle 5	Zeile 1, Zelle 6
Zeile 2, Zelle 1		Zeile 2, Zelle 3	Zeile 2, Zelle 4	Zeile 2, Zelle 5	Zeile 2, Zelle 6
Zeile 3, Zelle 1	Zeile 3, Zelle 2	Zeile 3, Zelle 3	Zeile 3, Zelle 4	Etwas Besonderes	Zeile 3, Zelle 6

Abbildung 8-4: Einzelne Zellen einer Tabelle erhalten ein eigenes Design

Lösung 1

Versehen Sie die gewünschten Zellen je nach Verwendungszweck mit class- oder id-Attributen, die Sie dann im Stylesheet auswählen können:

```
td.hervorheben {
    background-color: yellow;
}

td#individuell {
    padding: 1em;

    background-color: rgb(100,100,100);
    color:          rgb(90%, 90%, 90%);

    font-family: Helvetica, Arial, sans-serif;
    font-size:   1.2em;
}
```

Hier erhalten alle Zellen, deren class-Attribut den Wert hervorheben hat, eine gelbe Hintergrundfarbe. Die Tabellenzeile, die individuell gestaltet werden soll, erhält einen größeren Innenabstand und einen dunkelgrauen Hintergrund, während der Text fast weiß dargestellt wird. Außerdem wurde eine serifenlose Schrift gewählt, die um 20 Prozent größer dargestellt werden soll als die Standardschrift des Browsers (oder des umgebenden Elements). Der HTML-Code könnte so aussehen:

```
...
<td id="individuell">Etwas Besonderes</td>
<td class="hervorheben">Zeile 3, Zelle 6</td>
...
```

Lösung 2

Verwenden Sie Selektoren für Nachkommenelemente und benachbarte Geschwisterelemente (+) sowie die Pseudoklasse :first-child, um die gewünschten Zellen gezielt auszuwählen. Für diese Lösung müssen Sie gut zählen können:

```
tr:first-child + tr + tr td:first-child,
tr:first-child      td:first-child + td,
tr:first-child + tr      td:first-child + td + td + td,
tr:first-child + tr      td:first-child + td + td + td + td + td,
tr:first-child + tr + tr td:first-child + td + td + td + td + td
{
    background-color: lightgreen;
}

tr:first-child + tr + tr td:first-child + td + td + td + td {
    padding: 1em;

    background-color: rgb(100,100,100);
    color:          rgb(90%, 90%, 90%);

    font-family: Helvetica, Arial, sans-serif;
    font-size:   120%;
}
```

Wie Sie sehen, sind die Selektoren hier deutlich komplexer als bei der Verwendung von Klassen und IDs. Der Vorteil besteht darin, dass Sie den HTML-Code nicht zu verändern brauchen, um die Stile zuzuweisen.

Diskussion

Beide Lösungen haben ihren Charme. Während Lösung 1 mit wenigen Selektoren auskommt, funktioniert die zweite Lösung auch, wenn Sie keinen Zugriff auf den HTML-Code haben.

Beim Ermitteln der richtigen Selektoren sollten Sie darauf achten, dass Sie zuerst angeben müssen, in welcher Zeile (tr) sich die Zelle befindet. Danach zählen Sie die Zellen von links nach rechts durch, bis Sie die gewünschte gefunden haben. Denken Sie hierbei auch daran, dass die Zeile mit den Tabellenüberschriften (th) mitgezählt werden muss.

Damit die Zählung auf jeden Fall bei der ersten Zeile bzw. Spalte beginnt, markieren wir das erste tr- bzw. td-Element mit der Pseudoklasse :first-child.

Hierbei gehen wir davon aus, dass die Zeile mit den Tabellenüberschriften (th) mit einem thread-Element umgeben ist. Die übrigen Zeilen befinden sich in einem tbody-Element:

```
<table>
<caption>Geschwister</caption>
<thead>
<tr>
  <th>Spalte 1</th>
  ...
  <th>Spalte 6</th>
</tr>
</thead>
<tbody>
<tr>
  <td>Zeile 1, Zelle 1</td>
  ...
  <td>Zeile 1, Zelle 6</td>
</tr>
</tbody>
</table>
```

Dadurch wird die Zeile mit den Überschriften bei der Zählung nicht berücksichtigt.

CSS3

Für moderne Browser, die bereits mit der Pseudoklasse :nth-child() umgehen können, gibt es eine noch elegantere Möglichkeit, einzelne Zellen auszuwählen. Die oben gezeigten Regeln könnten Sie auch so schreiben:

```
tr:nth-child(3) td:nth-child(1),
tr:nth-child(1) td:nth-child(2),
tr:nth-child(2) td:nth-child(4),
tr:nth-child(2) td:nth-child(6),
```

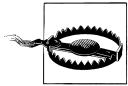
```
tr:nth-child(3) td:nth-child(6) {
    background-color: #ffe4c4;
}

tr:nth-child(3) td:nth-child(5) {
    padding: 1em;

    background-color: rgb(100,100,100);
    color:            rgb(90%, 90%, 90%);

    font-family: Helvetica, Arial, sans-serif;
    font-size:    120%;
}
```

Hier geben wir die Nummer der Zeile und Spalte als Parameter für die Pseudoklasse `:nth-child()` an. Wir gehen wieder davon aus, dass die `th`-Zellen mit einem `thead`-Element umgeben sind, die `td`-Elemente mit `tbody`.



Der Gebrauch von `thead` und `tbody` ist laut HTML5-Spezifikation nicht unbedingt erforderlich. Allerdings kommt in Safari 5.1 die Zählung der Elemente bei der Verwendung von `:nth-child()` durcheinander, wenn diese Elemente fehlen.

Sollten Sie einen anderen Dokumenttyp verwenden als HTML5, müssen die Bestandteile der Tabelle in folgender Reihenfolge angegeben werden: `thead`, `tfoot`, `tbody`. Erst in HTML5 kann `tfoot` auch unterhalb von `tbody` definiert werden.

Bei der Verwendung der Elemente `thead`, `tbody` und `tfoot` entsteht ein neuer Kontext für die Zählung der Tabellenzeilen. Wenn Sie beispielsweise schreiben:

```
tr:first-child td {
    background-color: yellow;
}
```

gilt diese Regel nicht nur für die erste Zeile mit Tabellenzellen, sondern möglicherweise auch für die letzte, sofern `tfoot` nur eine Zeile enthält. Die Regel gilt für das erste Kind-element des `tbody`- und des `tfoot`-Elements, weil der Zähler für jedes umgebende Element wieder zurückgesetzt wird.

Siehe auch

Das Rezept 1.14 zur Verwendung von Klassen- und ID-Selektoren; das Rezept 1.7 zur Verwendung von Nachkommenselektoren; das Rezept 1.9 zur Verwendung von Selektoren für benachbarte Geschwisterelemente; das Rezept 1.14 zur Verwendung der Pseudoklasse `:first-child`; das Rezept 1.14 zu möglichen Werten für die Pseudoklasse `nth-child()`.

8.5 Leere Tabellenzellen verstecken

Problem

Sie möchten, dass Tabellenzellen ohne Inhalt nicht dargestellt werden, wie in Abbildung 8-5 zu sehen ist.

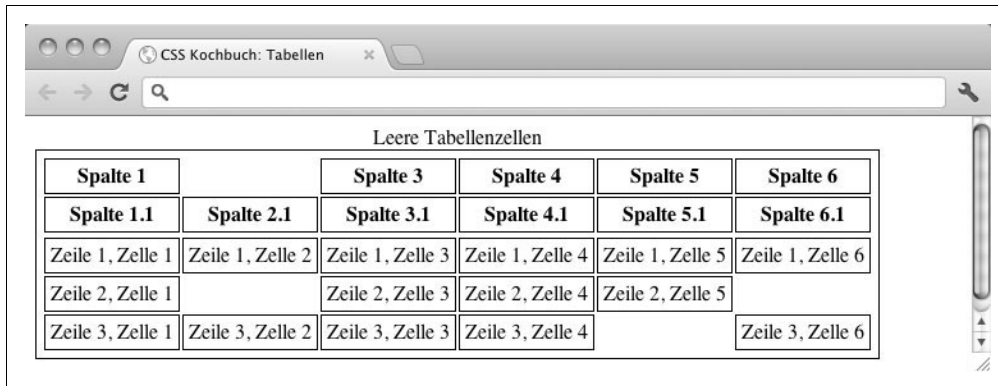


Abbildung 8-5: Leere Tabellenzellen werden beim Darstellungsmodell der separaten Rahmen nicht mit angezeigt

Lösung

Geben Sie der Eigenschaft `empty-cells` den Wert `hide`, wie hier gezeigt:

```
table {  
  border-collapse: separate;  
  empty-cells:    hide; /* raw */  
}
```

Diskussion

Die Eigenschaft `empty-cells` funktioniert nur mit dem Darstellungsmodell der getrennten Rahmen, d.h., wenn die Eigenschaft `border-collapse` den Wert `separate` hat.

Standardmäßig werden Zellen auch dann dargestellt, wenn sie keinen Inhalt haben. Gelegentlich ist es bei der Gestaltung von Tabellen jedoch gewünscht, leere Zellen auszublenden. Dabei hilft die Eigenschaft `empty-cells`.

Beachten Sie, dass auch »unsichtbare« Zeichen wie ` ` als Inhalt betrachtet werden und das Ausblenden der betreffenden Zelle verhindern.

Siehe auch

Das Rezept 8.2 zur Verwendung zusammengefasster und getrennter Rahmen.

8.6 Stildefinitionen für die Tabellenüberschriften

Problem

Für Tabellenüberschriften (th) sollen eigene Stile definiert werden. Abbildung 8-6 zeigt eine Tabelle mit Überschriften in der Standarddarstellung, in Abbildung 8-7 sehen Sie die gleiche Tabelle, diesmal mit Stildefinitionen für die Überschriften.

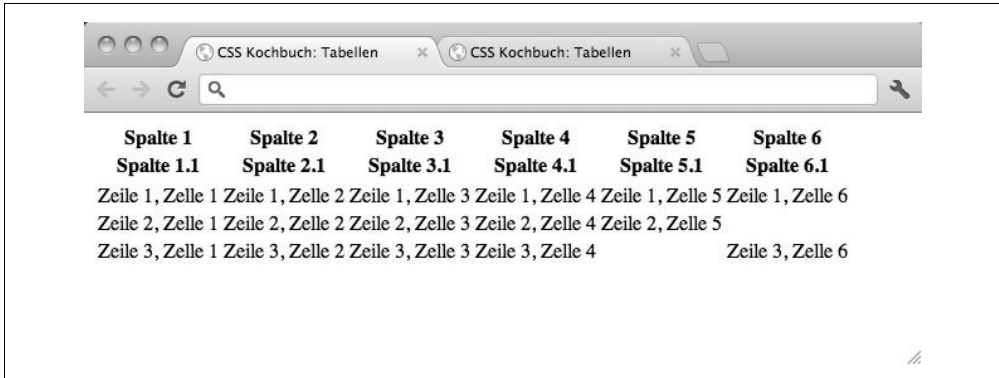


Abbildung 8-6: Die Tabelle in der Standarddarstellung von Google Chrome

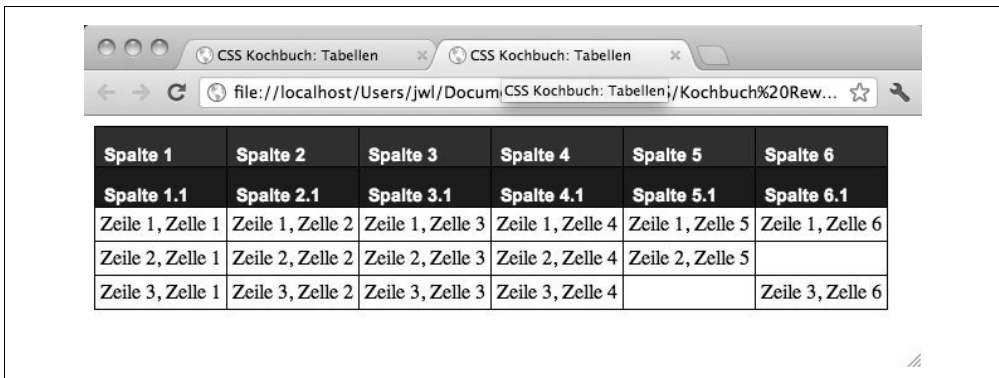


Abbildung 8-7: Eine Überschriftenzeile mit Stildefinitionen

Lösung

Verwenden Sie den Elementselektor th, um den Überschriftenzellen der Tabelle die entsprechenden Stile zuzuweisen:

```
th {  
    padding: 1em 1.5em 0.1em 0.5em;  
  
    font-family: Arial, Helvetica, Verdana, sans-serif;  
    font-size: 0.9em;  
}
```

```

text-align: left;
color:      white;
}

```

Die erste Deklaration versieht die einzelnen th-Elemente mit Innenabständen. Danach wird eine serifenlose Schrift definiert, die um zehn Prozent kleiner als die Standardschriftgröße und links ausgerichtet dargestellt werden soll. (Standardmäßig werden Tabellenüberschriften in den meisten Browsern zentriert.) Als Schriftfarbe wird hier Weiß gewählt.

Die beiden Überschriftenzeilen sollen eine unterschiedliche Hintergrundfarbe erhalten. Daher wählen wir zuerst mithilfe der Pseudoklasse `:first-child` die erste Zeile aus:

```

tr:first-child th {
    background-color: blue;
}

```

Hierdurch erhalten die Überschriftenzellen der ersten Zeile einen blauen Hintergrund. Um explizit die zweite Zeile mit Überschriften anzusprechen, wird der oben verwendete Selektor nochmals erweitert:

```

tr:first-child + tr th {
    background-color: #009;
}

```

Durch den Selektor für benachbarte Geschwisterelemente können wir nur den Überschriften in der zweiten Tabellenzeile eine eigene Hintergrundfarbe zuweisen.

Diskussion

Das Element `th` kennzeichnet den Inhalt einer Zelle als Überschrift. Um Überschriftenzellen visuell von einfachen Tabellenzellen (`td`) zu unterscheiden, können Sie für beide Zellentypen verschiedene Stile definieren. Hierfür reicht es bereits aus, den Eigenschaften `font-family`, `background-color`, `font-size`, `font-weight` oder auch `text-align` unterschiedliche Werte zu geben.

Siehe auch

Das *Selectors*-Modul der CSS3-Spezifikation zu Elementselektoren unter <http://www.w3.org/TR/css3-selectors/#type-selectors>; das Rezept 1.14 zur Verwendung der Pseudoklasse `:first-child`; das Rezept 3.1 zur Definition von Schriftfamilien; die Rezepte 3.2 und 3.5 zum Festlegen von Schriftgrößen und -stilen.

8.7 Stildefinitionen für Tabellenbeschriftungen

Problem

Die Beschreibung einer Tabelle (`caption`) soll mit eigenen Stildefinitionen versehen werden.

Lösung

Verwenden Sie den Elementselektor `caption`:

```
caption {  
    padding:      1em;  
    font:         1.5em Georgia, "Times New Roman", Times, serif;  
    caption-side: bottom;  
}
```

Die erste Deklaration versieht die Beschriftung rundum mit einem Innenabstand von 1em. Für die Definition von Schriftgröße und -art verwenden wir hier die Kurzschrifteigenschaft `font`.

Mit der dritten Deklaration (`caption-side`) wird festgelegt, dass die Beschriftung nicht wie normalerweise über, sondern unterhalb der Tabelle angezeigt werden soll.

Diskussion

Auch wenn sich die Darstellung des Elements `caption` von Browser zu Browser unterscheiden kann, stellen alle Browser es dar. Es kann per CSS mit Stildefinitionen versehen werden und ist die zugänglichste Methode, Tabellen mit Beschriftungen zu versehen.

Für Fälle, in denen die Beschriftung unterhalb einer Tabelle angezeigt werden soll, hat sich das World Wide Web Consortium (W3C) die Eigenschaft `caption-side` ausgedacht, mit der Sie – wie in der Lösung gezeigt – selbst bestimmen können, wo die Beschriftung angezeigt wird.

Die Eigenschaft `caption-side` kennt zwei Werte

- `top` (Standardwert, die Beschriftung wird über der Tabelle dargestellt)
- `bottom` (die Beschriftung wird unterhalb der Tabelle dargestellt)

Bis auf den Internet Explorer 7 wird `caption-side` von allen aktuellen Browsern unterstützt.

Siehe auch

Die HTML5-Spezifikation zum Element `caption` unter <http://www.w3.org/TR/html5-author/the-caption-element.html>.

8.8 Rollover-Effekte für Tabellenzeilen

Problem

Wenn sich der Cursor über einer bestimmten Tabellenzeile befindet, soll diese durch einen Rollover-Effekt hervorgehoben werden (siehe Abbildung 8-8).

Lösung

Verwenden Sie die Pseudoklasse `:hover` zusammen mit dem Elementselektor `tr`:

```
tr:hover {  
    background: yellow;  
}
```



Abbildung 8-8: Die Tabellenzeile, über der sich gerade der Mauszeiger befindet, wird hervorgehoben

Diskussion

Die Pseudoklasse `:hover` kennen Sie vielleicht schon vom Erzeugen von Rollover-Effekten für Links. Die CSS-Spezifikation beschränkt ihren Einsatz aber nicht darauf. Sie kann auf beliebige Elemente wie z.B. `p` oder `div` angewandt werden.

Wenn Sie für einige Zeilen bereits einen Hintergrund definiert haben – z.B. mithilfe von `tr:nth-child(even)` – kann es nötig sein, dem Selektor für den Rollover-Effekt eine höhere Spezifität zu geben, damit die Hervorhebung für alle Zeilen funktioniert:

```
table tr:hover {  
    background-color: yellow;  
}
```

Oder aber Sie verwenden die `!important`-Direktive. Genau für diese Zwecke ist sie schließlich gedacht:

```
tr:hover {  
    background-color: yellow !important;  
}
```

Durch den zusätzlichen Elementselektor `table` erhält die Regel eine höhere Spezifität und hat damit Vorrang vor den wechselnden Zeilenhintergründen.

Siehe auch

Das *Selectors*-Modul der CSS3-Spezifikation zu dynamischen Pseudoklassen unter <http://www.w3.org/TR/css3-selectors/#dynamic-pseudos>; das Rezept 1.14 zu Pseudoklas-

sen; das Rezept 1.20 zum Thema Spezifität; das Rezept 8.11 zur Definition wechselnder Hintergründe für Tabellenzeilen; das Rezept 1.21 zur !important-Direktive; das Rezept 12.1 zur Definition von Rollover-Effekten für andere HTML-Elemente.

8.9 Tabellenzeilen individuell gestalten

Problem

Sie möchten einzelne oder mehrere Tabellenzeilen individuell gestalten, wie in Abbildung 8-9 gezeigt. Für dieses Rezept gehen wir davon aus, dass die Tabelle die folgende Struktur hat:

```
<table>
<caption>Tabellen mit Stil</caption>
<tr>
  <th>Spalte 1</th>
  <th>Spalte 2</th>
  <th>Spalte 3</th>
</tr>
<tr>
  <td>Zeile 1, Zelle 1</td>
  <td>Zeile 1, Zelle 2</td>
  <td>Zeile 1, Zelle 3</td>
</tr>
<tr>
  <td>Zeile 2, Zelle 1</td>
  <td>Zeile 2, Zelle 2</td>
  <td>Zeile 2, Zelle 3</td>
</tr>
... weitere Zeilen ...
</table>
```

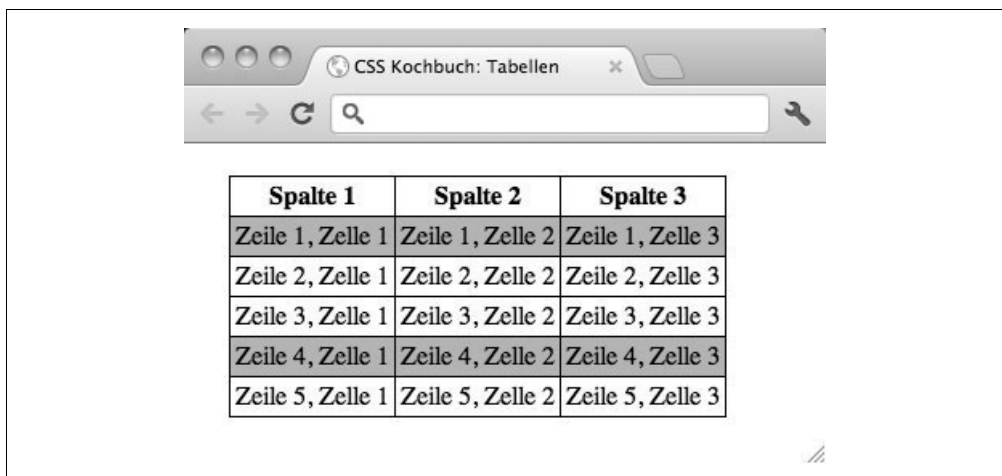


Abbildung 8-9: Bestimmte Tabellenzeilen erhalten eigene Stile.

Lösung 1

Verwenden Sie den CSS3-Selektor `:nth-child()`, um bestimmte `tr`-Elemente auszuwählen. Um die erste und die vierte Zeile nach den Spaltenüberschriften auszuwählen, könnten Sie beispielsweise schreiben:

```
tr:nth-child(2),
tr:nth-child(5) {
    background: rgb(180,180,180);
}
```

In dieser Lösung verwenden wir die einfachste Schreibweise von `:nth-child()`, um ganz bestimmte Elemente auszuwählen. Dafür schreiben Sie in die runden Klammern des Selektors einfach die Nummer des Elements, das Sie gestalten wollen.

Weil auch die Spaltenüberschriften von einem `tr`-Element umgeben sind, beginnt die Zählung der Zeilen für `:nth-child()` schon hier. Um also die erste reguläre Zeile anzusprechen, müssen Sie `nth-child(2)` schreiben und so fort.

Der Einfachheit halber haben wir die Selektoren für die erste und die vierte Zeile in dieser Lösung gruppiert. Die einzelnen Selektoren werden dabei durch Kommata voneinander getrennt.

Für die Hintergrundfarbe benutzen wir die `rgba()`-Notation aus dem *CSS3-Colors*-Modul. Dieses hat seit dem 7. Juni 2011 den Status einer offiziellen W3C-Empfehlung und wird von allen modernen Browsern unterstützt.

Lösung 2

Benutzen Sie eine Kombination aus der Pseudoklasse `:first-child` und dem Selektor für benachbarte Geschwisterelemente (`+`), um die gewünschten Zeilen separat zu gestalten:

```
tr:first-child + tr,
tr:first-child + tr + tr + tr +tr {
    background: rgb(180,180,180);
}
```

Um eindeutig anzugeben, für welche Elemente die Deklaration gelten soll, verwenden wir die Pseudoklasse `:first-child`. Hierdurch wird sichergestellt, dass die Zählung tatsächlich mit der ersten Tabellenzeile (Spaltenüberschrift) beginnt. Danach hängen wir eine passende Anzahl von Selektoren für benachbarte Geschwisterelemente an.

Diese Methode hat den Vorteil, dass sie auch von älteren Browsern zuverlässig verstanden wird. Die Nachteile sind allerdings ebenfalls nicht zu übersehen: Jede Zeile, die speziell gestaltet werden soll, muss mit einem eigenen Selektor ausgewählt werden. Das kann bei großen Tabellen eine Menge Arbeit bedeuten. Bei einer unbekannten Anzahl von Zeilen ist diese Methode ebenfalls kaum praktikabel.

Damit auch die Zuweisung der Hintergrundfarbe funktioniert, haben wir hier anstelle der `rgba()`-Notation einen passenden `rgb`-Wert gewählt.

Lösung 3

Versehen Sie die `tr`-Elemente der Zeilen, die eigene Stile erhalten sollen, mit einem passenden `class`-Attribut. Dieses können Sie dann im Stylesheet nach Ihren Wünschen gestalten.

CSS:

```
tr.eigener-stil {  
    background: rgb(180,180,180);  
}
```

HTML:

```
<tr class="eigener-stil">...</tr>
```

Dies ist die »klassische« Methode, bestimmte Zeilen mit eigenen Stilen zu versehen. Hierbei erweitern Sie die entsprechenden `tr`-Elemente um ein `class`-Attribut, das Sie dann im Stylesheet über den passenden Klassenselektor ansprechen können.

Im Gegensatz zur vorigen Lösung ist es hierbei nicht nötig, für jede Zeile einen eigenen Selektor zu verwenden. Stattdessen muss jedoch der HTML-Code modifiziert werden.

Diskussion

Jede der hier vorgestellten Lösungen hat ihre Vor- und Nachteile. Je nachdem, für welche Zielgruppe Sie Ihre Seiten gestalten und mit welchen Mitteln der HTML-Code erzeugt wird, sind die Methoden unterschiedlich gut geeignet.

Wenn Sie wissen, dass der allergrößte Teil Ihrer Besucher mit modernen Browsern auf Ihre Seiten zugreift, reicht Lösung 1 möglicherweise schon aus. Sie sollten jedoch im Vorfeld überlegen, ob die Gestaltung der Spalten für das Verständnis des Inhalts absolut notwendig ist. Falls nicht, können die Stile im Sinne des Progressive Enhancement für ältere Browser vielleicht einfach weggelassen werden.

Soll die Gestaltung dagegen mit möglichst vielen Browsern funktionieren, sind die Lösungen 2 oder 3 vermutlich besser geeignet. Hat die Tabelle eine überschaubare Anzahl von Zeilen, bietet sich die Verwendung benachbarter Geschwisterelemente an, weil keine Änderungen im HTML-Code nötig sind.

Sobald die Anzahl der Zeilen zu groß wird (wann das ist, entscheiden Sie!), wird es zu mühsam und unübersichtlich, für jede gewünschte Regel einen eigenen Selektor zu definieren. Zudem bläst das Ihre CSS-Datei unverhältnismäßig stark auf.

In diesem Fall bietet sich die dritte Lösung an. Sie kommt mit einem schlichten Klassenselektor aus. Stattdessen müssen Sie aber beim HTML-Dokument Hand anlegen. Jede Zeile, die gesondert gestaltet werden soll, benötigt ein eigenes `class`-Attribut.

Werden die Tabellen dynamisch erzeugt (z.B. aus einer Datenbank), lässt sich die dafür verwendete Software (z.B. ein CMS) eventuell so erweitern, dass das Attribut bei bestimmten Zeilennummern automatisch eingefügt wird.

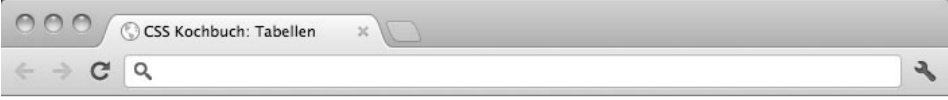
Siehe auch

Das *Selectors*-Modul der CSS3-Spezifikation unter <http://www.w3.org/TR/css3-selectors/>; Kapitel 1, *Allgemeines*, zur Verwendung der verschiedenen Selektoren; das Rezept 1.14 zu den Pseudoklassen `:nth-child()` und `:first-child`; das Rezept 1.4 zur Gruppierung von Selektoren; das Rezept 1.9 zum Selektor für benachbarte Geschwisterelemente.

8.10 Tabellenspalten gestalten

Problem

Bestimmte Tabellenspalten sollen eine eigene Hintergrundfarbe erhalten, und es soll eine serifenlose, fett ausgezeichnete Schrift verwendet werden, wie in Abbildung 8-10 gezeigt wird.



Spalte 1	Spalte 2	Spalte 3	Spalte 4	Spalte 5	Spalte 6
Zeile 1, Zelle 1	Zeile 1, Zelle 2	Zeile 1, Zelle 3	Zeile 1, Zelle 4	Zeile 1, Zelle 5	Zeile 1, Zelle 6
Zeile 2, Zelle 1	Zeile 2, Zelle 2	Zeile 2, Zelle 3	Zeile 2, Zelle 4	Zeile 2, Zelle 5	Zeile 2, Zelle 6
Zeile 3, Zelle 1	Zeile 3, Zelle 2	Zeile 3, Zelle 3	Zeile 3, Zelle 4	Zeile 3, Zelle 5	Zeile 3, Zelle 6
Zeile 4, Zelle 1	Zeile 4, Zelle 2	Zeile 4, Zelle 3	Zeile 4, Zelle 4	Zeile 4, Zelle 5	Zeile 4, Zelle 6

Abbildung 8-10: Bestimmte Tabellenspalten erhalten eigene Stile

Lösung 1

Benutzen Sie die CSS3-Pseudoklasse `:nth-child()`, um die Zellen in einer bestimmten Tabellenspalte auszuwählen. Um beispielsweise jede dritte Spalte mit einem gelben Hintergrund zu versehen, könnten Sie schreiben:

```
th:nth-child(3n),
td:nth-child(3n) {
  background-color: yellow;
  font-family:    Helvetica, Arial, sans-serif;
  font-weight:    bold;
}
```

Diese Regel verwendet zwei Selektoren, um jeweils die dritte Zelle einer Tabellenzeile auszuwählen. Da für Spaltenüberschriften ein anderes HTML-Element (`th`) benutzt wird als für normale Zellen (`td`), benötigen wir hier einen eigenen Selektor. Der Wert in den

runden Klammern der Pseudoklasse `:nth-child()` sorgt dafür, dass jede dritte Spalte ausgewählt wird. In unserem Fall sind das die Spalten vier und sechs.

Die Schreibweise `:nth-child(3n)` steht dabei für »jedes dritte Element«. Näheres zu den Werten von `:nth-child()` finden Sie in Rezept 1.15.

Lösung 2

Verwenden Sie eine Kombination aus der Pseudoklasse `:first-child` und Selektoren für benachbarte Geschwisterelemente. Um den gleichen Effekt zu erreichen wie in Lösung 1, könnten Sie auch Folgendes schreiben:

```
th:first-child + th + th,  
th:first-child + th + th + th + th + th,  
td:first-child + td + td,  
td:first-child + td + td + td + td + td {  
    background-color: yellow;  
    font-family:      Helvetica, Arial, sans-serif;  
    font-weight:      bold;  
}
```

Gegenüber der ersten Lösung ist der Selektor-Teil der Regel dieses Mal schon deutlich komplexer. Das liegt daran, dass wir für jede Spalte, die angesprochen werden soll, einen eigenen Selektor erstellen müssen. Außerdem benötigen wir für die Spaltenüberschriften (th) ebenfalls einen passenden Satz Selektoren.

Die Pseudoklasse `:first-child` sorgt dafür, dass die Zählung tatsächlich in der ersten Spalte begonnen wird. Nach Bedarf werden dann Selektoren für benachbarte Geschwisterelemente hinzugefügt, um die gewünschten Spalten auszuwählen.

Lösung 3

Versehen Sie die Tabellenzellen in den gewünschten Spalten mit einem passenden class-Attribut, das Sie dann im Stylesheet auswählen können, um die Spalte nach Belieben zu gestalten:

HTML:

```
<table>  
<caption>Tabellen mit Stil</caption>  
<tr>  
    <th>Spalte 1</th>  
    <th>Spalte 2</th>  
    <th class="spaltenstil">Spalte 3</th>  
    <th>Spalte 4</th>  
    <th>Spalte 5</th>  
    <th class="spaltenstil">Spalte 6</th>  
</tr>  
<tr>  
    <td>Zeile 1, Zelle 1</td>  
    <td>Zeile 1, Zelle 2</td>
```

```

    <td class="spaltenstil">Zeile 1, Zelle 3</td>
    <td>Zeile 1, Zelle 4</td>
    <td>Zeile 1, Zelle 5</td>
    <td class="spaltenstil">Zeile 1, Zelle 6</td>
</tr>
... noch mehr Inhalt
</table>

```

CSS:

```

.spaltenstil {
    background-color: yellow;
    font-family:      Helvetica, Arial, sans-serif;
    font-weight:      bold;
}

```

Bei dieser Lösung liegt die Hauptarbeit nicht im CSS-Code, sondern in der Erweiterung der Spaltenüberschriften und Tabellenzellen im HTML-Dokument.

Anstatt für th- und td-Elemente einen eigenen Selektor anzugeben, verwenden wir hier der Einfachheit halber einen allgemeinen Klassenselektor, der für alle Elemente gilt, die mit der Klasse `spaltenstil` markiert wurden. Dieses Rezept basiert auf folgendem HTML-Code:

```

<table>
<caption>Tabellen mit Stil</caption>

<tr>
    <th>Spalte 1</th>
    <th>Spalte 2</th>
    <th class="spaltenstil">Spalte 3</th>
    <th>Spalte 4</th>
    <th>Spalte 5</th>
    <th class="spaltenstil">Spalte 6</th>
</tr>
<tr>
    <td>Zeile 1, Zelle 1</td>
    <td>Zeile 1, Zelle 2</td>
    <td class="spaltenstil">Zeile 1, Zelle 3</td>
    <td>Zeile 1, Zelle 4</td>
    <td>Zeile 1, Zelle 5</td>
    <td class="spaltenstil">Zeile 1, Zelle 6</td>
</tr>

... weitere Zeilen ...
</table>

```



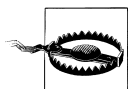
Die Verwendung der `class`-Attribute ist nur für Lösung 3 erforderlich. Für die ersten beiden Lösungen müssen keine Änderungen am HTML-Code vorgenommen werden.

Diskussion

Ähnlich wie bei der Gestaltung von Tabellenzeilen gibt es auch hier drei verschiedene Lösungsansätze. Tabellen werden im HTML-Code als Folge von Zeilen (tr) definiert, die einzelne Zellen (td) enthalten.

Um eine Spalte zu gestalten, müssen Sie daher Stile definieren, die nur für bestimmte Zellen (td) einer Zeile gelten. Das funktioniert am einfachsten mit der CSS3-Pseudoklasse `:nth-child()`. Die anderen beiden Lösungen können entweder schnell sehr komplex werden (Lösung 2) oder erfordern eine Erweiterung des HTML-Codes.

Besonders spannend wird die Sache, wenn sich einige Zellen über mehrere Spalten erstrecken sollen (z.B. per `colspan="2"`). In diesem Fall kommen Sie an der Verwendung spezieller Klassen im HTML-Quelltext kaum vorbei.



Die Lösungen 1 und 2 funktionieren am besten, wenn alle Zellen den gleichen Wert für `colspan` haben oder das Attribut gar nicht benutzen. Ansonsten kann es passieren, dass die Zählung durcheinanderkommt.

Und was ist mit col und colgroup?

Wenn Sie sich mit HTML gut auskennen, sind Ihnen die Elemente `col` und `colgroup` vermutlich nicht fremd. Mit ihnen kann man Spalten bzw. Spaltengruppen definieren.

Leider sind die Möglichkeiten von CSS hier sehr beschränkt. Es können nur die Eigenschaften `border`, `width`, `background` und `visibility` verwendet werden. Wenn Sie ein Hintergrundbild für ein `colgroup`-Element definieren wollen, ist die Darstellung außerdem nicht einheitlich. Während Webkit-basierte Browser wie Chrome oder Safari ein für `col` bzw. `colgroup` definiertes Hintergrundbild in jeder Tabellenzelle einzeln darstellen, erstreckt es sich in Firefox über den gesamten Bereich.

Aufgrund dieser Beschränkungen und Inkonsistenzen haben wir `col` und `colgroup` in unseren Lösungen nicht verwendet, sondern die Zellen direkt mit Stilen versehen.

Rollover-Effekte

Eine Definition von Rollover-Effekten für Spalten mit der Pseudoklasse `:hover` ist nur mithilfe von JavaScript möglich. Hier gibt es allerdings einige recht gute Lösungen, z.B. das jQuery-Plug-in *tableHover* von Roman Weich (einen Link dazu finden Sie unter »Siehe auch«).

Siehe auch

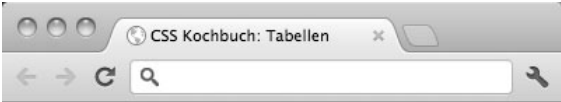
Den Artikel über die Gestaltung von Tabellenspalten von Eric A. Meyer unter <http://archivist.incutio.com/viewlist/css-discuss/110084>; eine ausführliche Erklärung, warum die Definition von Stilen für `col` und `colgroup` nur sehr eingeschränkt funktioniert, unter <http://ln.hixie.ch/?count=1&start=1070385285>; das Rezept 8.11 zur Gestaltung von Tabellenzeilen; das Rezept 1.14 zu den Pseudoklassen `:first-child` und `:nth-`

child()); das Rezept 1.12 zur Verwendung von Klassenselektoren; die CSS 2.1-Spezifikation zur Gestaltung von Spalten unter <http://www.w3.org/TR/CSS2/tables.html#columns>; das *tableHover*-Plug-in für jQuery von Roman Weich unter <http://p.sohei.org/jquery-plugins/tablehover/>.

8.11 Abwechselnde Hintergrundfarben für Tabellenzeilen

Problem

Eine Tabelle soll mit abwechselnden Hintergrundfarben versehen werden, wie in Abbildung 8-11 gezeigt.



The screenshot shows a web browser window titled "CSS Kochbuch: Tabellen". Below the address bar is a table with 3 columns and 5 rows. The columns are labeled "Spalte 1", "Spalte 2", and "Spalte 3". The rows are labeled "Zeile 1, Zelle 1" through "Zeile 5, Zelle 3". The rows have alternating background colors: the first, third, and fifth rows are white, while the second, fourth, and fifth rows are light gray.

Spalte 1	Spalte 2	Spalte 3
Zeile 1, Zelle 1	Zeile 1, Zelle 2	Zeile 1, Zelle 3
Zeile 2, Zelle 1	Zeile 2, Zelle 2	Zeile 2, Zelle 3
Zeile 3, Zelle 1	Zeile 3, Zelle 2	Zeile 3, Zelle 3
Zeile 4, Zelle 1	Zeile 4, Zelle 2	Zeile 4, Zelle 3
Zeile 5, Zelle 1	Zeile 5, Zelle 2	Zeile 5, Zelle 3

Abbildung 8-11: Eine Tabelle mit abwechselnden Hintergrundfarben für die Zellen

Lösung 1

Verwenden Sie die CSS3-Pseudoklasse `:nth-child()` mit dem Wert `odd` bzw. `even`, wie hier gezeigt:

```
tr:nth-child(even) {  
    background: rgba(100,100,100,0.4);  
}
```

Durch den Wert `even` erhält jede zweite Tabellenzeile einen grauen Hintergrund.

Lösung 2

Benutzen Sie Selektoren für benachbarte Geschwisterelemente, um jeweils jede zweite Tabellenzeile auszuwählen:

```
table#geschwister tr:first-child + tr,  
table#geschwister tr:first-child + tr + tr + tr,  
table#geschwister tr:first-child + tr + tr + tr + tr + tr {  
    background: rgba(100,100,100,0.4);  
}
```


Bei dieser Lösung ist es nicht nötig, die einzelnen tr-Elemente mit zusätzlichen class-Attributen zu versehen. Allerdings ist diese Methode eher für kleine Tabellen mit wenigen Zeilen geeignet, da für jede ein eigener Selektor verwendet werden muss.

Lösung 3

Erstellen Sie für geradzahlige Tabellenzeilen einen eigenen class-Selektor:

```
tr.even {  
    background: rgba(180,180,180);  
}
```

Danach versehen Sie jede zweite Zeile der Tabelle mit dem entsprechenden class-Attribut, das den Wert even erhält:

```
<tr>  
    [...]  
</tr>  
  
<tr class="even">  
    [...]  
</tr>  
  
<tr>  
    [...]  
</tr>  
  
<tr class="even">  
    [...]  
</tr>
```

Diskussion

Je mehr Browser die CSS3-Pseudoklasse :nth-child() unterstützen, desto mehr wird vermutlich die erste Lösung zur Anwendung kommen. Die zweite hier gezeigte Möglichkeit hat den Vorteil, dass sie auch in älteren Browsern noch zuverlässig funktioniert. Der Preis ist die Definition eines eigenen Selektors für jede zweite Zeile.

Während die beiden ersten Lösungen ohne zusätzlichen HTML-Code auskommen, basiert Lösung 3 auf der Verwendung des class-Attributs und einem entsprechenden Selektor.

Jedes zweite tr-Element manuell mit einem class-Attribut zu versehen, kann besonders bei langen Tabellen recht mühsam sein. Dafür funktioniert diese Technik mit allen CSS-fähigen Browsern.

Wir haben hier, entsprechend dem Schlüsselwort für die Pseudoklasse :nth-child(), even als Wert für das class-Attribut gewählt. Im Stylesheet könnten Sie die Selektoren dann beispielsweise gruppieren, wie hier:

```
tr:nth-child(even), tr.even {  
    background: rgb(180,180,180);  
}
```

Auf diese Weise sehen Sie schnell, an welcher Stelle Ihre CSS-Regeln gegebenenfalls angepasst werden müssen – beispielsweise wenn die abwechselnde Gestaltung in der ersten Zeile beginnen soll.



Ist die Tabelle im HTML-Code in `thead`, `tbody` und `tfoot` unterteilt, beginnt die Zählung der Elemente jeweils von Neuem. In diesem Fall müssen Sie die Regeln entsprechend anpassen.

JavaScript

Andere Lösungen gehen über die reine Verwendung von CSS hinaus. Eine Möglichkeit, abwechselnde Hintergrundfarben für Tabellenzeilen zu benutzen, stützt sich beispielsweise auf JavaScript-Code, der mit dem Document Object Model (DOM) interagiert und die Stile automatisch auf jede zweite Tabellenzeile anwendet. Ein Beispiel für solch eine Lösung finden Sie unter <http://www.alistapart.com/articles/zebratables>. Der Nachteil dieser Vorgehensweise besteht darin, dass sie nur funktioniert, wenn der Benutzer in seinem Browser JavaScript auch aktiviert hat.

Serverseitige Lösungen

Eine weitere Lösung besteht in der Verwendung einer serverseitigen Programmiersprache wie PHP oder ColdFusion. Mit einem einfachen Skript können Sie die Erzeugung der Tabelle automatisieren. (Diese Technik ist auch dann vorteilhaft, wenn ein Datenbank-Backend für die Erzeugung und Verwaltung der Tabellendaten verwendet wird.) Eine mögliche PHP-Lösung zur Aufgabenstellung finden Sie unter <http://www.phpfreaks.com/tutorials/5/0.php>.

Siehe auch

Das *Selectors*-Modul der CSS3-Spezifikation zur Pseudoklasse `:nth-child` unter <http://www.w3.org/TR/css3-selectors/#nth-child-pseudo>; das Rezept 1.14 zu den Pseudoklassen `:first-child` und `:nth-child()`; das Rezept 1.12 zur Verwendung von Klassenselektoren.

8.12 Tabellen mit einer festen Breite versehen

Problem

Sie möchten die Breite einer Tabelle ausdrücklich festlegen. Das kann sinnvoll sein, wenn Ihr Seitenlayout mit Pixelgrößen arbeitet oder Sie ein Stylesheet für Ausdrücke erstellen.

Lösung

Geben Sie der Tabelle eine feste Breite und verwenden Sie die Eigenschaft `table-layout` mit dem Wert `fixed`:

```
table {  
    width: 100px;  
    table-layout: fixed;  
}
```

Diskussion

In diesem Beispiel erhält die Tabelle eine relativ geringe Breite von 100 Pixeln. Dadurch kann es passieren, dass die Tabelle zu schmal ist, um die Inhalte darzustellen. Im Normalfall nimmt sich die Tabelle einfach den benötigten Platz.

Mit der Eigenschaft `table-layout` können Sie dieses Verhalten selbst steuern. Durch den Wert `fixed` wird in allen aktuellen Browsern die tatsächlich angegebene Breite verwendet. Dadurch kann es natürlich passieren, dass die Inhalte über die Ränder ihrer Zellen oder auch der Tabelle selbst hinausragen.

Das können Sie ändern, indem Sie für die Tabellenüberschriften (`th`) und -zellen (`td`) einen passenden Wert für die Eigenschaft `overflow-x` definieren, zum Beispiel:

```
td, th {  
    font-size: 3em;  
    overflow-x: auto;  
}
```

Ist der Inhalt zu breit für die Tabellenzelle, wird bei Bedarf ein horizontaler Scrollbalken angezeigt.

Ein Nebeneffekt der Verwendung von `table-layout: fixed` ist, dass die Darstellung der Tabelle beschleunigt wird, weil der Browser nicht zuerst die maximale und minimale Breite der einzelnen Spalten ermitteln muss, sondern einfach den angegebenen Wert benutzen kann.

Die Eigenschaft `table-layout` wird von allen aktuellen Browsern unterstützt.

Siehe auch

Die CSS 2.1-Spezifikation zur Eigenschaft `table-layout` unter <http://www.w3.org/TR/CSS2/tables.html#width-layout>; das Rezept 3.13 zu verschiedenen Methoden, mit Textüberlauf und Zeilenumbrüchen umzugehen.

8.13 Serviervorschlag: eine Tabelle mit Stil

Problem

Eine Tabelle soll mithilfe von Stylesheets möglichst sauber und übersichtlich gestaltet werden (siehe Abbildung 8-12). Hierbei sollen folgende Vorgaben erfüllt werden:

- Die Spaltenüberschriften sollen einen eigenen Hintergrund erhalten.
- Die Zeilen sollen abwechselnd mit unterschiedlichen Hintergrundfarben versehen werden.
- Die Zeile, über der sich der Cursor gerade befindet, soll mit einem Rollover-Effekt versehen werden.
- Es wird davon ausgegangen, dass hauptsächlich moderne Browser für den Zugriff auf die Tabelle verwendet werden, ältere Browser sollen die Tabelle dennoch ebenfalls nutzen können.
- Dementsprechend dürfen CSS3-Gestaltungselemente zwar verwendet werden, allerdings nur nach dem Prinzip des Progressive Enhancement. Die Tabelle muss auch ohne diese Elemente funktionieren.
- Die erste und die letzte Spalte sollen eigene Stile erhalten. Spaltenüberschriften und die Fußzeile der Tabelle sollen davon nicht betroffen sein.
- Die Anzahl der Einträge in der Tabelle ist nicht bekannt.
- Es sollen keine Änderungen am Quellcode vorgenommen werden.



Abbildung 8-12: Eine Tabelle mit Stil

Lösung

Hier kombinieren Sie die in diesem Kapitel vorgestellten Rezepte, um die genannten Vorgaben zu erfüllen. Zu Beginn geben wir der Tabelle ein paar passende Innen- und Außenabstände. Außerdem wird festgelegt, dass die Rahmen der einzelnen Zellen zusammengefasst werden sollen. Als Schriftfamilie legen wir Helvetica, Arial oder eine andere serifenlose Schrift fest:

```
table {
  padding:      3px;
  margin:       20px;
  border-collapse: collapse;

  font-family: Helvetica, Arial, sans-serif;
}
```

Es folgen die Anweisungen für die Tabellenbeschriftung. Per `caption-side: bottom` wird festgelegt, dass sie unterhalb der Tabelle erscheinen soll. Außerdem erhält sie eine eigene Schrift (nach Möglichkeit eine Variante der Times oder eine andere Serifenschrift), die kursiv dargestellt werden soll:

```
caption {
  caption-side: bottom;
  font-family: "Times New Roman", Times, serif;
  font-style: italic;
}
```

Um die Spaltenüberschriften mit einem eigenen Hintergrund zu versehen, benutzen wir hier den Elementselektor `thead`, um einen eigenen Wert für die Eigenschaft `background` anzugeben:

```
thead {
  background: rgba(20,200,200,0.4);
}
```

Die zweite Vorgabe (sich abwechselnde Hintergründe für die Tabellenzeilen) wird erfüllt durch die Verwendung des CSS3-Selektors `nth-child()` mit dem Wert `even`. Da wir gemäß den Vorgaben den HTML-Code der Tabelle nicht verändern dürfen und die Anzahl der Einträge in der Tabelle nicht bekannt ist, ist dies die einzige Möglichkeit, die verschiedenen Hintergründe umzusetzen.

```
tr:nth-child(even) {
  background: rgba(200,200,200,0.4);
}
```

Gleichzeitig ist diese Form der Zeilengestaltung für die Benutzung der Tabelle nicht unbedingt notwendig. Wir können diese Form des Progressive Enhancement hier also bedenkenlos einsetzen.

Der nächste Schritt versieht alle Tabellenzellen (`td`, auch die im Fußteil) und Spaltenüberschriften (`th`) mit einem grauen Rahmen und etwas Innenabstand, der die Lesbarkeit der Inhalte erleichtern soll.

```
td, th, td {
    border: 1px solid #aaa;
    padding: 0.3em 0.5em;
}
```

Danach folgt die Stildefinition für aktive tr-Elemente, also diejenigen Tabellenzeilen, über denen sich der Cursor gerade befindet. Wir setzen hierfür den Elementselektor tr zusammen mit der Pseudoklasse :hover ein, um den Hintergrund bei Bedarf gelb einzufärben:

```
tbody tr:hover {
    background-color: yellow;
}
```

Laut unseren Vorgaben sollen die erste und die letzte Spalte mit eigenen Stilen versehen werden. Um zu verhindern, dass die Stile auch auf die Fußzeile der Tabelle angewandt werden, verwenden wir hier eine Kombination aus Nachkommeselektor und den Pseudoklassen :first-child bzw. :last-child, um jeweils das erste und letzte td-Element einer Zeile im Tabellenkörper (tbody) auszuwählen.

Die Inhalte sollen ins Fettschrift erscheinen und mit einem 3 Pixel breiten orangefarbenen Rahmen versehen werden:

```
tbody td:first-child, tbody td:last-child {
    font-weight: bold;
    border: 3px solid orange;
}
```

Zum Abschluss noch ein bisschen was fürs Auge: Die Zellen der Spaltenüberschriften (th) erhalten per text-shadow einen subtilen 3-D-Look, indem wir rechts unterhalb des Texts einen 1 Pixel breiten weißen Textschatten definieren:

```
th {
    text-shadow: 1px 1px 1px #fff;
}
```

Auch die Tabelle selbst erhält einen dunkelgrauen Schlagschatten. Damit sich dieser besser an den Hintergrund der Seite anpassen kann, entscheiden wir uns hier für die rgba()-Schreibweise.

```
table {
    -webkit-box-shadow: 3px 3px 5px rgba(0,0,0,0.5);
    -moz-box-shadow: 3px 3px 5px rgba(0,0,0,0.5);
    box-shadow: 3px 3px 5px rgba(0,0,0,0.5);
}
```

Da die Schattenwürfe für die Benutzung der Tabelle selbst nicht unbedingt notwendig sind, müssen wir uns keine Sorgen darum machen, dass sie in älteren Browsern möglicherweise nicht dargestellt werden können.

Zur Erstellung dieser Lösung haben wir den folgenden HTML-Code verwendet:

```
<table>
<caption>Eine Tabelle mit Stil</caption>
```

```

<thead>
<tr>
<th>Spalte 1</th>
<th>Spalte 2</th>
<th>Spalte 3</th>
<th>Spalte 4</th>
<th>Spalte 5</th>
<th>Spalte 6</th>
</tr>
</thead>

<tfoot>
<tr>
<td colspan="6">Dies ist die Fußzeile der Tabelle</td>
</tr>
</tfoot>

<tbody>
<tr>
<td>Zeile 1, Zelle 1</td>
<td>Zeile 1, Zelle 2</td>
<td>Zeile 1, Zelle 3</td>
<td>Zeile 1, Zelle 4</td>
<td>Zeile 1, Zelle 5</td>
<td>Zeile 1, Zelle 6</td>
</tr>

[... mehr Inhalt...]

</tbody>
</table>

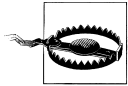
```

Diskussion

Die CSS-Spezifikationen 2.1 und 3 enthalten einige Eigenschaften, die speziell zur Darstellung von Tabellen vorgesehen sind. Dadurch ist es in vielen Fällen nicht mehr nötig, HTML-Attribute wie `border` oder `bordercolor` zu benutzen, stattdessen können Sie die Darstellung direkt im Stylesheet steuern.

Diese Möglichkeiten können besonders dann wertvoll sein, wenn Sie bestehende Tabledesigns überarbeiten möchten, ohne groß den HTML-Code anpassen zu wollen. Wurde beispielsweise für die Tabelle mithilfe des `border`-Attributs ein Rahmen definiert, können Sie diesen ohne Probleme im Stylesheet umdefinieren, ohne die Seite selbst ändern zu müssen.

Wir haben in unserer Lösung die Elemente `thead`, `tbody` und `tfoot` benutzt, um die Struktur der Tabelle genauer zu definieren und um Zellen im Stylesheet einzeln auswählen zu können. Außerdem haben wir für die Gestaltung der letzten Spalte des CSS3-Selektors `:last-child` benutzt. Dieser wird allerdings nur von modernen Browsern unterstützt. Damit auch ältere Versionen des Internet Explorer in den Genuss Ihres Designs kommen, können Sie beispielsweise eine JavaScript-Bibliothek wie *selectivizr* einsetzen.



In Webkit-Browsern (Safari, Chrome usw.) gibt es einen Programmierfehler. Dieser sorgt dafür, dass sich die untere Rahmenkante über alle Zellen erstreckt, die in `tfoot` für die erste Zelle definiert wurde.

Bei der Definition der Stile für »aktive« Zeilen sollten Sie daran denken, dass der `:hover`-Selektor eine höhere Spezifität besitzen muss als die allgemeine Definition für Zeilenhintergründe. Ansonsten wird nur jede zweite Zeile beim Überfahren mit der Maus tatsächlich hervorgehoben.

Wir haben das `tfoot`-Element in unserem HTML-Beispiel vor die Definition des Tabellenkörpers (`tbody`) gestellt, um eine möglichst hohe Browserunterstützung zu erreichen. In neueren Versionen der HTML5-Spezifikation kann `tfoot` aber auch nach `tbody` stehen.²

Siehe auch

Eine Sammlung verschiedener Tabellenlayouts unter <http://icant.co.uk/csstablegallery/tables/1.php>; die Website der JavaScript-Bibliothek *selectivizr* unter selectivizr.com; die Rezepte 2.5 und 2.6 zur Definition von Außen- und Innenabständen per `margin` bzw. `padding`; das Rezept 2.7 zur Definition von Rahmen; das Rezept 8.2 zu den Eigenschaften `border-collapse` und `border-spacing`; die Rezepte 3.1 und 3.4 zu den Eigenschaften `font-family` und `font-weight`; das Rezept 8.7 zur Verwendung der Eigenschaft `caption-side`; die Rezepte 2.8 bis 2.15 zur Definition von Hintergründen; das Rezept 1.14 zu Pseudoklassen (`:first-child`, `:last-child`, `:nth-child()`, `:hover`) und das Rezept 1.15 zu den möglichen Werten für die CSS3-Pseudoklasse `:nth-child()`; das Rezept 8.4 zur Gestaltung von einzelnen Tabellenzellen (`td`); das Rezept 8.6 zur Gestaltung von Tabellenüberschriften (`th`); das Rezept 12.1 zur Definition von Rollover-Effekten; das Rezept 3.23 zur Verwendung der Eigenschaft `text-shadow`; das Rezept 12.7 zur Definition von Schattenwürfen für bestimmte Elemente mit der Eigenschaft `box-shadow`; das Rezept 2.3 zur Definition von Farben; das Rezept 13.18 zur Verwendung von Polyfills; das Rezept 1.20 zum Thema Spezifität.

² In der Version vom 6. August 2011 war die Formulierung recht komplex. Unter <http://www.whatwg.org/specs/web-apps/current-work/multipage/tabular-data.html#the-tfoot-element> finden Sie weitere Informationen.

Einführung

Primär werden Webdokumente für die Anzeige auf einem Bildschirm erstellt. Sollen die Seiten dagegen auf einem Drucker ausgegeben werden, sind Dinge wie Navigationsleisten, Werbebanner oder große mehrfarbige Logos wenig nützlich oder sogar ganz überflüssig.

Ohne CSS müssten Sie den Text einer Website manuell von HTML-Elementen befreien, die beim Ausdrucken nicht dargestellt werden sollen. Mit CSS lassen sich Ihre Dokumente dagegen automatisch für den Ausdruck umwandeln. Der zusätzliche Zeitaufwand und der Bedarf an weiteren Ressourcen des Webserver, die normalerweise nötig wären, um die Druckversion beispielsweise über ein Skript zu erzeugen, entfallen.

Der CSS-Medientyp `print` wird von allen aktuellen Browsern unterstützt. Dabei sind einige CSS-Eigenschaften nur für die Steuerung von Ausdrucken vorgesehen, die von den Browsern unterschiedlich gut unterstützt werden.

Aufgrund dieser Tatsache und des Anspruchs dieses Buchs, sich an praktischen und browserübergreifenden Techniken zu orientieren, befassen sich die Rezepte in diesem Kapitel eher mit passenden Stildefinitionen für den Druck als mit den speziellen CSS-Eigenschaften für Druckmedien. Nähere Informationen zu diesen Eigenschaften finden Sie in *CSS – Das umfassende Handbuch* von Eric A. Meyer (O'Reilly 2007).

Bei der Erstellung von Stylesheets für das Ausdrucken gibt es eine Reihe von Besonderheiten, die direkt oder indirekt mit dem Ausgabemedium Drucker zu tun haben. So sind manche Maßeinheiten besser geeignet als andere. Gleichzeitig gibt es andere andere Dinge, wie beispielsweise Seitenumbrüche, die in der Bildschirmdarstellung keine Rolle spielen.

9.1 Spezielle Stylesheet-Regeln für das Ausdrucken angeben

Problem

Es soll eine druckfreundliche Seite erstellt werden, ohne dass hierfür manuell oder dynamisch ein weiteres HTML-Dokument angelegt werden muss.

Lösung 1

Verwenden Sie eine `@import`-Regel mit dem passenden Medientyp in einem externen Stylesheet, um ein weiteres Stylesheet zu laden, das die CSS-Regeln für die Druckversion enthält:

```
@import url('druck.css') print;
```

Im Gegensatz zum `link`-Tag wird die `@import`-Regel innerhalb eines Stylesheets festgelegt. Hierbei spielt es keine Rolle, ob die Regel in einem internen oder externen Stylesheet angegeben wurde.

Lösung 2

Verwenden Sie einen `@media`-Block, um die Regeln für das Ausdrucken im gleichen Stylesheet zu definieren wie die Anweisungen für die Bildschirmdarstellung. Am einfachsten und am besten zu warten ist diese Lösung, wenn Sie die Stile für den Ausdruck in einem externen Stylesheet definieren:

```
@media screen {  
  body {  
    font-size:      medium;  
    background-color: black;  
    color:          white;  
  }  
}  
  
@media print {  
  body {  
    font-size:      10pt;  
    background-color: white;  
    color:          black;  
  }  
}
```

Lösung 3

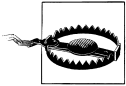
Verwenden Sie für jedes Ausgabemedium (hier: Bildschirm und Druck) ein eigenes `link`-Element. Über das HTML-Attribut `media` geben Sie jeweils den passenden Medientyp (`screen` bzw. `print`) an, wie hier gezeigt:

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

Diskussion

Um separate Stylesheets für die Bildschirmdarstellung und das Ausdrucken definieren zu können, verwenden Sie verschiedene Medientypen.

Neben dem am häufigsten benutzten Medientyp `screen` für die Darstellung von Dokumenten auf dem Bildschirm wurde in der CSS 2.1-Spezifikation auch der Medientyp `print` definiert. Dieser legt fest, wie das Dokument dargestellt werden soll, wenn es auf einem Drucker ausgegeben werden soll.



Ein häufiger Fehler bei der Erstellung von Druck-Stylesheets besteht darin, keinen ausdrücklichen Medientyp für das allgemeine Stylesheet anzugeben. Dadurch kann es passieren, dass bestimmte Stile für die Bildschirmdarstellung auch beim Ausdruck verwendet werden.

Die einfachste und wartbarste Methode ist in Lösung 1 beschrieben. Hierbei sind keinerlei Änderungen am HTML-Dokument notwendig. Das Druck-Stylesheet wird anhand einer `@import`-Regel mit dem passenden Medientyp eingebunden. Der Preis ist ein zusätzlicher HTTP-Request, um das Druck-Stylesheet zu laden.

Die zweite Möglichkeit besteht darin, die Regeln für das Ausdrucken im gleichen Stylesheet zu definieren wie die allgemeinen Anweisungen für die Darstellung, und sie mit passenden `@media`-Blöcken zu umgeben.

Der Vorteil dieser Methode besteht darin, dass kein zusätzliches Dokument vom Server abgerufen werden muss. Andererseits kann das Stylesheet bei diesem Verfahren recht schnell unübersichtlich werden.



Ältere Versionen der Screenreader-Software JAWS (vor 7.1) haben offenbar Probleme, wenn verschiedene `@media`-Blöcke im selben Stylesheet definiert werden.

Bei der Verwendung des `link`-Tags geben Sie den Medientyp in Form eines HTML-Attributs (`media`) an. Hierdurch wird ein eigenständiges Stylesheet mit den Regeln für das Ausdrucken geladen. Es ist allerdings nötig, Änderungen am HTML-Dokument vorzunehmen.

Siehe auch

Die Erläuterungen zu Medientypen im Abschnitt 7 der CSS 2.1-Spezifikation unter <http://www.w3.org/TR/CSS21/media.html>; das Rezept 1.23 zur Verwendung verschiedener Medientypen; die Website *printfriendly* mit einer JavaScript-basierten Lösung für das Ausdrucken von Seiten unter <http://www.printfriendly.com/>; einen Artikel über das Print-Stylesheet der Seitenvorlage *HTML5 Boilerplate* unter <http://designshack.net/articles/css/6-thinks-i-learned-about-print-stylesheets-from-html5-boilerplate/>.

9.2 Die richtigen Längeneinheiten und Werte für das Ausdrucken verwenden

Problem

Sie möchten wissen, welche Maßeinheiten und Werte für die Erstellung von Druck-Style-sheets geeignet sind.

Lösung

Verwenden Sie für die Angabe von Schriftgrößen nach Möglichkeit die Maßeinheit Punkt (pt). Für absolute Breiten- und Höhenangaben bieten sich »reale« Maßeinheiten wie Millimeter (mm) und Zentimeter (cm). Auch die Verwendung von Prozentwerten ist in Ordnung. Bedenken Sie aber, dass sich der Ansichtsbereich auf die ausgedruckte Seite bezieht.

Angaben in Pixeln sollten nach Möglichkeit vermieden werden, weil ein Pixel auf dem Bildschirm etwas vollkommen anderes ist als für einen Drucker. Hinzu kommt, dass ein Pixel je nach verwendetem Druckverfahren (Laser, Tintenstrahl usw.) ebenfalls eine unterschiedliche Bedeutung haben kann.

Diskussion

Ein wesentlicher Unterschied zwischen der Bildschirmdarstellung und der Ausgabe auf einem Drucker besteht darin, dass das Dokument auf eine oder mehrere Seiten fester Größe verteilt wird.

In unseren Breiten ist das meistens ein hoch- oder querformatiges Blatt Papier im Format DIN A4 (210mm x 297mm). Dies ist der maximal nutzbare Bereich. Der bedruckbare Bereich ist je nach Drucker meist etwas kleiner.

Standardmäßig werden Schriftgrößen beim Drucken in Punkt angegeben. Für normalen Fließtext werden meist Größen zwischen 12 und 16 Punkt verwendet. Überschriften sind entsprechend größer.

Siehe auch

Die Grundrezepte 2.1 und 2.2 zum Verständnis von Längeneinheiten und ihrer Verwendung; den Artikel *Going to Print* von Eric Meyer unter <http://www.alistapart.com/articles/goingtoprint/>.

9.3 Eine Seite für Schwarz-Weiß-Ausdrucke vorbereiten

Problem

Sie möchten das Farbschema einer Webseite im Druck-Stylesheet so anpassen, dass sie auch dann noch lesbar bleibt, wenn sie auf einem Schwarz-Weiß- oder Graustufendrucker ausgegeben wird.

Lösung

Verwenden Sie den universellen Selektor, um die Farben für Hintergründe, Schriften und Rahmen für alle Elemente zurückzusetzen:

```
* {  
  background: white !important;  
  color: black !important;  
  border-color: black !important;  
  
  box-shadow: none !important;  
  text-shadow: none !important;  
}
```

Durch diese einfache Regel erhalten alle Elemente einen weißen Hintergrund, während Schrift und Rahmen schwarz ausgedruckt werden. Außerdem entfernen wir alle möglicherweise vorhandenen Schattenwürfe. Zusätzlich haben wir die Deklarationen hier mit einer `!important`-Direktive ausgestattet.

Diskussion

Bei der Druckausgabe von Webseiten verändern einige Browser die Darstellung von Farben und Graustufen mit dem Zweck, den in der Seite enthaltenen Text leserlicher zu machen. Selbst wenn Sie angeben, dass bestimmte Teile beispielsweise hellgrau dargestellt werden sollen, erscheinen sie im Druck ganz in Schwarz. Da die Browser sich hierbei nicht einheitlich verhalten, ist es wenig sinnvoll, zu versuchen, diese Änderungen im Stylesheet auszugleichen.

Außerdem sind Ausdrucke, für die nur schwarze Tinte oder schwarzer Toner gebraucht wird, deutlich schneller, da keine Zwischenfarben gemischt werden müssen.

Auch die Ausgabe von weißer Schrift auf schwarzem Hintergrund ist kritisch, da dies meistens eine Menge Tinte/Toner kostet, ohne dass Text dadurch besser lesbar würde.

Die Darstellung von Schattenwürfen (schwarz-weiß oder farbig) für Elemente und Text macht das Lesen meist auch nicht leichter. Daher werden diese Gestaltungsmerkmale hier ebenfalls ausgeblendet.

Wie Sie sehen, haben wir in dieser Lösung alle Deklarationen mit einer `!important`-Direktive versehen. Auch wenn normalerweise diese Methode mit Bedacht verwendet werden sollte, hat sie in diesem Zusammenhang ihre Berechtigung.

Wenn Sie beispielsweise ein Druck-Stylesheet haben, das bestimmte Stile in einem allgemeinen Stylesheet ohne eigenen Medientyp überschreiben soll, kann es passieren, dass Regeln mit höherer Spezifität gegenüber den Regeln für das Ausdrucken Vorrang bekommen. Da die Regeln für das Ausdrucken keinen Einfluss auf die Bildschirmdarstellung haben, ist die Verwendung von `!important` in diesem Fall weitestgehend gefahrlos.

Siehe auch

Die Website von Joe Lewis mit einer Beschreibung der »farblichen Optimierung« von Websites durch verschiedene Browser unter <http://www.sanbeiji.com/archives/953>; den Artikel *When Using !important is the Right Choice* unter <http://css-tricks.com/when-using-important-is-the-right-choice/>; das Rezept 2.2 zur Definition von Farben; die Rezepte 2.8, 3.3 und 2.7 zu den Eigenschaften `background`, `color` und `border-color`; die Rezepte 12.7 und 3.23 zu den Eigenschaften `box-shadow` und `text-shadow`; das Rezept 1.21 zur `!important`-Direktive.

9.4 Nach einem Link den URI anzeigen

Problem

Wenn die Seite gedruckt wird, sollen nach einem Link die dazugehörigen URIs (Uniform Resource Identifiers) angezeigt werden.

Lösung

Weisen Sie den Browser mit dem Pseudoelement `:after` an, die URIs der Links in einem Absatz mit auszugeben:

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

Diskussion

Mithilfe des Pseudoelements `:after` (oder auch mit dem verwandten Selektor `:before`) und der Eigenschaft `content` können Sie nach (oder vor) einem Element automatisch erzeugte Inhalte einfügen. In diesem Rezept wird der Wert des Attributs `href`, das den URI enthält, der angezeigt werden soll, nach jedem `a`-Element innerhalb eines Absatzes eingefügt.

Die um den URI zusätzlich angezeigten spitzen Klammern müssen in der CSS-Regel mit Anführungszeichen umgeben werden. Um den URI visuell besser vom übrigen Inhalt abzusetzen, wird vor und nach den spitzen Klammern außerdem jeweils ein Leerzeichen eingefügt. Zwischen den Klammern steht die Funktion `attr()`, mit der wir den Wert des `href`-Attributs auslesen. Dieser wird nun als Zeichenkette in den erzeugten Inhalt eingebunden.

Cascading Stylesheets `<http://www.w3.org/Style/CSS/>`

Ein weiteres Beispiel für die Vielseitigkeit dieses Pseudoelements zeigt sich, wenn die Bedeutung einer Abkürzung angezeigt werden soll, sobald sich der Mauszeiger darüber befindet. Dazu muss das vollständige Wort im `title`-Attribut eines `abbr`- oder `acronym`-Elements stehen:

```
<p>Das <acronym title="World Wide Web Consortium">W3C</a> erfindet wunderbare  
Dinge wie zum Beispiel <abbr title="Cascading Style Sheets">CSS</abbr>!</p>
```

In einer CSS-Regel weisen Sie dann den Browser an, den Wert des Attributs `title` auszugeben:

```
abbr:after, acronym:after {  
    content: " (" attr(title) ") ";  
}
```

Vor Root-Links den Domainnamen anzeigen

Bei Links relativ zum Wurzelverzeichnis werden beim Ausdruck nur der Schrägstrich und die folgenden Verzeichnis- und Dateinamen angezeigt. Durch den Einsatz des Attributselektors für Substrings aus der CSS3-Spezifikation können Sie dieses Problem umgehen:

```
p a:after {  
    content: " <" attr(href) "> " ;  
}  
p a[href^="/"]:after {  
    content: " <http://www.csskochbuch.com" attr(href) "> " ;  
}
```

Das Caret-Zeichen (^) weist den Selektor an, nur Links zu finden, die mit einem Schrägstrich beginnen. Auf diese Weise können wir die Root-Links von relativen und absoluten Links unterscheiden und ihnen die nötigen Regeln zuweisen.

Momentan funktionieren mithilfe von `:before` und `:after` erzeugte Inhalte in allen aktuellen Browsern außer Internet Explorer 7. Die Verwendung des erweiterten Attributselektors für Substrings wird von allen aktuellen Browsern außer IE7 und 8 unterstützt.

Siehe auch

Das Rezept 1.14 zur Verwendung der Pseudoelemente `:before` und `:after`; das Rezept 13.7 zur Eigenschaft `content`; das Rezept 3.1 zur Definition von Schriften in einem Webdokument; die CSS 2.1-Spezifikation zu erzeugten Inhalten unter <http://www.w3.org/TR/REC-CSS2/generate.html#content>.

9.5 Sonderzeichen vor Links einfügen

Problem

Vor Links soll beim Ausdrucken eines Dokuments ein Sonderzeichen, z.B. », eingefügt werden.

Lösung

Wie bereits in Rezept 9.4 gezeigt wurde, verwenden Sie im Stylesheet für die Druckausgabe das Pseudoelement `:after`, um den URI nach dem tatsächlichen Link einzufügen:

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

Dieses Konzept erweitern wir jetzt, um vor dem URI das Sonderzeichen » darzustellen. Anstelle von `:after` benutzen wir dieses Mal `:before`, um den gewünschten Effekt zu erzielen:

```
p a:before {  
  text-decoration: underline;  
  content: " \00BB " attr(href);  
}
```

In der Druckversion wird der Link nun folgendermaßen dargestellt:

Das CSS Kochbuch » <http://www.csskochbuch.com/>

Diskussion

Vergessen Sie nicht, dem hexadezimalen Wert einen Backslash (\) voranzustellen, damit das Zeichen im Browser nicht als einfacher Text angezeigt wird. Der korrekte Wert für » lautet `\00BB`. Ohne den Backslash sähe die Darstellung so aus:

00BB <http://www.csskochbuch.com/>

Aufgrund der CSS-Syntax können Sie für die Eigenschaft `content` keine HTML-Entities (hier: `»` bzw. `»`) verwenden. Stattdessen wird der hexadezimale Wert des entsprechenden Unicode-Zeichens mit einem vorangestellten Backslash benutzt.

Diese Sonderzeichen können natürlich nicht nur für die Druckausgabe, sondern für alle Medientypen verwendet werden. Testen Sie das ruhig einmal in Ihrer Webseite. Damit die Ausgabe auch im Browser dargestellt wird, muss der Medientyp `all` oder `screen` lauten.

Die hier gezeigte Technik funktioniert in allen aktuellen Browsern außer Internet Explorer 7.

Siehe auch

Ein Onlinewerkzeug zum Ermitteln der nötigen Hex-Werte finden Sie beispielsweise unter <http://www.rishida.net/tools/conversion/>; weitere Erläuterungen zu Sonderzeichen in CSS gibt es unter <http://www.w3.org/TR/CSS21/syndata.html#escaped-characters>.

9.6 Seitenumbrüche anlegen

Problem

Normalerweise legen die Browser Seitenumbrüche nach Bedarf an. Sie möchten aber genauer steuern, an welchen Stellen Seitenumbrüche angelegt werden.

Lösung

Verwenden Sie die speziellen Eigenschaften `page-break-before` und `page-break-after`, um die Platzierung von Seitenumbrüchen genauer zu steuern. Um beispielsweise festzulegen, dass vor einer Überschrift prinzipiell ein Umbruch eingefügt werden soll, können Sie Folgendes schreiben:

```
h1, h2 {
    page-break-before: always;
}
```

Damit stehen Überschriften erster und zweiter Ordnung immer am Seitenanfang. Ohne Angabe dieses Werts wird der Standardwert `auto` verwendet, und der Browser steuert die Umbrüche selbst.

Um nach einem Element einen Seitenumbruch einzufügen, gibt es analog dazu die Eigenschaft `page-break-after`. Soll beispielsweise nach einer Liste auf jeden Fall eine neue Seite begonnen werden, könnten Sie diese Regel verwenden:

```
ul {
    page-break-after: always;
}
```

Manchmal kann es passieren, dass innerhalb eines Elements ein Umbruch eingefügt werden muss, weil es nicht ganz auf eine Druckseite passt. Mit der Eigenschaft `page-break-inside` können Sie den Browser anweisen, dies zu vermeiden, z.B.:

```
p {
    page-break-inside: avoid;
}
```

Diskussion

Die Eigenschaften `page-break-before` und `page-break-after` werden auch von aktuellen Browsern nur teilweise unterstützt. Bei Drucklegung dieses Buchs verstanden Chrome (17), Safari (5.1) und Firefox (10) nur die Werte `auto` und `always`. Im Internet Explorer 8+

und Opera 9+ funktionieren auch die Werte `avoid` (Umbruch vermeiden), `left` (nach Umbruch »linke« Seite ausgeben) und `right` (nach Umbruch »rechte« Seite ausgeben).

Im Gegensatz zu den vorigen beiden Eigenschaften wird `page-break-inside` von allen aktuellen Browsern recht gut unterstützt. Neben dem Standardwert `auto` (der Browser entscheidet selbst) und `inherit` (Wert wird geerbt) gibt es hier allerdings auch nur den Wert `avoid` (der Browser entscheidet ebenfalls selbst, versucht aber, einen Umbruch im Element zu vermeiden).

Die Eigenschaften `page-break-before`, `page-break-after` und `page-break-inside` können nur auf Blockelemente angewendet werden, die nicht als `Float` definiert wurden oder von einem `Float` umgeben sind. Als Werte für `position` sind nur der Standardwert `static` oder `relative` möglich. Ansonsten wird die Deklaration nicht berücksichtigt.

Von Witwen und Waisen

Gelegentlich kommt es vor, dass eine einzelne Zeile eines Abschnitts am Ende einer Seite stehen bleibt, während der restliche Text des Abschnitts bereits auf der folgenden Seite steht. Diese einzelne Zeile wird fachsprachlich als »Waise« (engl. *orphan*, auch als »Schusterjunge« bekannt) bezeichnet. Wenn hingegen der meiste Text eines Abschnitts am Ende einer Seite steht und lediglich eine einzelne Zeile auf die Folgeseite rutscht, spricht man von einer »Witwe« (engl. *widow*, wird auch »Hurenkind« genannt).

Mit der Eigenschaft `orphans` geben Sie die minimale Anzahl von Textzeilen innerhalb eines Elements an, die allein am unteren Ende einer Seite stehen dürfen.

Entsprechend können Sie mit der Eigenschaft `widows` die minimale Anzahl von Textzeilen festlegen, die allein am oberen Ende einer Seite stehen dürfen. Hier ein Beispiel:

```
p { widows: 4; }
ul { orphans: 2; }
```

Muss in einem Absatz ein Seitenumbruch eingefügt werden, sollen auf der folgenden Seite mindestens vier Textzeilen stehen. Passen beim Ausdrucken einer Liste nicht alle Einträge auf eine Seite, sollen auf der vorangehenden Seite mindestens zwei Einträge stehen.

Die Eigenschaften `orphans` und `widows` können in allen aktuellen Browsern außer Firefox 10 und anderen Gecko-basierten Programmen benutzt werden.

Siehe auch

Den englischen Wikipedia-Artikel mit einem Vergleich der verschiedenen Layout-Engines unter [http://en.wikipedia.org/wiki/Comparison_of_layout_engines_\(Cascading_Style_Sheets\)#gecko_page-break-before-after](http://en.wikipedia.org/wiki/Comparison_of_layout_engines_(Cascading_Style_Sheets)#gecko_page-break-before-after); den Mozilla Bug-Tracker zu den Eigenschaften `page-break-before`/`page-break-after` und `orphans`/`widows` unter https://bugzilla.mozilla.org/show_bug.cgi?id=132035 bzw. https://bugzilla.mozilla.org/show_bug.cgi?id=137367.

9.7 Überschriften von Tabellenspalten auf jeder Seite mitdrucken

Problem

Eine Tabelle erstreckt sich beim Ausdrucken über mehrere Seiten. Um beim Lesen den Überblick nicht zu verlieren, sollen die Spaltenüberschriften auf jeder Seite wiederholt werden.

Lösung

Damit die Tabellenüberschriften auf jeder Seite erneut erscheinen, müssen sie im HTML-Code mit `thead`-Tags umgeben werden, wie hier gezeigt:

```
<table>
<caption>Eine Tabelle mit Stil</caption>

<thead>
  <tr>
    <th>Spalte 1</th>
    <th>Spalte 2</th>
    <th>Spalte 3</th>
  </tr>
</thead>
<tbody>
  ...
</tbody>
</table>
```

Sollen auch ältere Versionen des Internet Explorer (vor Version 7) unterstützt werden, müssen Sie die Darstellungsrolle ausdrücklich angeben:

```
thead {
  display: table-header-group;
}

tbody {
  display: table-row-group;
}
```

Diskussion

Die hier gezeigte Lösung funktionierte beim Schreiben dieses Buchs nur in Firefox und Internet Explorer, dort aber auch in älteren Versionen. Opera und Webkit (Safari, Chrome usw.) boten bei Drucklegung dieses Buchs noch keine Unterstützung für die Wiederholung der Spaltenüberschriften.

Siehe auch

Den Blogeintrag von Kristin Bradley zum Thema unter <http://kristinbradley.com/blog/2011/06/15/print-table-headers-on-every-page/>; das Wiki der *css-discuss*-Mailingliste unter http://css-discuss.incutio.com/wiki/Printing_Tables; eine Anleitung zur Erstellung von wiederholten Spaltenüberschriften unter <http://www.xefteri.com/articles/show.cfm?id=26>; den Bug-Report für Webkit-Browser unter https://bugs.webkit.org/show_bug.cgi?id=34218; die Website *Browsersupport.net* zum Wert `table-header-group` für die Eigenschaft `display` unter <http://www.browsersupport.net/CSS/display%3Atable-header-group>; das Kapitel 8, *Tabellen*, zur Gestaltung von Tabellen mit CSS.

9.8 Ein Webformular für den Ausdruck aufbereiten

Problem

Ein Formular, das online von Benutzern ausgefüllt werden kann, soll so aufbereitet werden, dass es auch ausgedruckt manuell auszufüllen ist. In Abbildung 9-1 sehen Sie das Formular in der Webdarstellung.

The screenshot shows a web browser window with the title "CSS Kochbuch: Formulare". The browser's address bar is empty. The form itself is a two-column layout. The first column, labeled "Schritt 1: Persönliche Angaben", contains input fields for "Vorname:" (filled with "Jørgen"), "Nachname:" (filled with "Lang"), "Straße/Hausnummer:" (filled with "Strasse/Hausnummer"), "Stadt:" (filled with "Stadt"), "PLZ:" (filled with "PLZ"), and "Land:" (filled with "D"). The second column, labeled "Schritt 2: Kontaktinformationen", contains input fields for "Telefon (privat):" (filled with "Private Telefonnummer"), "Telefon (geschäftlich):" (filled with "Gesch. Telefonnummer"), "Telefon (mobil):" (filled with "Mobiltelefon"), "E-Mail:" (filled with "E-Mail-Adresse"), and "Website:" (filled with "Website-Adresse"). At the bottom, labeled "Schritt 3: Abschicken", there are two buttons: "Abschicken" with a checkmark icon and "Zurücksetzen" with a cross icon.

Abbildung 9-1: Ein Onlineformular

Lösung

Legen Sie ein separates Stylesheet für den Ausdruck an und nehmen Sie dort die nötigen Anpassungen vor. Als Ausgangsbasis dient uns das zweispaltige Formular aus Rezept 7.16. Für den Druck sollen alle Felder untereinander angeordnet werden. Das Stylesheet wird in einer separaten Datei gespeichert und anschließend per `@import`-Regel in das HTML-Dokument eingebunden. Hier die nötigen CSS-Regeln:

```
* {  
    background: white;  
    color: black;  
    border-color: black;  
}
```

Durch den universellen Selektor (*) sorgt die obige Regel dafür, dass alle Elemente einheitlich mit einem weißen Hintergrund versehen werden. Als Schriftfarbe und Farbe für möglicherweise definierte Rahmen wählen wir Schwarz. (Bei Bedarf können Sie in dieser Regel auch mögliche Schattenwürfe ausblenden, wie in Rezept 9.3 beschrieben wird.)

```
fieldset {  
    margin-bottom: 1cm;  
    border: 1px solid black;  
}
```

Um zwischen den beiden `fieldset`-Elementen etwas Platz zu schaffen, definieren wir einen 1 Zentimeter breiten unteren Außenabstand. Da das Formular ausgedruckt werden soll, sind absolute Maßeinheiten (siehe Rezept 2.1, 2.2) meistens besser geeignet.

Eine Ausnahme bildet die Haarlinie, die um die gruppierten Formularfelder erzeugt werden soll. Selbst ein Millimeter wäre zu dick. Daher benutzen wir hier die kleinste mögliche Einheit, 1 Pixel, um die Rahmenbreite anzugeben.

```
fieldset:last-child {  
    display: none;  
}
```

Die Buttons im Onlineformular werden in der Druckausgabe nicht benötigt. Daher blenden wir das gesamte `fieldset` per `display: none` einfach aus. Damit diese Regel nur für das letzte `fieldset` gilt, verwenden wir die Pseudoklasse `:last-child` aus der CSS3-Spezifikation, die von den meisten aktuellen Browsern (im Internet Explorer ab Version 9) bereits unterstützt wird.

Als Nächstes kommen wir zu den Eingabefeldern und ihren Beschriftungen:

```
label, input {  
    font-size: 12pt;  
}
```

Durch die Gruppierung der Selektoren können wir für Beschriftungen (`label`) und Eingabefelder (`input`) die Schriftgröße in einer Regel mit 12 Punkten angeben.

Es soll immer ein Paar aus Beschriftung und Eingabefeld auf einer Zeile stehen. Daher bietet sich hier die Verwendung der Eigenschaft `float` und die Vergabe fester Breiten für beide Elemente an:

```
label {  
  float: left;  
  clear: left;  
  width: 5cm;  
  
  position: relative;  
  top: 2mm;  
  text-align: right;  
}
```

Die Kombination aus `float` und `clear` sorgt dafür, dass die Beschriftung möglichst weit nach links verschoben wird und so zusammen mit der Breitenangabe von 5 Zentimetern genug Platz für das nachfolgende `input`-Element schafft.

Die Angabe `clear: left` legt gleichzeitig fest, dass `label`-Elemente andere Floats nicht umfließen sollen. Es entsteht ein impliziter »Zeilenumbruch«.

Wir wollen in der folgenden Regel die Eigenschaft `border-bottom` benutzen, um eine gestrichelte Linie zu erzeugen. Da Rahmen anders als Textausschmückungen per `text-decoration` nicht auf der Grundlinie erzeugt werden, müssen wir diesen Versatz ausgleichen und die Beschriftung etwas tiefer darstellen als das Eingabefeld.

Hierfür benutzen wir die Eigenschaft `position` mit dem Wert `relative`, um die Beschriftung per `top: 2mm` um 2 Millimeter nach unten zu verschieben. Dadurch erscheint die Grundlinie des `label`-Elements bündig mit der unteren Rahmenkante des folgenden `input`-Elements.

Zusätzlich sorgen wir für etwas mehr Symmetrie, indem wir den Beschriftungstext rechtsbündig ausrichten.

```
input {  
  position: relative;  
  left: 0.5cm;  
  width: 10cm;  
  
  float: left;  
  color: transparent;  
  
  border-bottom: 1px dotted black;  
}
```

Für die Eingabefelder verwenden wir eine ähnliche Methode, um sie ein wenig von ihrer Beschriftung abzurücken. Ebenfalls relativ positioniert, werden sie per `left: 0.5cm` ein Stück nach rechts verschoben und erhalten eine Breite von 10 Zentimetern.

Abschließend erzeugen wir die gepunktete Linie, die dem Benutzer das Ausfüllen des Formulars erleichtern soll.



In den meisten Fällen werden Webseiten hierzulande hochkant im DIN-A4-Format ausgedruckt. Die Breitenangaben für Beschriftung und Eingabefelder orientieren sich an diesen Standardmaßen. (Ein DIN-A4-Blatt hat eine Breite von 21 und eine Höhe von 29,7 Zentimetern.)

Unser Webformular verwendet bereits das HTML5-Attribut `placeholder`, um dem Besucher beim Ausfüllen der Onlineversion zu helfen. Um den Platzhaltertext in Webkit-Browsern wie Chrome und Safari nicht mit auszudrucken, wenden wir einen Trick an:

```
input::-webkit-input-placeholder {  
    color: transparent;  
}
```

Chrome und Safari benutzen intern den speziellen Selektor `::-webkit-input-placeholder`, um Stile für den Platzhaltertext zu definieren. Das machen wir uns zunutze, um den Text einfach per `color: transparent` bei der Druckausgabe auszublenden. In Firefox wird der Platzhaltertext beim Ausdrucken nicht mit ausgegeben.

Das druckfertige Formular sehen Sie in Abbildung 9-2.

Vorschau von „CSS Kochbuch– Formulare“.pdf (1 Seite)

Vorherige Nächste Zoomen Bewegen Text Auswählen Anmerken Seitenleiste

Schritt 1: Persönliche Angaben

Vorname:
Nachname:
Straße/Hausnummer:
Stadt:
PLZ:
Land:

Schritt 2: Kontaktinformationen

Telefon (privat):
Telefon (geschäftlich):
Telefon (mobil):
E-Mail:
Website:

Abbrechen Drucken

Abbildung 9-2: Das gleiche Formular in der Druckdarstellung

Um das Druck-Stylesheet in Ihr Webdokument einzubinden, verwenden Sie im head-Teil der Seite ein `link`-Element oder eine der anderen in Rezept 9.1 gezeigten Methoden. Vergessen Sie nicht, den passenden Medientyp `print` anzugeben.

Diskussion

Die Linien in einem Formular zeigen dem Benutzer an, an welchen Stellen das Formular ausgefüllt werden soll. Mit der Eigenschaft `border` lassen sich diese Linien leicht in einem Browser erzeugen. Dadurch können Webformulare bequem online und offline genutzt werden.

Weil die meisten Browser die Eigenschaft `border` allerdings auch verwenden, um Eingabefeldern ihr typisches Aussehen zu geben, müssen wir sie im Druck-Stylesheet zuerst zurücksetzen. Danach können die gewünschten Linien per `border-bottom` erzeugt werden:

```
input {  
    border: none;  
    border-bottom: 1px dotted black;  
}
```

Um die Grundlinie der Beschriftung mit der erzeugten Linie auf gleiche Höhe zu bringen, benutzen wir in der Lösung die Eigenschaft `position` mit dem Wert `relative`, um die benötigte Verschiebung zu erreichen. Der Grund liegt darin, dass die Eigenschaft `padding` durch die Definition von `label` und `input` als Floats hier nicht funktioniert.

Mehr Benutzerfreundlichkeit

Da der *Abschicken*-Button in der Druckversion des Formulars fehlt, benötigt der Benutzer zusätzliche Informationen darüber, wie er Ihnen das Formular nach dem Ausdrucken und Ausfüllen zukommen lassen kann. Um eine Postadresse in das Formular zu integrieren, könnten Sie das Webdokument wie folgt erweitern:

```
<div id="adresse-fuer-ausdruck">  
<p>Um uns das Formular zuzusenden, schicken Sie es bitte an die folgende Adresse:</p>  
  
<div class="vcard">  
  <div class="org">Die Firma GmbH & Co. KG</div>  
  <div class="adr">  
    <div class="street-address">Beispielstraße 81</div>  
    <span class="postal-code">50670</span>  
    <span class="locality">Beispielstadt</span>,&br/>    <span class="country-name">Deutschland</span>  
  </div>  
</div>  
</div>
```




Wir verwenden hier das für Adressen übliche Mikroformat. Aus diesem Grund ist die Struktur der Elemente etwas komplizierter als eigentlich nötig, und die Klassen tragen englische Bezeichnungen.

Damit dieses `div`-Element in der Onlinedarstellung *nicht* auftaucht, muss es im Stylesheet für den Medientyp `screen` ausdrücklich ausgeblendet werden:

```
#adresse-fuer-ausdruck { display: none; }
```

Internet Explorer

Eine Kleinigkeit gibt es noch zu tun. Da die Versionen 7 und 8 des Internet Explorer die Pseudoklasse `:last-child` noch nicht verstehen, müssen wir hier ein wenig nachhelfen. Damit auch hier die Buttons nicht mit ausgedruckt werden, erstellen wir eine zusätzliche CSS-Datei mit folgendem Inhalt:

```
fieldset:first-child + fieldset + fieldset {  
    display: none;  
}  
  
fieldset {  
    padding-bottom: 0.5cm;  
}
```

Diese Datei können Sie anschließend mit einem `link`-Tag (siehe Rezept 9.1) in Ihr Webdokument einbinden. Hierbei ist es empfehlenswert, die Anweisung mit Conditional Comments zu umgeben, damit sie nur von IE7 und 8 geladen werden:

```
<!--[if lte IE 8]>  
<link rel="stylesheet" href="druck-ie.css" media="print" charset="utf-8" />  
<![endif]-->
```

Anstelle von `:last-child` benutzen wir hier die Pseudoklasse `:first-child` und den Selektor für benachbarte Geschwisterelemente (`+`), um für das dritte `fieldset`-Element eine eigene Regel zu definieren.

Und weil wir gerade dabei sind, können wir auch gleich ein weiteres Darstellungsproblem lösen. In IE7 und 8 ist der Abstand zwischen der letzten Zeile und der Unterkante des umgebenden `fieldset`-Elements zu gering. Das gleichen wir aus, indem wir dem `fieldset` per `padding-bottom` etwas zusätzlichen Innenabstand verschaffen.

Siehe auch

Das Grundrezept 2.7 zur Verwendung der Eigenschaft `border`; das Rezept 7.7 mit weitergehenden Informationen zu den Präfixselektoren in Gecko und Webkit; das Rezept 13.13 zur Verwendung von Conditional Comments; den Wikipedia-Eintrag zu Mikroformaten unter <http://de.wikipedia.org/wiki/Mikroformate>.

9.9 Bestimmte Dokumentteile nicht mit ausdrucken

Problem

Bestimmte Teile eines Webdokuments sollen nicht mit ausgedruckt werden.

Lösung

Erstellen Sie ein spezielles Druck-Stylesheet, das die gewünschten Teile beim Ausdrucken ausblendet, zum Beispiel:

```
#kopfteil .diaschau, #navigation, #seitenleiste, #blog-kommentare, .werbung {  
    display: none;  
}
```

Natürlich können Sie auch alle anderen CSS-Selektoren verwenden, um die Dokumentteile auszuwählen, die in gedruckten Seiten nicht erscheinen sollen. Sollen Bilder und Multimedia-Inhalte bzw. deren Platzhalter nicht mitgedruckt werden, könnten Sie beispielsweise schreiben:

```
img, object, embed, video, audio {  
    display: none;  
}
```

Das Stylesheet können Sie mit einer der in Rezept 9.1 beschriebenen Methoden in Ihre Seiten einbinden. Vergessen Sie nicht, das allgemeine Stylesheet und das Druck-Stylesheet mit dem passenden Medientyp zu kennzeichnen.

Diskussion

Bestimmte Inhalte einer Webseite sind beim Ausdrucken nicht nötig. Dinge wie zum Beispiel Navigationselemente, jQuery-Slider oder auch Blogkommentare sind in der Papierversion oft nicht nötig. Und interaktive Komponenten wie Video- und Audiokomponenten lassen sich auf Papier höchstens durch eine Platzhaltergrafik oder einen entsprechenden Text darstellen.

Formulare können dagegen so aufbereitet werden, dass sie auch im Ausdruck sinnvoll sind, wie in Rezept 9.8 beschrieben wird.

Auf vielen Websites findet man in diesem Zusammenhang den Rat, die nicht benötigten Teile mit einer speziellen Klasse zu kennzeichnen. Das kann allerdings schnell zu einem wartungstechnischen Albtraum werden. Besser ist es, wenn Sie, wie hier gezeigt, bereits vorhandene IDs (und Klassen) verwenden, um die gewünschten Elemente auszuwählen und auszublenden.

Sollte das nicht funktionieren, lassen sich die meisten Bestandteile einer Seite aber auch anhand der verschiedenen Selektoren bzw. deren Kombination auswählen.

Da sich durch diese Maßnahmen das Layout erheblich ändern kann, ist es unter Umständen sinnvoll, Elemente, die als Floats definiert wurden, wieder in den normalen Dokumentenfluss zu integrieren. Da für Floats fast immer auch eine Breite angegeben wird, sollten Sie diese gleich mit zurücksetzen:

```
#hauptspalte {  
    float: none;  
    width: auto;  
}
```

Hier erhält die Hauptspalte ihre normale Position innerhalb des Dokuments zurück. Die Breite wird auf den Standardwert `auto` zurückgesetzt, wodurch der vorhandene horizontale Platz so weit wie möglich ausgenutzt wird.

Der Versuch, den Ausdruck einer Seite komplett zu unterbinden

Wenn Sie das Ausdrucken einer Seite verhindern möchten, könnten Sie versuchen, die Anweisung `display: none` einfach für das `body`-Element zu definieren:

```
@media print {  
    body { display: none; }  
}
```

Das lässt sich allerdings leicht umgehen, indem Sie als Besucher der Seite die Verwendung von Stylesheets im Browser deaktivieren. Und selbst wenn das nicht möglich ist, können Sie immer noch einen Screenshot der Bildschirmdarstellung anfertigen und diesen dann ausdrucken.

Oder aber Sie speichern die Seite als HTML-Datei auf Ihrem Computer. Möglicherweise in den Quelltext eingebettete Anweisungen, die das Ausdrucken verhindern sollen, lassen sich dann leicht entfernen. Danach können Sie die Seite mit dem Browser Ihrer Wahl öffnen und ausdrucken.

Fazit: Die obige Regel schafft nur vermeintliche Sicherheit. Genauso wenig, wie Sie verhindern können, dass Benutzer mit ausreichend Know-how die Bilder Ihrer Website herunterladen, können Sie verhindern, dass Ihre Webdokumente auf die eine oder andere Art ausgedruckt werden.

Siehe auch

Das Rezept 1.12 zur Verwendung von Klassen- und ID-Selektoren; das Rezept 1.6 zu Elementselektoren; das Rezept 1.17 zur Eigenschaft `display` und den verschiedenen Darstellungsrollen; das Rezept 1.23 zur Verwendung von Medientypen; das Rezept 9.8 zur Aufbereitung von Formularen für das Ausdrucken.

9.10 Das Ausdrucken von Hintergrundbildern selbst steuern

Problem

In den meisten Browsern ist das Ausdrucken von Hintergrundbildern standardmäßig deaktiviert. Sie möchten die Darstellung aber selbst bestimmen.

Lösung 1

Lassen Sie Ihre Besucher selbst entscheiden, ob sie Hintergrundbilder mitdrucken wollen oder nicht. Ganz im Ernst – was würden Sie denn sagen, wenn ein Webdesigner die komplette Website mit einem schwarzen Hintergrund hinterlegt hat und Ihrem Drucker nach der zehnten Seite die Tinte ausgeht?

Die meisten Browser bieten im Druckdialog ein Ankreuzfeld, das vor dem Ausdruck aktiviert werden kann, um Hintergrundbilder mitzudrucken, wie in Abbildung 9-3 zu sehen.



Abbildung 9-3: Das Ausdrucken von Hintergrundbildern kann vom Benutzer selbst gesteuert werden

Lösung 2

Also gut, natürlich könnten Sie dem Browser das Hintergrundbild als erzeugten Inhalt »unterjubeln«, wie in Abbildung 9-4 gezeigt.

Definieren Sie hierfür einen @media-Block, in dem Sie die nötige Regel angeben:

```
@media print {  
  body:after {  
    display: block;  
    position: absolute;  
  
    left: 0;  
    top: 0;  
  
    z-index: -10;  
  
    height: 100%;  
    width: 100%;  
  
    content: url('bilder/hintergrund.jpg');  
  }  
}
```

Zu Beginn wird ein eigener @media-Block definiert, der nur zur Anwendung kommt, wenn das Dokument ausgedruckt wird.

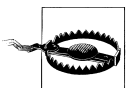
Wir benutzen hier das Pseudoelement :after, um das gewünschte Bild in die Seite einzufügen, ohne den HTML-Code selbst ändern zu müssen. Nach der Definition als Blockelement wird es absolut positioniert und in der linken oberen Ecke des Ansichtsbereichs verankert.

Durch die Angabe von z-index: -10 soll sichergestellt werden, dass das Bild nicht versehentlich vor den eigentlichen Inhalten der Seite angezeigt wird.



Der Wert -10 ist willkürlich gewählt. Sofern Sie die Eigenschaft in der Seite ansonsten nicht benutzen, würde auch die Angabe z-index: -1 ausreichen. Sobald Ihr Stylesheet weitere Angaben zur Stapelungsreihenfolge anderer Elemente enthält (siehe Rezept 2.21), müssen Sie allerdings sicherstellen, dass das »Hintergrundbild« den niedrigsten Wert für z-index besitzt. Ansonsten werden möglicherweise andere Elemente verdeckt.

Durch die Angabe von height und width wird das vermeintliche Hintergrundbild außerdem so weit gestreckt und gedehnt, dass der gesamte Inhaltsbereich davon bedeckt wird. In der darauffolgenden Deklaration wird das Bild anhand der Eigenschaft content schließlich eingefügt.



Diese Methode funktionierte zur Drucklegung dieses Buchs nur in neueren Versionen von Safari, Chrome und Firefox. Außerdem kann es je nach Motiv des Hintergrundbilds zu unschönen Verzerrungen kommen.

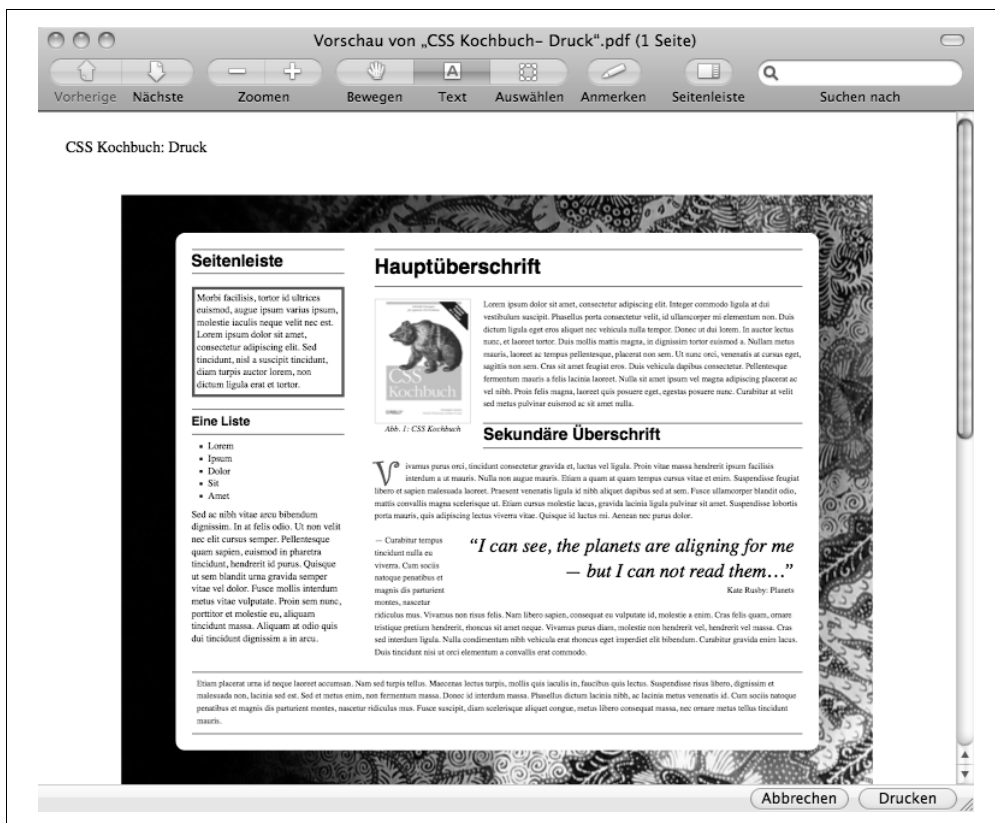


Abbildung 9-4: Das Hintergrundbild wird als erzeugter Inhalt definiert

Diskussion

Das Ausdrucken von Hintergründen ist in den meisten Browsern aus gutem Grund deaktiviert. Meistens leidet nicht nur die Lesbarkeit, sondern auch der Vorrat an Druckertinte Ihrer Benutzer. Üblicherweise kann die Ausgabe von Hintergrundfarben und -bildern im Druckdialog des Browsers aber vom Benutzer ein- und ausgeschaltet werden.

Die in Lösung 2 gezeigte Methode funktioniert nur in manchen modernen Browsern. Wenn Sie sich entscheiden, dieses Verfahren einzusetzen, sollten Sie sich vorher ganz genau darüber klar sein, ob die Ausgabe von Hintergrundbildern wirklich notwendig bzw. erwünscht ist.

Siehe auch

Das Rezept 1.23 zu den verschiedenen Medientypen; das Rezept 1.14 zur Verwendung der Pseudoelemente `:before` und `:after`; das Wiki der Mailingliste [css-discuss](http://css-discuss.incutio.com/wiki/Print_Stylesheets_Backgrounds) zum Drucken von Hintergrundbildern unter http://css-discuss.incutio.com/wiki/Print_Stylesheets_Backgrounds.

Mobile Geräte

Einführung

Die Bedeutung von Smartphones hat in den letzten Jahren stark zugenommen, und auch künftig wird der Markt für mobile Endgeräte weiter wachsen.

Im Januar 2012 waren Android- und iOS-Geräte die Haupt-Player in diesem Markt, wobei die Verbreitung von Android-Smartphones die von iOS-Geräten bereits überflügelt.

Obwohl über 80% der mobilen Internetnutzung mit Webkit-Browsern stattfindet, gibt es teilweise erhebliche Unterschiede zwischen den Browsern. Außerdem steigt bei diesen Zahlen die Gefahr der Monokultur. Das heißt, selbst wenn Sie primär für mobile Plattformen entwickeln, sollten Sie auch andere Browser berücksichtigen.

Auf den folgenden Seiten finden Sie Rezepte, die Ihnen bei der Erstellung, Anpassung und Optimierung Ihrer Seiten für das mobile Web helfen sollen.

10.1 Media Queries verwenden, um mobile und Desktop-Browser zu unterscheiden

Problem

Sie möchten spezielle Stylesheets für die Browser von Mobilgeräten erstellen.

Lösung

Verwenden Sie CSS3-Medienabfragen (*Media Queries*), um Details über das verwendete Gerät zur Unterscheidung zu erhalten. Damit ein Stylesheet speziell für Smartphones ausgeliefert werden kann, können Sie beispielsweise schreiben:

```
@import url("max480px.css") only screen and (max-width: 480px);
```

Im Prinzip handelt es sich hier um eine einfache `@import`-Regel, wie sie in Rezept 1.1 beschrieben wird.



Das Schlüsselwort `only` am Anfang der Deklaration sorgt dafür, dass ältere Browser, die noch keine Medienabfragen verstehen, die Deklaration schlicht ignorieren. Auf diese Weise können Sie verhindern, dass ein mobiles Stylesheet versehentlich geladen wird.

Auf die Angabe des Medientyps `screen` folgt die eigentliche Medienabfrage. Sie ergänzt die zuvor gemachten Angaben um weitere Details. In unserem Fall soll das Stylesheet `max480px.css` nur geladen werden, wenn der Bildschirm nicht breiter ist als 480 Pixel. Das entspricht beispielsweise einem iPhone im Querformat (Landscape).

Üblicherweise verwenden Sie eine solche Anweisung nicht allein, sondern im Zusammenhang mit dem »Responsive Webdesign«. Hierbei werden unterschiedliche Stylesheets beispielsweise für verschiedene Bildschirmbreiten/-höhen definiert.



Weitere Informationen zum Ermitteln der tatsächlichen Bildschirmbreite finden Sie in Rezept 10.3.

Diskussion

Aktuellen Statistiken zufolge sind mittlerweile 80% aller mobilen Endgeräte Smartphones. Hierbei haben die Systeme iOS und Android klar die Nase vorn.

Diese Geräte verwenden nicht den ursprünglich für Mobilgeräte vorgesehenen Medientyp `handheld`, sondern `screen`. Der Grund liegt darin, dass die meisten aktuellen Smartphones in der Lage sind, Websites so darzustellen, wie Sie es von Ihrem PC gewohnt sind – nur kleiner.

Um die Merkmale verschiedener Endgeräte (Bildschirmgröße, Auflösung, Ausrichtung usw.) besser unterscheiden zu können, hat sich das W3C die sogenannten *CSS3-Media Queries* (Medienabfragen) ausgedacht, die auch in unserer Lösung zum Einsatz kommen.

Mithilfe von Media Queries können Sie Stylesheets bedarfsgerecht (engl. »responsive«) laden. Je nachdem, welche Voraussetzungen ein bestimmtes Gerät oder Browser erfüllt, werden die passenden Regeln verwendet.

Letztendlich ist es aber nicht entscheidend, ob Sie ein iPhone 5, ein Galaxy Tab oder einen 28-Zoll-Monitor verwenden, sondern welche Merkmale diese Geräte haben. Mithilfe von Medienabfragen können Sie gezielt auf bestimmte Dinge testen und diese zur Unterscheidung nutzen.

Die meisten Smartphones geben als Bildschirmbreite beispielsweise einen Wert von 320 Pixeln im Porträtmodus und 480 Pixeln im Landschaftsmodus an. Durch entsprechende Medienabfragen können Sie Stylesheets speziell für diese Geräte und die verwendeten Browser schreiben.

max-width oder max-device-width?

Wenn Sie verschiedene Geräte anhand der Breite in Pixeln unterscheiden möchten, können Sie hierfür entweder `max-width` oder `max-device-width` verwenden. Aber wo genau liegt der Unterschied?

Während sich `max-device-width` auf die Breite des Bildschirms bezieht, steht `max-width` für die Breite des Ansichtsbereichs (bei Desktopbrowsern ist das das Browserfenster). Das ist zwar auf vielen Mobilgeräten das Gleiche, aber eben nicht überall.

Um sicherzugehen, empfehlen wir die Verwendung von `max-device-width`. Hier können Sie davon ausgehen, dass sich der Wert nicht plötzlich ändert. Wenn Sie dagegen mehrere Stylesheets für verschiedene Fenstergrößen erstellen wollen, ist `max-width` vermutlich die bessere Wahl.

Und was ist mit dem iPad?

Das iPad liegt irgendwo zwischen Smartphones und Laptops. Einerseits hat es einen relativ großen Bildschirm, andererseits wird es aber nicht über eine Tastatur, sondern mit Fingergesten gesteuert.

Um Stylesheets speziell für ein iPad auszuliefern, können Sie die folgenden Medienabfragen benutzen:

```
@media only screen
and (min-device-width: 768px)
and (max-device-width: 1024px)
and (orientation: portrait) {
    /* hier stehen die Regeln für iPads im Porträtmodus */
}

@media only screen
and (min-device-width: 768px)
and (max-device-width: 1024px)
and (orientation: landscape) {
    /* hier stehen die Regeln für iPads im Landschaftsmodus */
}
```

Diese Medienabfragen funktionieren für das iPad 1 und 2, deren Bildschirme 768 x 1.024 Pixel groß sind. Das iPad 3 unterstützt diese Abfragen ebenfalls, auch wenn intern mit einer höheren Auflösung gearbeitet wird.

Weitere Informationen zu `orientation` finden Sie in Rezept 10.2.

Stylesheets für Geräte mit höherer Auflösung

Immer mehr Smartphones arbeiten intern mit einer höheren Auflösung, als dies bei Desktopbrowsern der Fall ist. Um beispielsweise ein iPhone oder iPad 3 mit Retina-Display anzusprechen, können Sie schreiben:

```
@media only screen
and (-webkit-min-device-pixel-ratio : 2)
/* hier stehen die Regeln für iPhones und iPads mit Retina-Display */
}
```

Wir verwenden hier die Webkit-eigene Präfixschreibweise `-webkit-min-device-pixel-ratio`, um speziell für iPhones mit Retina-Display eigene Stile definieren zu können. Allerdings sind iPhone 4+ und iPad 3 nicht die einzigen Geräte, die intern mit einer höheren Auflösung arbeiten.

Um auch weitere Mobilgeräte mit hoher Auflösung zu unterstützen, können Sie beispielsweise die folgende `@media`-Regel verwenden:

```
@media only screen and (-webkit-min-device-pixel-ratio: 1.5),
only screen and (min--moz-device-pixel-ratio: 1.5),
only screen and (-o-device-pixel-ratio: 3/2),
only screen and (min-device-pixel-ratio: 1.5) {
/* hier stehen die speziellen CSS-Regeln für Geräte
mit hoher Bildschirmauflösung */
}
```

Am besten betrachten wir diesen Code der Reihe nach. Sie können bei Medienabfragen mehrere Optionen durch Kommata getrennt angeben. Alle Zeilen beginnen mit der Deklaration `only screen`. Hiermit gelten die Regeln für alle bildschirmbasierten Browser, die erweiterte Medienabfragen unterstützen.

Es folgt die Bedingung für iPhones (Webkit-Browser), bei denen die Anzahl der tatsächlichen Pixel mindestens um das Anderthalbfache höher liegt als der »offiziell« (z.B. für die Verwendung in `viewport-meta`-Tags) angegebene Wert.

Die zweite Zeile entspricht weitestgehend der ersten, nur wird hier die Präfixschreibweise für Firefox (und andere Gecko-basierte Browser) verwendet. Die Bedingung wird hier tatsächlich als `min--moz-device-pixel-ratio` (mit zwei Bindestrichen nach dem `min`) geschrieben.

In der dritten Zeile sehen wir die Angaben für Opera Mobile. Anstelle eines Faktors wird hier das Verhältnis von tatsächlichen zu »offiziellen« Pixeln angegeben, das mindestens vorhanden sein muss, damit die Bedingung erfüllt wird. Die vierte Zeile verwendet die Direktive ohne Präfix, wiederum mit einem Wert von 1.5 ist.



iPads der dritten Generation verfügen ebenfalls über ein Retina-Display mit einer Auflösung von 2.048×1.536 Pixeln. Daher ist es nötig, neben der Pixeldichte auch die »offizielle« Bildschirmbreite bzw. -höhe abzufragen, um iPhones und iPads unterscheiden zu können.

Media Queries und Responsive Webdesign

Auch wenn es in diesem Kapitel vorwiegend um Mobilgeräte geht, wollen wir das Thema kurz erwähnen. Unter Responsive Webdesign versteht man eine Technik, bei der sich die Darstellung einer Website oder -applikation an die Umgebung (Bildschirmgröße, -auflösung usw.) anpasst, in der sie gerade betrachtet oder benutzt wird.

Hierbei spielen Media Queries eine wesentliche Rolle. Je nachdem, für welche Geräte die Website angepasst werden soll, werden sogenannte Breakpoints definiert, an denen die Gestaltung wechselt.

Media Queries auch in älteren Versionen von Internet Explorer nutzen

CSS3-Medienabfragen werden erst ab dem Internet Explorer 9 unterstützt. Ältere Versionen verstehen zwar die Definition verschiedener Medientypen (screen, print usw.), aber keine komplexen Abfragen wie die in der Lösung gezeigten.

Mithilfe eines sogenannten »Polyfill« (siehe Rezept 13.8) können Sie diese Fähigkeiten aber auch in älteren Versionen von IE nachrüsten, indem Sie folgenden Code in den head-Teil Ihrer Website einbinden:

```
<!--[if lt IE 9]>
  <script src="http://css3-mediaqueries-js.googlecode.com/svn/trunk/css3-
mediaqueries.js"></script>
<![endif]-->
```

Hierbei müssen Sie darauf achten, dass der JavaScript-Code im Dokument vor den eigentlichen Medienabfragen steht, damit er auch rechtzeitig geladen wird.

Weitere Informationen finden Sie unter <http://code.google.com/p/css3-mediaqueries-js/>. Wir haben das Skript hier direkt von der Google-Website eingebunden. Sie können es aber auch auf Ihrer eigenen Website ablegen und von dort aus laden.

Siehe auch

Das *Media Queries*-Modul der CSS3-Spezifikation unter <http://www.w3.org/TR/css3-mediaqueries/>; den Artikel *Allen recht* unter <http://www.heise.de/ix/artikel/Allen-recht-1058764.html>; den Vortrag *Media Queries – Auf Geräte und Browser reagieren* von Michael Jendryschik unter <http://jendryschik.de/weblog/2010/07/05/wordcamp-session-css-media-queries/>; das Buch *Responsive Web Design* von Ethan Marcotte (A Book Apart) unter <http://www.abookapart.com/products/responsive-web-design>.

10.2 Längs- oder Querformat ermitteln

Problem

Sie möchten wissen, ob die Website im Landschafts- oder im Porträtmodus angezeigt wird.

Lösung

Nutzen Sie die folgenden Medienabfragen, um zu ermitteln, ob sich das Gerät im Hoch- oder Querformat befindet:

```
@media screen and (orientation: portrait) {  
    /* Anweisungen für den Porträtmodus stehen hier */  
}  
@media screen and (orientation: landscape) {  
    /* Anweisungen für den Landschaftsmodus stehen hier */  
}
```

Möglicherweise wollen Sie diese Abfrage mit einem Test auf die verfügbare Bildschirmbreite kombinieren, wie in Rezept 10.3 beschrieben.

Diskussion

Moderne Smartphones, hier speziell iPhone und iPad, können dank eines eingebauten Gyroskops erkennen, ob sie im Längs- oder Querformat benutzt werden. Gelegentlich wollen Sie für jede Darstellungsart einen eigenen Stil definieren. Mithilfe der Medienabfrage `orientation` können Sie das Format ermitteln.

Der Wechsel von `portrait` nach `landscape` findet statt, sobald der Ansichtsbereich breiter ist als hoch. (Wir schreiben hier absichtlich `Ansichtsbereich`, weil diese Methode auch in vielen Desktopbrowsern funktioniert.)



Von einer Benutzung von `orientation` im `media`-Attribut eines `link`-Tags wird abgeraten, da dies zusätzliche HTTP-Anfragen zur Folge haben kann. Es ist besser, die Anweisungen per `@media`-Regel in einem gemeinsamen Stylesheet zu definieren.

Die Medienabfrage `orientation` funktioniert unter anderem in iOS 3.2+ sowie Android 2.0+ und steht damit für eine große Zahl an Mobilgeräten zur Verfügung.

Allerdings sollten Sie sich bei der Verwendung von `orientation` darüber klar werden, ob diese Abfrage wirklich nötig ist. Die wichtige Information ist eigentlich die Breite (und eventuell die Höhe) des Ansichtsbereichs, die Sie mit der Medienabfrage `width` bzw. `device-width` ermitteln können.

Sie sollten Ihr mobiles Layout jedenfalls nicht ausschließlich von der Orientierung des Geräts abhängig machen. Auf der Quirksmode-Website wird daher empfohlen, den Test auf die Orientierung prinzipiell mit weiteren Medienabfragen auf die Breite des Geräts (bzw. des Ansichtsbereichs) zu kombinieren.

Kreativ eingesetzt, kann eine Änderung der Orientierung aber auch ihren Reiz haben. Ein gutes Beispiel ist die Chamäleon-Demo auf der Website von *cssninja* (Link im Abschnitt »Siehe auch«). Hier wird die Medienabfrage nach der Orientierung geschickt mit einem CSS3-Übergang (transition) kombiniert.

Siehe auch

Das Rezept 10.1 zur Verwendung von Media Queries; das Rezept 10.3 zur Steuerung der Größe und Skalierung des Ansichtsbereichs; den Artikel *iPad Orientation CSS (Revised)* zur Verwendung der Medienabfrage `orientation` unter <http://catharsis.tumblr.com/post/501657271/ipad-orientation-css-revised>; eine kritische Betrachtung zur Verwendung von `orientation` auf der Quirksmode-Website unter http://www.quirksmode.org/blog/archives/2010/04/the_orientation.html; ein Beispiel für die kreative Verwendung von `orientation` unter http://www.thecssninja.com/demo/css_chameleon/.

10.3 Die Größe und Skalierung des Ansichtsbereichs steuern

Problem

Obwohl das richtige Stylesheet für die mobile Version Ihrer Website geladen wird, erscheinen die Inhalte viel zu klein. Sie möchten die Größe des Ansichtsbereichs und dessen Skalierung selbst bestimmen.

Lösung

Verwenden Sie im `head`-Teil der Seite (bzw. der Seitenschablone) ein `meta`-Element, dessen `name`-Attribut den Wert `viewport` trägt. Über das Attribut `content` können Sie dem Browser anhand verschiedener sogenannter Direktiven die Breite, Höhe und Skalierung des Dokuments mitteilen. Ein Beispiel:

```
<meta name="viewport" content="width=device-width, initial-scale=2.3,
user-scalable=no" />
```

Eine Direktive besteht aus einem Schlüssel und einem Wert, die durch ein Gleichheitszeichen miteinander verbunden werden. Sie steuern einige grundsätzliche Darstellungsparameter, die über CSS selbst nicht zugänglich sind.

Hier geben wir an, dass die Breite des Inhalts an die Breite des Ansichtsbereichs angepasst werden soll. Gleichzeitig wird die anfängliche Skalierung auf einen Wert von 2.3 gesetzt. (Anstelle des Dezimalkommas wird hier ein Punkt verwendet.) Die Website

erscheint also etwas mehr als doppelt so groß. Abschließend wird noch festgelegt, dass der Benutzer die Darstellung nicht selbst zoomen kann.

Sie können auch mehrere Direktiven miteinander kombinieren, indem Sie sie durch Kommata voneinander getrennt angeben. Im Gegensatz zu @media-Regeln oder dem media-Attribut in einem link-Tag werden der Name einer Anweisung und ihr Wert jeweils durch ein Gleichheitszeichen getrennt.

Diskussion

Um Smartphones von Desktopbrowsern unterscheiden zu können, nutzen Sie Medienabfragen (Media Queries), um die Breite (oder Höhe) des Ansichtsbereichs zu ermitteln. Die meisten Smartphones verwenden hierbei eine »offizielle« Breite von 320 bzw. 480 Pixeln.

Intern arbeiten Smartphones allerdings oft mit einer höheren Auflösung als Desktopbrowser. Während hier die Auflösung in den meisten Fällen bei 96dpi (»dots per inch«) liegt, haben iPhones mit Retina-Display bereits eine Auflösung von 326dpi (das Samsung Galaxy Tab 2 soll sogar auf 350dpi kommen).

Das heißt, auch wenn ein Handy die darstellbare Breite mit 320 oder 480 Pixeln angibt, liegt diese oft deutlich darüber. Hinzu kommt, dass mobile Browser eine Website von sich aus skalieren (zoomen), damit sie in voller Breite dargestellt werden kann, ohne dass horizontal gescrollt werden muss. Das bedeutet, die vom Gerät verwendeten Pixel entsprechen nicht unbedingt den Pixeln in Ihrem Stylesheet.

Angenommen, Sie geben einem Element in Ihrem Stylesheet eine Breite von 100%, damit es sich über den gesamten Bildschirm erstreckt:

```
h1, p {  
    width: 100%;  
}
```

Das hat den Vorteil, dass die Breite der Elemente sich automatisch an die vom Gerät angegebene Breite anpasst. Das Ergebnis sehen Sie in Abbildung 10-1.

Offiziell kann das hier gezeigte iPhone 4 Dokumente mit einer Breite von 320 Pixeln ausgeben. Durch die höhere Auflösung sind es tatsächlich aber 980! Die Darstellung erscheint deutlich kleiner als erwartet.

Dieses Problem können wir durch die Verwendung des viewport-meta-Tags in den Griff bekommen. Hiermit können Sie festlegen, dass die Breite des Layouts (in »CSS-Pixeln«, siehe auch Rezept 2.1) der vom Gerät angegebenen Breite des Ansichtsbereichs entsprechen soll:

```
<meta name="viewport" content="width=device-width" />
```

Nach der Anpassung sieht die Darstellung aus, wie in Abbildung 10-2 gezeigt.



Abbildung 10-1: Überschrift und Absatz nehmen die gesamte Breite des Bildschirms ein, wirken aber kleiner als erwartet

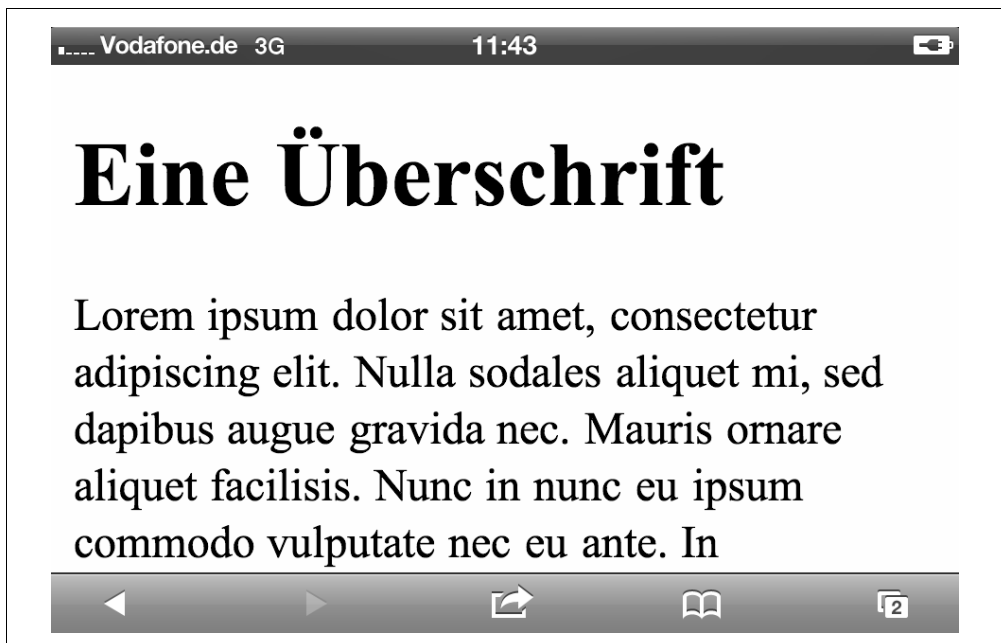


Abbildung 10-2: Durch die Verwendung des viewport-meta-Tags wurde die im Stylesheet angegebene Breite an die Breite des Geräts angepasst

Sie können die Breite des Ansichtsbereichs auch in Pixeln festlegen:

```
<meta name="viewport" content="width=320" />
```

Neben der Breite können Sie die Höhe angeben. Das meta-Tag sieht recht ähnlich aus:

```
<meta name="viewport" content="height=480" />
```

Auch hier gibt es, analog zur Breite, das Schlüsselwort `device-height`, mit dem die Höhe des `body`-Elements an die des Geräts angepasst werden kann.

Skalierung

Ohne weitere Angaben skalieren die meisten Smartphones Ihre Website so, dass sie angezeigt werden kann, ohne dass horizontal (von links nach rechts) gescrollt werden muss.

Allerdings sind viele Elemente dadurch nur schwer oder gar nicht benutzbar, weil sie schlicht zu klein dargestellt werden. Um die Benutzung eines Navigationsmenüs oder das Lesen einzelner Teile der Seite zu erleichtern, kann (bzw. muss) der Benutzer die Website zum Beispiel durch »Pinch-Zoom« nach Bedarf vergrößern und verkleinern.

In manchen Fällen, etwa bei Webapplikationen, ist das aber gar nicht erwünscht. In diesem Fall können Sie den `viewport-meta`-Tag verwenden, um die Skalierung durch den Benutzer zu deaktivieren:

```
<meta name="viewport" content="user-scalable=no" />
```

Um das Zoomen durch den Benutzer ausdrücklich zu erlauben, können Sie der Direktive `user-scalable` auch den Wert `yes` geben.

Sie können ebenfalls festlegen, welcher minimale bzw. maximale Skalierungsfaktor verwendet werden soll. Hierfür verwenden Sie die Direktiven `minimum-scale` bzw. `maximum-scale`.

```
<meta name="viewport" content="minimum-scale=0.5" />  
<meta name="viewport" content="maximum-scale=2.5" />
```

Die obere Zeile legt fest, dass der Ansichtsbereich höchstens auf die Hälfte verkleinert werden kann; die untere besagt, dass maximal um das Zweieinhalbfache vergrößert werden kann.

Zudem ist es möglich, anzugeben, wie groß der anfängliche Skalierungsfaktor sein soll. Um die Seite bereits beim Laden um das Doppelte zu vergrößern, können Sie schreiben:

```
<meta name="viewport" content="initial-scale=2" />
```

Der Wert für alle drei Direktiven ist eine positive Zahl zwischen 0,25 und 10. (Beachten Sie, dass anstelle des Kommas ein Punkt eingegeben werden muss!)

Mehrere Werte auf einmal angeben

Wie oben zu sehen, können Sie auch mehrere viewport-meta-Tags angeben. Ersparen Sie sich dabei etwas Schreibarbeit (und Bandbreite), indem Sie mehrere Direktiven und ihre Werte durch Kommata getrennt in einem Tag unterbringen:

```
<meta name="viewport" content="width=device-width, height=device-height,  
user-scalable=yes, initial-scale=2, minimum-scale=0.5, maximum-scale=2.5" />
```

Hier haben wir alle möglichen Direktiven in einem gemeinsamen meta-Element zusammengefasst.



Das W3C arbeitet gegenwärtig daran, die Angaben des viewport-meta-Tags mithilfe einer @viewport-Regel auch direkt im Stylesheet machen zu können. Hiermit soll versucht werden, die Funktionalität des meta-Tags zu vereinheitlichen und zu standardisieren. Allerdings liegt die Spezifikation beim Schreiben dieses Buchs nur als Arbeitsentwurf (Working Draft) vor.

Immerhin unterstützt Opera Mobile 11 @viewport Regel bereits in der Präfixschreibweise als @-o-viewport.

In der folgenden Tabelle sehen Sie die Direktiven und die möglichen Werte

Tabelle 10-1: Werte für die Direktiven des viewport-meta-Tags

Direktive	Werte	Schlüsselwörter
width	Pixelwert zwischen 200 und 10.000	device-width
height	Pixelwert zwischen 223 und 10.000	device-height
initial-scale	numerischer Wert zwischen 0 und 10, Nachkommastellen erlaubt	
minimum-scale	numerischer Wert zwischen 0 und 10, Nachkommastellen erlaubt, Standardwert: 0.25	
maximum-scale	numerischer Wert zwischen 0 und 10, Nachkommastellen erlaubt, Standardwert: 1.6 (iPhone)	
user-scalable		yes bzw. no

Siehe auch

Den Artikel *A Pixel is not a Pixel* unter http://www.quirksmode.org/blog/archives/2010/04/a_pixel_is_not.html; den Artikel *A Tale of two Viewports* unter <http://www.quirksmode.org/mobile/viewports2.html>; den Artikel *An introduction to meta viewport and @viewport* unter <http://dev.opera.com/articles/view/an-introduction-to-meta-viewport-and-viewport/>; den Webkrauts-Artikel *Ein Blick durch den Viewport* unter <http://www.webkrauts.de/2011/12/21/ein-blick-durch-den-viewport/>.

10.4 Stile für Mobilgeräte zuerst laden

Problem

Sie möchten vermeiden, dass für Mobilgeräte überflüssige CSS-Anweisungen mitgeladen werden.

Lösung

Handeln Sie nach dem Prinzip »Mobile First«. Laden Sie zunächst ein Stylesheet, das sämtliche Stildefinitionen für die Darstellung in Mobilgeräten enthält.

Mithilfe von Medienabfragen können Sie dann im zweiten Schritt die Regeln für größere Bildschirme und andere Medien nachladen. Durch die Beschränkungen in den Media Queries werden die zusätzlichen Stylesheets nur geladen, wenn sie tatsächlich gebraucht werden:

```
<link rel='stylesheet' href='css/mobile.css' media='all' />
<link rel='stylesheet' href='css/desktop.css' media='screen and (min-width: 481px)' />
```

Anstatt erst das Stylesheet für Desktopbrowser zu laden, werden hier zunächst nur die Stile geladen, die für die Darstellung in Mobilgeräten notwendig sind. Erst im zweiten link-Tag werden bei Bedarf zusätzliche Stile für die Darstellung in Desktopbrowsern eingebunden.

Diskussion

Häufig werden Websites zuerst für die Darstellung in Desktopbrowsern erstellt. Erst danach wird die Gestaltung für die Anzeige in Mobilgeräten angepasst. Dadurch müssen Smartphones und Tablets zunächst den gesamten Satz an Anweisungen laden, nur um diese danach wieder zu überschreiben.

Hierfür sind oft diverse zusätzliche HTTP-Requests nötig, die für die mobile Darstellung aber keinen Nutzen haben. In unserer Lösung wird zunächst das Stylesheet für Mobilgeräte geladen, damit diese sofort mit der Darstellung beginnen können.

Desktopbrowser, die in der Regel über eine bessere Internetverbindung verfügen, laden bei Bedarf das zweite Stylesheet nach.

Siehe auch

Das Rezept 10.1 zur Unterscheidung von mobilen und Desktopbrowsern mithilfe von Medienabfragen; die Artikel *Mobile First* und *Mobile First Helps with Big Issues* von Luke Wroblewski unter <http://www.lukew.com/ff/entry.asp?933> bzw. <http://www.lukew.com/ff/entry.asp?1117>; das Rezept 10.5 mit Tipps zum Verringern der Ladezeit von Stylesheets.

10.5 Das Laden von Stylesheets beschleunigen

Problem

Sie möchten die Ladezeiten der Stylesheets für Ihre (mobile) Website möglichst klein halten.

Lösung

Fassen Sie einzelne Stylesheets gegebenenfalls zu einer Datei zusammen und »verschlanen« (engl. minify) Sie so Ihren Code. Oftmals findet man Anweisungen wie diese:

```
@import "css/reset.css";
@import "css/schriften.css";
@import "css/menue.css";
@import "css/website.css";
```

Auch wenn das schön übersichtlich ist, müssen für jede Datei mehrere Abfragen an DNS- und Webserver gestellt werden. Abhängig von der Qualität Ihrer Internetverbindung dauert das seine Zeit.

Um die Zahl der Abfragen möglichst klein zu halten, ist es sinnvoll, den Code einzelner Stylesheets in einer Datei zusammenzufassen und diese direkt im Kopf der Seite mit einem link-Tag einzubinden.

```
<link rel='stylesheet' media='only screen and (max-device-
width=480)' href='css/alles_auf_einmal.css' />
```

Anstelle mehrerer per @import eingebundener CSS-Dateien laden wir hier ein einziges Stylesheet, das alle für die Darstellung nötigen Anweisungen enthält.

Nach Möglichkeit können Sie die Datei außerdem noch verschlanken. Mithilfe eines speziellen Werkzeugs werden, wie in der Diskussion beschrieben, unnötige Leerzeichen und Zeilenumbrüche entfernt und redundante Regeln zusammengefasst. Allein hierdurch lässt sich die Größe eines Stylesheets oft schon auf die Hälfte reduzieren.

Diskussion

Bei Mobilgeräten steht selten die gleiche Bandbreite zur Verfügung wie bei Desktoprechnern, die per DSL o.Ä. mit dem Internet verbunden sind. Außerdem lassen sich die Mobilfunkanbieter die Datenübertragung gern fürstlich bezahlen.

Daher sollten Sie auch bei der Erstellung Ihrer Stylesheets darauf achten, die übertragene Datenmenge und die Zahl der Serverabfragen möglichst gering zu halten.

Letzteres erreichen Sie durch das Zusammenfassen mehrerer Dateien. Was bei der Entwicklung Ihrer Website für Übersicht und sauberes Arbeiten sorgt, ist im Alltagsbetrieb möglicherweise ein Bremsklotz.



Zusätzlich können Sie die Dateien mit Programmen wie *gzip* auch auf Serverseite komprimieren. Hierbei werden die Daten vor dem Versenden an den Browser durch den Webserver kodiert und nach der Übertragung vom Webbrowser wieder entpackt.

Auf diese Weise lässt sich die Übertragungsgeschwindigkeit teilweise deutlich steigern. Dieses Verfahren wird von den meisten Browsern verstanden, erfordert aber spezielle Einstellungen am Webserver.

Die übertragene Datenmenge können Sie auf verschiedene Weise reduzieren. Wir haben in der Lösung bereits das Verschlinken erwähnt. Zahlreiche Websites bieten die Möglichkeit, Ihren Code nach bestimmten Kriterien (möglichst klein, trotzdem noch für Menschen lesbar usw.) kompakter zu machen. Links hierzu finden Sie im Abschnitt »Siehe auch«.

Das Verschlinken kann aber automatisch auf Serverseite vorgenommen werden. Zahlreiche Plug-ins und PHP-Skripten helfen Ihnen dabei. Eine Google-Suche nach »minify CSS« bringt Sie sicher auf den richtigen Weg.

Siehe auch

Die Google-Website mit Tipps und Anregungen zum Verringern der Ladezeit von Websites unter http://code.google.com/intl/de-DE/speed/page-speed/docs/rules_intro.html; ein webbasiertes Werkzeug zum »Minifizieren« von CSS-Dateien unter <http://www.minifycss.com/css-compressor/>; eine Liste mit Werkzeugen (webbasiert und für den eigenen Rechner) zum Verkleinern von CSS-Dateien unter <http://www.minifycss.com/css-compressor/>; den CSS-JS-Booster, der verschiedene Methoden zum Verkleinern von CSS- (und JavaScript-)Dateien kombiniert, unter <https://github.com/Schepp/CSS-JS-Booster#readme>.

10.6 Mobile Seiten mit CSS3 schneller machen

Problem

Sie möchten die Ladezeit Ihrer mobilen Website verringern, indem auf die Verwendung zusätzlicher Dateien (insbesondere Grafiken) verzichtet wird. Stattdessen sollen CSS3-Eigenschaften und -werte verwendet werden.

Lösung

Glücklicherweise unterstützt der größte Teil der mobilen Browser heute schon viele der CSS3-Eigenschaften. Hierdurch kann in vielen Fällen auf Layoutgrafiken (Farbverläufe, abgerundete Ecken, Schattenwürfe usw.) verzichtet werden.

Wir zeigen Ihnen das hier am Beispiel eines Farbverlaufs, der für Mobilgeräte mit einer maximalen Bildschirmbreite von 480 Pixeln definiert werden soll. Um das Beispiel einfach zu halten, verwenden wir zunächst nur die offizielle Schreibweise:

```
@media screen and (max-width: 480px) {  
  body, h1, h2 {  
    background-image: linear-gradient(top, #79E5FF, #275B82);  
  }  
}
```

Anstatt ein Hintergrundbild zu definieren, versehen wir hier mit der Funktion `linear-gradient()` sowohl die Seite selbst wie auch Überschriften erster und zweiter Ordnung mit einem blauen Farbverlauf.

Da die hier gezeigte W3C-Form beim Schreiben dieses Buchs nur von Android 4+ unterstützt wurde, sollten Sie in einem Produktions-Stylesheet die gängigen Präfixvarianten mit angeben, wie hier gezeigt:

```
@media screen and (max-width: 480px) {  
  body, h1, h2 {  
    background: #275B82;  
    background-image: -webkit-gradient(linear, left top, left bottom, from(#79E5FF),  
    to(#275B82));  
    background-image: -webkit-linear-gradient(top, #79E5FF, #275B82);  
    background-image: -moz-linear-gradient(top, #79E5FF, #275B82);  
    background-image: -ms-linear-gradient(top, #79E5FF, #275B82);  
    background-image: -o-linear-gradient(top, #79E5FF, #275B82);  
    background-image: linear-gradient(top, #79E5FF, #275B82);  
  }  
}
```

Zu Beginn definieren wir für alle Fälle eine schlichte Hintergrundfarbe, die von einer der folgenden Deklarationen wieder überschrieben wird, sofern der Browser sie versteht. Die zweite Deklaration wird benötigt, damit auch ältere Versionen von iOS Safari (3.2 bis 4.3) die Anweisungen verstehen.

Diese Codewiederholungen sehen nicht besonders schön aus und bedeuten eine Menge zusätzliche Tipparbeit für Webentwickler. Im Moment stellen sie aber die einzige Möglichkeit dar, CSS3-Eigenschaften zu nutzen.

Diskussion

Die CSS3-Unterstützung ist in vielen Smartphone-Browsern bereits jetzt erstaunlich gut. Obwohl viele Eigenschaften momentan noch in der Präfixschreibweise angegeben werden müssen, können Sie durch die browsereigene Erzeugung von grafischen Elementen eine Menge Bandbreite sparen.

Die in der Lösung gezeigte Regel hat eine Größe von etwas mehr als 300 Byte. Eine PNG-Datei für den gleichen Zweck wäre um ein Vielfaches größer.

Neben den Eigenschaften für Farbverläufe unterstützen viele Mobilbrowser auch die Erzeugung von abgerundeten Ecken (`border-radius`), Transformationen (`transform`), Animationen (`transition`), Schattenwürfen (`box-shadow`, `text-shadow`), RGB-Farben mit Alphakanal (`rgba()`) und vieles mehr.

Durch die konsequente Nutzung dieser Techniken können Sie die Ladezeit, nicht nur für Ihre mobile Website, deutlich verringern.

Siehe auch

Das Rezept 2.12 zur Definition von Hintergründen; das Rezept 12.4 zur Definition von Farbverläufen mit CSS3; das Rezept 12.6 zur Erzeugung von abgerundeten Ecken ohne zusätzliche Grafiken; die Rezepte 12.9 und 12.10 zur Definition von Transformationen und Animationen; die Website *haz.io* zum Überprüfen, welche CSS3-Eigenschaften der gerade verwendete Browser unterstützt, unter <http://haz.io/>; die Website *caniuse.com* zum Ermitteln der CSS3-Eigenschaften, die in verschiedenen Mobilgeräten (mit oder ohne Präfix) genutzt werden können, unter <http://caniuse.com>.

10.7 Nicht benötigte Elemente ausblenden

Problem

Sie möchten in der mobilen Version Ihrer Website nur die wirklich benötigten Elemente anzeigen.

Lösung

Blenden Sie Elemente, die nicht gebraucht werden, per `display: none` aus:

```
#seitenleiste, object, div.werbung, #soziale-netzwerke {  
  display: none;  
}
```

Dieser Code sorgt dafür, dass die `#seitenleiste`, die mit dem `object`-Tag eingebundenen Multimedia-Elemente, `div`-Elemente mit der Klasse `werbung` und das Element mit der ID `soziale-netzwerke` nicht dargestellt werden.



In Ihrem Stylesheet heißen die Elemente möglicherweise anders. Sollten die gewünschten Dinge nicht über Klassen- oder ID-Selektoren zugänglich sein, können Sie diese vermutlich mit anderen Selektoren oder deren Kombination auswählen.

Diskussion

Manche Elemente, die auf einer Desktop-Website gut aussehen, sind in der mobilen Version oft nicht sinnvoll. Verschanken Sie Ihre Seiten und zeigen Sie nur die Elemente an,

die wirklich nötig sind. iPhones sind beispielsweise gar nicht in der Lage, Flash-Animationen anzuzeigen, stattdessen wird konsequent auf HTML5 gesetzt.

Werbepartner wirken auf kleinen Bildschirmen schnell störend. Wenn Sie Werbung auf Ihrer mobilen Website schalten müssen, sollte diese an das Medium angepasst werden. Viele Werbepartner bieten inzwischen speziell auf Mobilgeräte zugeschnittene Lösungen.



Die Verwendung von `display: none` verhindert nicht unbedingt, dass die verborgenen Elemente oder mögliche Hintergrundbilder heruntergeladen werden.

Wir haben in der Lösung außerdem das Element ausgeblendet, das in der imaginären Version unserer Website die Links auf soziale Netzwerke enthält (`#soziale-netzwerke`). Für die meisten sozialen Netzwerke gibt es inzwischen eigene Apps, die für Mobilgeräte wesentlich besser geeignet sind.

Siehe auch

Das Rezept 10.8 zur Linearisierung eines Layouts für Mobilgeräte; das Rezept 4.4 zur richtigen Skalierung von Bildern; den englischen Wikipedia-Artikel zum Thema *Mobile Advertising* unter http://en.wikipedia.org/wiki/Mobile_advertising; die Website *Web Performance Best Practices* unter http://code.google.com/intl/de-DE/speed/page-speed/docs/rules_intro.html.

10.8 Seitenlayouts linearisieren

Problem

Sie verwenden ein mehrspaltiges Layout. Für viele Mobilgeräte ist das aber zu breit. Die Inhalte (Kopfteil, Hauptteil, Seitenleiste, Fußteil usw.) sollen deshalb bei Bedarf untereinander dargestellt werden.

Lösung

Setzen Sie die wichtigsten Eigenschaften auf ihren Standardwert zurück: Geben Sie der Eigenschaft `width` den Wert `auto`, entfernen Sie unnötige Innen- und Außenabstände und geben Sie für die Elemente bei Bedarf `float: none` und `clear: both` an. Passen Sie gegebenenfalls die Positionierung an und definieren Sie Containerelemente als Blockelemente, z.B.:

```
@media only screen and (max-device-width: 480px) {
  div, header, nav, footer {
    position: static;
    display: block;

    margin: 0;
```

```

width: auto;
float: none;
clear: both;
}
}

```

Hier definieren wir für alle `div`-Elemente sowie die strukturellen HTML5-Elemente `header`, `nav` und `footer` eine allgemeine Reset-Regel. Sie sorgt zunächst dafür, dass Positionierung und Darstellungsrolle wieder ihren Standardwert erhalten. Eventuell angegebene Außenabstände werden entfernt.

Wurde das ursprüngliche Layout mit Floats definiert, wird `float` auf den Wert `none` gesetzt. Für den Fall, dass z.B. ersetzte Elemente (wie Bilder) trotzdem als Float erhalten bleiben, geben wir in der letzten Deklaration per `clear: both` an, dass diese Elemente nicht umfließen werden sollen.

Diskussion

Unter *Linearisierung* versteht man eine Anpassung an die geringe Displaybreite von Mobilgeräten. Statt nebeneinander werden die wichtigsten Elemente wie Navigation, Hauptinhalt, Seitenleiste, Fußteil usw. in der Reihenfolge ihrer Definition im HTML-Quellcode dargestellt.

Wie in Kapitel 11, *Seitenlayouts*, beschrieben wird, gibt es mehrere Möglichkeiten, die Bestandteile einer Seite (Kopfteil, Navigation, Hauptteil, Seitenleiste, Fußteil usw.) im Layout anzuordnen. In den meisten Fällen kommen hierfür die Eigenschaften `position`, `margin`, `width` und `float` zum Einsatz. Daher müssen wir für eine lineare Darstellung zumindest diese Eigenschaften auf ihren Standardwert zurücksetzen.

Sofern die Bedingungen der `@media`-Regel erfüllt werden, erscheinen die wichtigsten Seitenelemente nun in der Reihenfolge ihrer Definition im HTML-Dokument untereinander.

Im Web wird gelegentlich empfohlen, zusätzlich die Regel `-webkit-text-size-adjust: none;` bzw. `-ms-text-size-adjust: none` anzugeben. Damit soll verhindert werden, dass die Textgröße in iOS-Geräten und IEMobile, z.B. beim Wechsel zwischen Porträt- und Landschaftsmodus, angepasst wird. Der schwedische Frontend-Experte Roger Johansson empfiehlt stattdessen, den Wert `100%` zu verwenden, weil Benutzer bei Verwendung von `auto` die Seite möglicherweise nicht mehr zoomen können.

```

-webkit-text-size-adjust: 100%;
-ms-text-size-adjust: 100%;

```

Um Darstellungsprobleme mit »überlaufenden« Elementen zu vermeiden, verwenden wir in der Lösung anstelle von `width: 100%` die Deklaration `width: auto`. Details hierzu finden Sie in Rezept 1.16 zum Boxmodell.

Die hier gezeigte Lösung ist nicht allgemeingültig. Sie soll als Ausgangspunkt für eine Linearisierung Ihrer eigenen Stile dienen. Vermutlich müssen Sie die hier gezeigten Regeln für Ihre Website anpassen oder erweitern.

Noch besser ist es natürlich, wenn Sie bei der Erstellung Ihrer Website dem »Mobile First«-Grundsatz folgen und die Stile zuerst für die mobile Version Ihrer Website erstellen und laden, wie in Rezept 10.4 beschrieben wird.

Siehe auch

Das Rezept 10.4 zum Thema »Mobile First«; den Artikel *The difference between width: auto and width: 100%* unter http://www.456bereastreet.com/archive/201112/the_difference_between_widthauto_and_width100/; das Rezept 10.7 zum Ausblenden nicht benötigter Elemente in mobilen Stylesheets; einen Artikel von Roger Johansson über die Steuerung der Textgröße in iOS Safari unter http://www.456bereastreet.com/archive/201012/controlling_text_size_in_safari_for_ios_without_disabling_user_zoom/.

10.9 Tabellen linearisieren

Problem

Sie möchten die Darstellung von Tabellen für die Anzeige in Mobilgeräten optimieren.

Lösung

Linearisieren Sie die Tabelle durch Verwendung relativer Maßeinheiten oder Prozentangaben. Stellen Sie die Tabellenzellen ab einer bestimmten Bildschirmbreite untereinander dar, wie in den Abbildungen 10-3 und 10-4 gezeigt.

Diese Lösung basiert auf folgendem HTML-Code:

```
<table>
<caption>KlangWelten-Festival 2011 - Tourneedaten</caption>
<tr>
  <th>Datum</th>
  <th>PLZ</th>
  <th>Ort</th>
  <th>Veranstaltungsort</th>
  <th>Zeit</th>
  <th>Kartentelefon</th>
  <th>Bemerkungen</th>
</tr>

<tr>
  <td>03.11.2011</td>
  <td>76131</td>
  <td>Karlsruhe</td>
  <td><a href="http://www.tollhaus.de">Tollhaus</a></td>
```

```

</td></td>
</tr>

```

```

<!-- weitere Datensätze hier -->
</table>

```

Und hier folgt das Stylesheet mit zunächst den Anweisungen, die für die Desktopbrowser und Mobilgeräte gleichermaßen gelten sollen. Diese werden mithilfe einer Medienabfrage teilweise wieder überschrieben:

```

table {
    width:          100%;
    border-collapse: collapse;
}

```

Datum	PLZ	Ort	Veranstaltungsort	Zeit	Kartentelefon	Bemerkungen
03.11.2011	76131	Karlsruhe	Tollhaus	20:00	0721-964050	
04.11.2011	94032	Passau	Redoute	20:00		leider ausverkauft
06.11.2011	76829	Landau	Altes Kaufhaus	19:00	0631-37016618	leider ausverkauft
09.11.2011	A-5400	Hallein	Ziegelstadl	20:00		
10.11.2011	78224	Singen	Gems	20:00	07731-67578	
12.11.2011	93047	Regensburg	St. Oswald	20:00	09402-5256	
13.11.2011	58313	Herdecke	Werner Richard Saal	19:00		
14.11.2011	56072	Koblenz	Cafe Hahn	20:00	0261-42302	
15.11.2011	53113	Bonn	Kunst- und Ausstellungshalle	20:00	0228-9171-431	leider ausverkauft
16.11.2011	66538	Neunkirchen	Bürgerhaus	20:00		
17.11.2011	55116	Mainz	Frankfurter Hof	20:00	frankfurter-hof@stadt.mainz.de	
18.11.2011	88471	Laupheim	Schloss Großlaupheim	20:00		
19.11.2011	77654	Offenburg	Reithalle	20:00	0781-822000	leider ausverkauft
20.11.2011	75172	Pforzheim	Kulturhaus Osterfeld	20:00		
21.11.2011	55543	Bad Kreuznach	Loge	20:00	0671-83600	leider ausverkauft
22.11.2011	01833	Stolpen	Stadtkirche	20:00	035973-27313	
23.11.2011	86199	Augsburg	ParkTheater im Kurhaus Göggingen	19:30	0821-9062222	
24.11.2011	87435	Kempten	Kornhaus	20:00	0831-206 222	leider ausverkauft
26.11.2011	73734	Esslingen	Dieselstraße	20:00	0711 / 387120	
27.11.2011	70794	Filderstadt	Waldorfschule Gutenhalde	18:00	0711/ 77 40 18	leider ausverkauft
29.11.2011	07743	Jena	Volkshaus	20:00	03641-806406	
30.11.2011	63897	Miltenberg	Bürgerhaus Eisenfeld	20:00		
01.12.2011	68167	Mannheim	Alte Feuerwache	20:00	0621-101011	
02.12.2011		Hannover	Markuskirche	20:00		
03.12.2011	41564	Kaarst	Albert Einstein Forum	20:00	02131-987 223	leider ausverkauft

Abbildung 10-3: Eine schlichte Datentabelle in einem Desktopbrowser

Die Tabelle soll sich über die gesamte Breite des Ansichtsbereichs erstrecken, die Rahmen um die Zellen werden zusammengefasst.

```
caption {
  margin-bottom: 0.5em;
  font-weight: bold;
  font-size: 1em;
}
```

Die Tabellenbeschriftung soll standardgemäß über der Tabelle angezeigt werden. Um zwischen der Beschriftung und der Tabelle etwas Platz zu schaffen, definieren wir einen unteren Außenabstand. Um die Beschriftung außerdem etwas hervorzuheben, soll die Schrift fett dargestellt werden.

```
table, tr, th, td {
  padding: 0.2em;
  border: 1px solid black;

  background-color: white;
}
```

Es folgen die Deklarationen, die für alle Bestandteile der Tabelle gleichermaßen gelten sollen. Wir haben hier das eigentlich unsichtbare Element `tr` mit in die Liste aufgenommen, damit wir die Stile im Stylesheet für Mobilgeräte nicht separat definieren müssen. Auf die Darstellung in Desktopbrowsern hat das keinen Einfluss.

```
tr:nth-child(odd) > th,
tr:nth-child(odd) > td {
  background-color: #ddd;
}
```

Hier verwenden wir die CSS3-Pseudoklasse `:nth-child()` zusammen mit einem Kindselektor, um Kopf- und Datenzellen in ungeraden Zeilen mit einem etwas dunkleren Hintergrund zu versehen.

Jetzt wird es spannend. Es folgen die Regeln für die Darstellung in Mobilgeräten und kleinen Browserfenstern:

```
@media screen and (max-device-width: 480px),
  screen and (max-width: 480px) {

  table { border: none; }

  tr, td {
    display: block;
    margin: 0.2em;
    border-radius: 0.3em;
  }

  tr:first-child { display: none; }
```

Nachdem wir den Rahmen um die Tabelle entfernt haben, kann die Arbeit beginnen. Der wichtigste Schritt ist die Änderung der Darstellungsrolle von Tabellenzeilen und Datenzellen. Durch die Darstellung als Blockelemente werden die Zellen untereinander angeordnet.

Die Spaltenüberschriften brauchen wir hier nicht. Daher werden sie über die Pseudoklasse `:first-child()` schlicht von der Darstellung ausgenommen. Stattdessen können Sie die einzelnen Zellen im Rahmen des Progressive Enhancement mit einer eigenen Beschriftung versehen.

```
tr {
    width:          99%;
    margin:         0;
    margin-bottom:  0.4em;
    background-color: #efefef;
}

td:empty { display: none; }

/* Ende der Medienabfrage */
```

Eigentlich sind die Tabellenzeilen (`tr`) nur strukturelle Elemente, die nicht selbst dargestellt werden. Durch die obige Definition als Blockelemente können wir sie für die Mobilversion jedoch aktiv in die Darstellung einbeziehen. Sie dienen hier als Behälter für jeweils einen Datensatz.

Da wir auch die Darstellungsrolle der Tabellenzellen geändert haben, funktioniert die Eigenschaft `empty-cells` hier nicht. Wir behelfen uns stattdessen mit der Pseudoklasse `:empty`, um Tabellenzellen ohne Inhalt auszublenden. Die linearisierte Tabelle sehen Sie in Abbildung 10-4.



Abbildung 10-4: Die gleiche Tabelle wird auf einem iPhone linearisiert dargestellt

Diskussion

Gerade die Darstellung von Datentabellen ist in Mobilgeräten nicht trivial. Oftmals muss horizontal gescrollt oder gezoomt werden, um die Inhalte der einzelnen Zellen richtig lesen zu können. Hierbei geht leicht der Kontext verloren, weil die Spaltenüberschriften nicht mehr sichtbar sind.

Wir begegnen dem Problem einerseits, indem wir konsequent relative Maßeinheiten einsetzen. Gleichzeitig nutzen wir hier die Möglichkeit, einzelnen HTML-Elementen unterschiedliche Darstellungsrollen zuzuweisen. Standardmäßig haben `tr`-Elemente die Rolle `table-row`, während Zellen (`td`) als `table-cell` angezeigt werden.

Durch die Änderung in Blockelemente (`display: block`) können wir die einzelnen Datensätze als »Karteikarten« untereinander anordnen. Dadurch sind sie auch auf kleinen Bildschirmen noch gut lesbar. Da die Spaltenüberschriften in dieser Darstellung nur wenig sinnvoll sind, werden sie einfach ausgeblendet.

Die meisten mobilen Browser besitzen eine sehr gute Unterstützung für CSS3-Selektoren (`:nth-child()` usw.) und erzeugte Inhalte. Daher können wir diese Techniken hier relativ gefahrlos einsetzen, um die einzelnen Zeilen mit einer eigenen Beschriftung zu versehen.

Progressive Enhancement

Im Rahmen des Progressive Enhancement können wir noch einige Verbesserungen an der mobilen Darstellung der Tabelle vornehmen, ohne dass ihre Funktionsfähigkeit davon abhängen würde. Im ersten Schritt wollen wir den Inhalt der einzelnen Zeilen mit einer passenden Beschriftung versehen.

In der oben gezeigten Darstellung werden die Werte der einzelnen Tabellenzellen einfach untereinander angeordnet. Die Datensätze sind zwar durch die Rahmen und wechselnden Hintergrundfarben gut voneinander zu unterscheiden, da wir die Spaltenüberschriften aber ausgeblendet haben, fehlen diese Informationen jetzt möglicherweise. Durch die Verwendung erzeugter Inhalte können wir das Verständnis der einzelnen Zeilen erleichtern.

Hierfür kombinieren wir die CSS3-Pseudoklasse `:nth-child()` mit dem Pseudoelement `:before`. So können wir den einzelnen Zeilen eines Datensatzes gezielt eine eigene Beschriftung zuweisen. Die zweite Regel sorgt dafür, dass die Zeilen einheitlich formatiert werden. Da diese Anweisungen nur für die mobile Darstellung gelten sollen, umgeben wir sie mit der gleichen Medienabfrage, die wir bereits aus der Lösung kennen:

```
@media screen and (max-device-width: 480px),
  screen and (max-width: 480px) {

  td:nth-child(1)::before { content: 'Datum: '; }
  td:nth-child(2)::before { content: 'PLZ: '; }
  td:nth-child(3)::before { content: 'Ort: '; }
  td:nth-child(4)::before { content: 'Veranstaltungsort: '; }
  td:nth-child(5)::before { content: 'Uhrzeit: '; }
```

```
td:nth-child(6)::before { content: 'Kartentelefon: '; }
td:nth-child(7)::before { content: 'Bemerkungen: '; }

td::before {
    display: inline-block;
    width: 10em;
    font-weight: bold;
}
```



Bedenken Sie, dass Sie bei einer Änderung in der Tabellenstruktur auch die Angaben im Stylesheet anpassen müssen. Überlegen Sie sich daher, ob der Nutzen in einem angemessenen Verhältnis zur Wartbarkeit steht.

Der zweite Schritt zielt auf das Design ab. Wir haben in den allgemeinen Stildefinitionen für die Tabelle bereits dafür gesorgt, dass die Zellen in jeder zweiten Zeile eine andere Hintergrundfarbe erhalten. Um das Ganze optisch etwas aufzuwerten, sollen die einzelnen Zeilen und der Behälter (siehe oben) mit einem Farbverlauf versehen werden. Durch die Verwendung des `rgba()`-Formats für die Farbangaben können wir die Verläufe für Zeilen und Zellen gemeinsam angeben:

```
@media screen and (max-device-width: 480px),
       screen and (max-width: 480px) {

    tr, td {
        background-image: -webkit-gradient(linear, left top, left bottom, color-stop
        (0%,rgba(0,0,0,0)), color-stop(100%,rgba(0,0,0,0.2))); /* iOS 4.3 Safari! */
        background-image: -webkit-linear-gradient(top, rgba(0,0,0,0) 0%,rgba(0,0,0,0.2)
        100%);
        background-image: -moz-linear-gradient(top, rgba(0,0,0,0) 0%,rgba(0,0,0,0.2) 100%);
        background-image: -o-linear-gradient(top, rgba(0,0,0,0) 0%,rgba(0,0,0,0.2) 100%);
        background-image: -ms-linear-gradient(top, rgba(0,0,0,0) 0%,rgba(0,0,0,0.2) 100%);
        background-image: linear-gradient(top, rgba(0,0,0,0) 0%,rgba(0,0,0,0.2) 100%);
    }
}
```

Selbstverständlich können Sie die hier gezeigten Regelblöcke auch in einem Media Query zusammenfassen oder mit den entsprechenden Regeln aus der Lösung kombinieren. Das Endprodukt sehen Sie in Abbildung 10-5.

Siehe auch

Die Inspiration für dieses Rezept unter <http://css-tricks.com/examples/ResponsiveTables/responsive.php>; das Rezept 1.17 zu Darstellungsrollen; das Rezept 1.14 zur Verwendung der Pseudoklassen `:nth-child()` und `:empty`; das Rezept 13.7 zu erzeugten Inhalten; das Rezept 10.1 zur Definition von Medienabfragen für Mobilgeräte.



Abbildung 10-5: Die Darstellung der linearisierten Tabelle wurde erweitert

10.10 Links als Buttons darstellen

Problem

Sie möchten die Links in einer Navigationsliste als Buttons darstellen, damit sie auch auf Mobilgeräten einfach zu bedienen sind, wie in Abbildung 10-6 gezeigt.

Lösung

Ändern Sie die Darstellungsrolle in `block` und definieren Sie die Links als `Floats`. Vergessen Sie nicht, einen passenden Wert für `width` anzugeben:

```
ul, li {
    margin: 0;
    padding: 0;
    list-style: none;
}

a {
    display: block;
    float: left;
    width: 10em;
    height: 1.6em;
    margin: 0.2em;
```



Abbildung 10-6: Links in einer Navigationsliste werden als Buttons dargestellt

```
border:      1px solid orange;
border-radius: 0.4em;
background-color: rgb(254,195,82);

color:       white;
font-family: Helvetica, Arial, sans-serif;
font-weight: bold;
text-decoration: none;
text-align:  center;
line-height: 1.6em; /* = height */
}

a:hover, a:active, a:focus {
  background-color: rgb(254,171,10);
  text-decoration: underline;
}
```

Am Anfang des Stylesheets entfernen wir die Einrückungen und Aufzählungszeichen für die Liste bzw. die einzelnen Einträge. Die übrigen Definitionen nehmen wir direkt für die verschiedenen Zustände des a-Elements vor.

Diskussion

Gerade auf kleinen Displays sind die üblichen Navigationsmenüs nicht immer leicht auszuwählen. Mithilfe einer passenden Media Query können Sie die Links ab einer bestimmten Bildschirmbreite als Buttons darstellen.

Wir haben die Links in der Lösung als Blockelemente definiert. Hierdurch ist der anklick- bzw. -antippbare Bereich nicht nur auf den Linktext beschränkt, sondern wird auf die gesamte Schaltfläche ausgedehnt.

Durch die Regel `float: left` und die feste Breite von 10 em-Einheiten werden die Buttons in iOS Safari (4.2+) und Android (4.0+) nach Möglichkeit nebeneinander dargestellt. Ist nicht genügend horizontaler Platz vorhanden, werden die Links untereinander angezeigt. Mobile Browser, die (noch) keine Floats darstellen können, verwenden nur die Anweisung `display: block`. Hierdurch werden die Links untereinander angeordnet.

Eine weitere Option ist die Verwendung von zusätzlichen Medienabfragen, die die Breite und Anordnung der Buttons steuern.

Eye Candy

Das in der Lösung gezeigte Basisdesign lässt sich noch verfeinern. Um den Buttons ein eher dreidimensionales Aussehen zu geben, können Sie beispielsweise CSS3-Farbverläufe und Schattenwürfe einsetzen, wie hier gezeigt:

```
a {
    background-color: rgb(254,195,82);

    background-image: -webkit-gradient(linear, left top, left bottom,
    from(rgb(254,230,160)), to(rgb(254,195,82)));
    background-image: -webkit-linear-gradient(top, rgb(254,230,160) 0px,
    rgb(254,195,82) 100%);
    background-image: -moz-linear-gradient(top, rgb(254,230,160) 0px,
    rgb(254,195,82) 100%);
    background-image: -o-linear-gradient(top, rgb(254,230,160) 0px,
    rgb(254,195,82) 100%);
    background-image: -ms-linear-gradient(top, rgb(254,230,160) 0px,
    rgb(254,195,82) 100%);
    background-image: linear-gradient(top, rgb(254,230,160) 0px,
    rgb(254,195,82) 100%);

    -webkit-box-shadow: 2px 2px 2px rgb(100,100,100);
    box-shadow: 2px 2px 2px rgb(100,100,100);
}

a:hover, a:active, a:focus {
    background-color: rgb(254,171,10);

    background-image: -webkit-gradient(linear, left top, left bottom,
    from(rgb(254,214,125)), to(rgb(254,171,10)));
    background-image: -webkit-linear-gradient(top, rgb(254,214,125) 0px,
    rgb(254,171,10) 100%);
}
```

```

background-image: -moz-linear-gradient(top, rgb(254,214,125) 0px,
rgb(254,171,10) 100%);
background-image: -o-linear-gradient(top, rgb(254,214,125) 0px,
rgb(254,171,10) 100%);
background-image: -ms-linear-gradient(top, rgb(254,214,125) 0px,
rgb(254,171,10) 100%);
background-image: linear-gradient(top, rgb(254,214,125) 0px,
rgb(254,171,10) 100%);
}

```

Die erste Deklaration dient als Fallback, sollten Farbverläufe nicht unterstützt werden. Damit der Verlauf auch in älteren Versionen von iOS (3.2 bis 4.3) angezeigt wird, verwenden wir zwei unterschiedliche Präfixschreibweisen für Webkit-Browser. Das Ergebnis sehen Sie in Abbildung 10-7.



Abbildung 10-7: Die Links der Navigationsliste wurden optisch aufgewertet

Siehe auch

Das Rezept 6.6 zum Vergrößern des anklickbaren Bereichs; das Grundrezept 2.23 zum Thema Floats; das Rezept 6.7 zur Gestaltung von Listen als Navigationselemente; das Rezept 12.4 zur Definition von Farbverläufen mit CSS3.

10.11 Testen

Problem

Sie möchten Ihre mobile Website in möglichst vielen Mobilgeräten testen.

Lösung

Verwenden Sie ein passendes SDK oder einen Onlinesimulator für bestimmte Mobilgeräte. Zur Validierung Ihres CSS-Codes gibt es eigens einen Validator. Eine Reihe von Links zu verschiedenen Werkzeugen finden Sie im Abschnitt »Siehe auch«.

Diskussion

Es ist so gut wie unmöglich, Ihre mobilen Stylesheets mit allen möglichen Kombinationen aus Mobilgerät, System und Browser zu testen. Dennoch gibt es eine Reihe von Websites und Werkzeugen, die Ihnen dabei helfen können, Ihren Code zumindest in den wichtigsten und am weitesten verbreiteten Geräten und Browsern zu überprüfen.

Siehe auch

Die Website *haz.io* zur Überprüfung der Fähigkeiten des gerade verwendeten Browsers: <http://haz.io/>; den *Mobile OK Checker* des W3C unter <http://validator.w3.org/mobile/>; eine Liste mit Emulatoren für verschiedene Mobilgeräte unter <http://www.mobilexweb.com/emulators>; die Website *blaze.io* zum Testen der Leistungsfähigkeit Ihrer mobilen Website unter <http://www.blaze.io/mobile/>; eine Linkliste zu Werkzeugen und weiterführenden Ressourcen zum Testen und Optimieren mobiler Seiten unter <http://stevesouders.com/mobileperf/>; eine Liste mit 45 Richtlinien für die Erstellung mobiler Websites unter <http://www.mobilexweb.com/blog/guidelines-mobile-web-design>; das Buch *Mobile First* (A Book Apart) von Luke Wroblewski; eine Schablone für die Erstellung mobiler Websites und Apps unter <http://html5boilerplate.com/mobile>.

Seitenlayouts

Einführung

Die Möglichkeit, Seitenlayouts mit CSS zu gestalten, gehört zu den größten Errungenschaften im Webdesign. CSS kann Ihnen eine Menge Arbeit abnehmen und bringt dabei noch eine Reihe von Vorteilen mit sich. Aussagekräftige Inhalte können in aussagekräftigen Überschriften- und Absatzelementen platziert werden – das macht den Code besser lesbar und kann gleichzeitig helfen, die Platzierung in den Suchmaschinen zu verbessern.

Neben ein- und mehrspaltigen Standardlayouts für verschiedene Einsatzgebiete finden Sie hier Rezepte für die Verwendung von Layoutrastern (Grid-Layout), zur Definition eines »Sticky Footers«, zum Erstellen von Spalten mit gleicher Höhe und sogar für den »heiligen Gral«. Hiermit ist nicht etwa ein Trinkgefäß gemeint, sondern ein dreispaltiges Layout, das aus zwei Seitenleisten mit fester Breite und einer mittleren Hauptspalte mit flexibler Breite besteht, die im HTML-Code vor den anderen Spalten definiert werden kann.

Die hier gezeigten Layouts sind als Templates für Ihr Seitendesign gedacht, aber nicht als »schlüsselfertige« Designs. Sie bilden quasi die Grundstruktur der Gestaltung von Webseiten. Kombiniert mit den Techniken aus den anderen Kapiteln, sind sie jedoch ein wichtiger Baustein bei der Erstellung moderner Websites.

11.1 Flexible zweispaltige Layouts erstellen

Problem

Es soll ein zweispaltiges Layout erstellt werden, bei dem sich die Spaltenbreiten an der Breite des Browserfensters orientieren, wie in Abbildung 11-1 gezeigt.

Lösung

Strukturieren Sie zunächst den Inhalt mit `div`-Elementen. Versehen Sie diese mit `id`-Attributen und passenden Werten.



Abbildung 11-1: Ein mit CSS erstelltes zweispaltiges Layout

Der Hauptteil der Seite soll in diesem Beispiel links stehen, während die Seitenleiste rechts angeordnet wird. Dafür legen wir zunächst die passenden `div`-Elemente im HTML-Code an.

```
<div id="hauptteil">
  [...]
</div>
<div id="seitenleiste">
  [...]
</div>
<div id="fussteil">
  [...]
</div>
```

Ohne Stildefinitionen sieht die Darstellung jetzt aus wie die in Abbildung 11-2.

Im Stylesheet verwenden wir die Eigenschaft `float`, um den Inhalt des Hauptteils nach links zu verschieben. Gleichzeitig legen wir die Breite dieser Spalte auf zwei Drittel der Gesamtbreite des Dokuments fest:

```
#hauptteil {
  float: left;
  width: 67%;
}
```

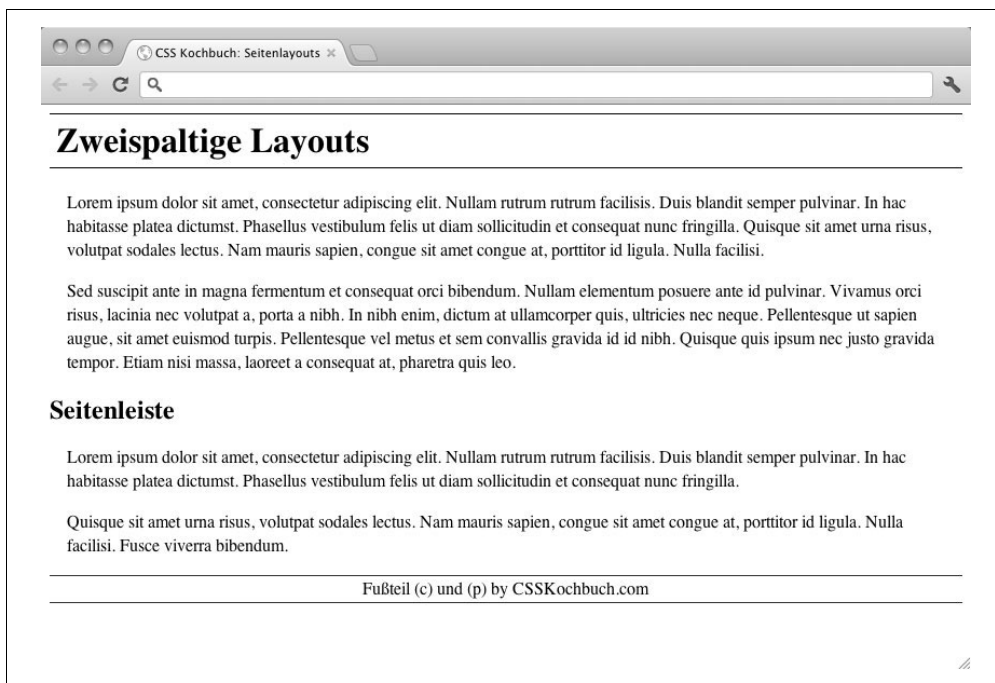


Abbildung 11-2: Die Standarddarstellung der Seite

Die rechte Spalte umfließt den Inhalt der linken Spalte. Damit die Spalte und das erste enthaltene Element auf gleicher Höhe mit der linken Spalte angezeigt werden, wird der obere Innen- und Außenabstand der Seitenleiste auf den Wert 0 gesetzt:

```
#seitenleiste {
  padding-left: 2em;
  margin-top: 0;
  padding-top: 0;
}

h1 {
  margin-top: 0;
  padding-top: 0;
}
```

Damit der Fußteil auch tatsächlich am unteren Ende des Dokuments angezeigt wird, erhält die Eigenschaft `clear` den Wert `both`:

```
#fussteil {
  clear: both;
  padding-bottom: 1em;
  border-top: 1px solid #333;
  text-align: center;
}
```

Diskussion

Die Eigenschaft `float` verhält sich so ähnlich wie das HTML-Attribut `align`, mit dem Bilder und andere Elemente links oder rechts angeordnet werden können, während die übrigen Elemente sie umfließen.

Um sicherzustellen, dass die Fußzeile auch tatsächlich unterhalb der beiden Spalten dargestellt wird, erhält die Eigenschaft `clear` für die Fußzeile den Wert `both`. Dadurch wird festgelegt, dass der Inhalt der Fußzeile die als `Float` definierte linke Spalte nicht umfließen darf und unterhalb von möglichen `Floats` dargestellt werden soll.

Bei dieser Technik zum Erstellen zweispaltiger Layouts gibt es allerdings eine Sache zu beachten: Der Inhalt der linken Spalte (unser Hauptteil) muss auf jeden Fall länger sein als der Inhalt der rechten Spalte (die Seitenleiste). Da der Inhalt der linken Spalte im Dokument zuerst auftaucht, wird diese vom Inhalt der rechten Spalte umflossen. Wenn die nicht als `Float` definierte Seitenleiste zu viel Inhalt enthält, führt das zu der in Abbildung 11-3 gezeigten Anomalie.



Abbildung 11-3: Ohne Angabe eines Außenabstands fließt der Text der Seitenleiste unterhalb des Hauptteils weiter

Dieses Problem lässt sich umgehen, indem Sie den linken Außenabstand mit der Breite des Hauptteils abgleichen. Hier die entsprechenden Regeln:

```
#hauptteil {  
  float: left;  
  width: 67%;  
}  
  
#seitenleiste {  
  margin-left: 67%;  
}
```



Sind die Werte für Rahmen und Innenabstände zu groß, kann es passieren, dass die Seitenleiste unter den Hauptteil verrutscht. In diesem Fall müssen Sie den Wert für `margin-right` der Seitenleiste vermutlich verringern.

Siehe auch

Das Rezept 11.2 zum Erstellen eines zweispaltigen Layouts mit festen Breiten; das Grundrezept 2.23 zur Verwendung von Floats.

11.2 Zweispaltige Layouts mit festen Breiten erstellen

Problem

Es soll ein zweispaltiges Layout erstellt werden, bei dem die Spalten feste Breiten besitzen.

Lösung

Zu Beginn umgeben Sie die Inhalte mit `div`-Elementen. Die einzelnen Elemente werden ihrerseits mit einem Container-`div` umgeben (siehe Abbildung 11-4):

```
<div id="container">  
  <div id="kopfteil">  
    [...]  
  </div>  
  
  <div id="navigation">  
    [...]  
  </div>  
  
  <div id="hauptteil">  
    [...]  
  </div>  
  
  <div id="fussteil">  
    [...]  
  </div>  
</div>
```

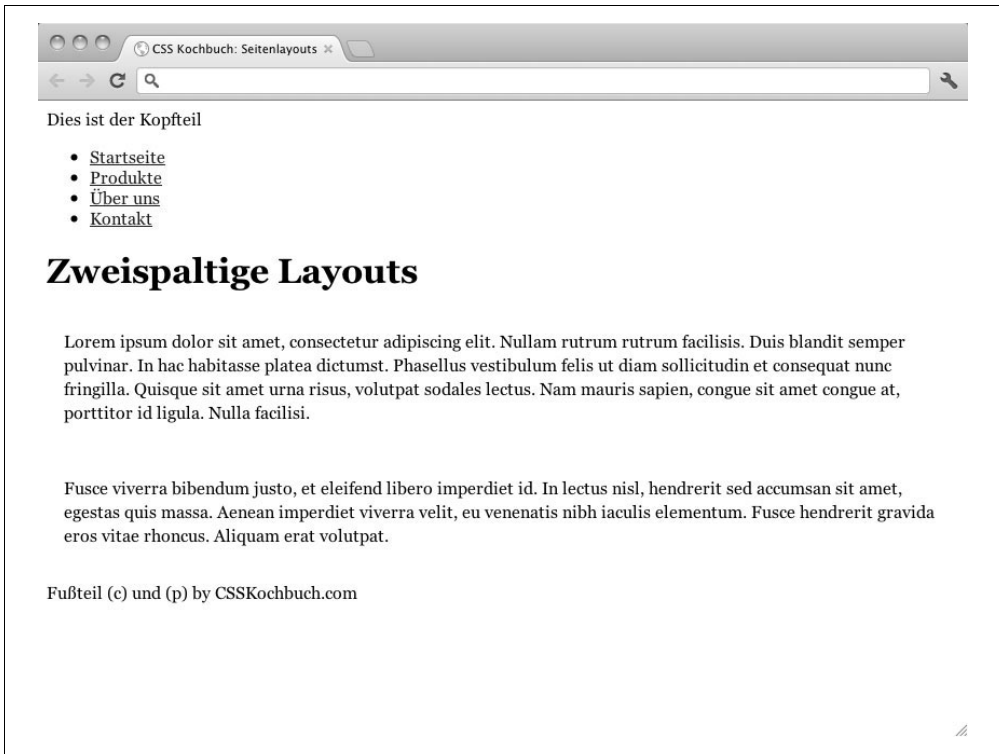



Abbildung 11-4: Die Seite in der Standarddarstellung

Verwenden Sie nun die Eigenschaft `float` und legen Sie die Breite des Containers und der linken Spalte unter Verwendung einer festen Längeneinheit fest. Die Seitenleiste mit den Navigationslinks erhält ebenfalls eine feste Breite und wird außerdem als `Float` definiert (siehe Abbildung 11-5):

```
#container {  
    width: 800px;  
    margin: auto;  
  
    border: 1px solid black;  
}  
  
#navigation {  
    float: left;  
    width: 160px;  
  
    margin-left: 10px;  
    padding-top: 1em;  
}
```

Damit sich der Hauptteil und die Seitenleiste nicht überschneiden, erhält der Hauptteil einen linken Außenabstand von 200 Pixeln, den wir mit der Kurzschrifteigenschaft `margin` angeben:

```
#hauptteil {
  padding-top: 1em;
  margin:    0 2em 0 200px;
}

#kopfteil, #fussteil {
  background-color: #ccc;
}

#kopfteil {
  border-bottom: 1px solid black;
}

#fussteil {
  clear:      left;
  padding:    5px 0 5px 200px;
  border-top: 1px solid black;

  font-size:  0.9em;
}
```



Abbildung 11-5: Das mit CSS erstellte zweispaltige Layout

Diskussion

Standardmäßig erstrecken sich Blockelemente über die Gesamtbreite des Elements, in dem sie enthalten sind. Ist das Browserfenster schmal, werden auch die Blockelemente in ihrer Breite entsprechend angepasst, wodurch der enthaltene Text umbrochen wird.

Nutzen Sie feste Längenangaben anstelle von Prozentwerten, bleibt die Breite der Spalten dagegen erhalten, selbst wenn sich die Breite des Browserfensters ändert.

Wenn Sie wollen, dass lediglich die Navigationsspalte eine feste Breite erhält, während die Breite der Hauptspalte variabel ist, brauchen Sie nur die Eigenschaft `width` für das Container-`div` zu entfernen.

HTML5

Damit auch ältere Browser mitspielen, haben wir in den Beispielen dieses Kapitels `div`-Elemente für die Strukturierung der Seite verwendet.

Alternativ dazu können Sie HTML5-Elemente benutzen. Die Struktur der Seite könnte dann so aussehen:

```
<div id="container">
  <header>      [...]  </header>
  <nav>         [...]  </nav>
  <div role="main"> [...] </div>
  <footer>      [...]  </footer>
</div>
```

Damit das in allen Browsern funktioniert, müssen Sie gegebenenfalls per JavaScript ein wenig nachhelfen. Eine Anleitung finden Sie in Kapitel 13, *Tipps und Tricks*.

Siehe auch

Das Rezept 11.1 zum Erstellen eines zweispaltigen Layouts mit flexiblen Breiten; die Rezepte 2.1 und 2.2 zur Verwendung verschiedener Längeneinheiten und der Definition von Höhen und Breiten; das Rezept 13.14 zur Definition von HTML5-Elementen in älteren Versionen des Internet Explorer.

11.3 Flexible mehrspaltige Layouts mit Floats erstellen

Problem

Es soll ein dreispaltiges Layout erstellt werden, in dem sich die Spaltenbreite an der Breite des Browserfensters orientiert (siehe Abbildung 11-6). Es soll außerdem eine Kopf- und eine Fußzeile geben, die sich jeweils über die gesamte Breite des Browserfensters erstrecken.

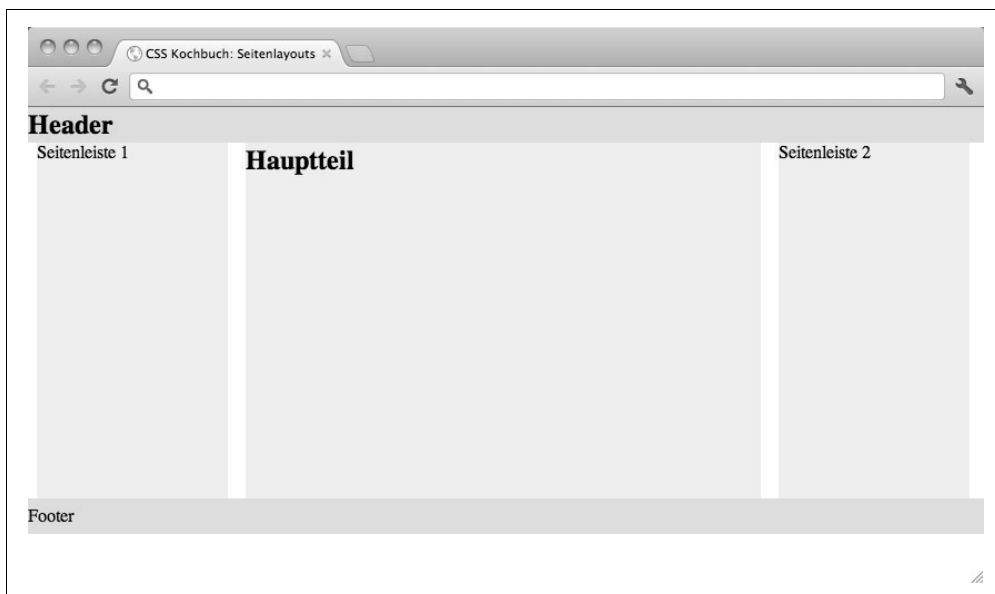


Abbildung 11-6: Ein mit CSS erstelltes dreispaltiges Layout

Lösung

Definieren Sie die Elemente, die nebeneinander angeordnet werden sollen, als Floats. Geben Sie die Breite der Elemente in Prozent an und sorgen Sie dafür, dass die gesamte Breite (inklusive Rahmen sowie Innen- und Außenabständen) 100% nicht überschreitet. Verwenden Sie für den Fußteil die Eigenschaft `clear` mit dem Wert `both`, damit er nicht von den Spalten umflossen wird.

Um eine möglichst gute Kontrolle über das Layout zu bekommen, setzen wir zunächst die Innen- und Außenabstände auf null:

```
* {
  margin: 0;
  padding: 0;
}
```

Danach werden die drei `div`-Elemente, die als Spalten dargestellt werden sollen, als links angeordnete Floats definiert:

```
#seitenleiste1, #hauptteil, #seitenleiste2 {
  float: left;
}
```

Da sich `div`-Elemente standardmäßig über die gesamte Breite des Ansichtsbereichs (Browserfensters) erstrecken, müssen sie mit einem passenden Wert für die Eigenschaft `width` versehen werden. Damit sich die Breite dynamisch an den verfügbaren Platz anpasst, verwenden wir hier Prozentwerte:

```
#hauptteil {
    width: 60%;
}

#seitenleiste1, #seitenleiste2 {
    width: 20%;
}
```

Um den Fußteil unterhalb der Floats zu platzieren, müssen wir schließlich einen passenden Wert für die Eigenschaft `clear` vergeben:

```
#fussteil {
    clear: left;
}
```

Und hier das vollständige Stylesheet, das wir für die Erstellung von Abbildung 11-6 verwendet haben:

```
* {
    margin: 0;
    padding: 0;
}

h1 {
    font-size: 1.5rem;
    line-height: 2rem;
}

#seitenleiste1, #hauptteil, #seitenleiste2 {
    float: left;
}

#hauptteil {
    width: 60%;
}

#seitenleiste1, #seitenleiste2 {
    width: 20%;
}

header, footer {
    line-height: 2rem;
}

footer {
    clear: both;
}
```

Die Definition der Hintergrundfarben und typografischen Stile fehlt aus Platzgründen an dieser Stelle. Sie finden das vollständige Stylesheet im Beispielcode zu diesem Buch unter http://examples.oreilly.de/german_examples/cssckbk3ger.

Diskussion

Diese Methode funktioniert, weil alle Spalten als links angeordnete Floats definiert wurden und zusammen nicht breiter als 100% sind. Werden die Floats rechts angeordnet, werden zwar die Spalten vertauscht, das Resultat ist aber weiterhin das gleiche.

Stellen Sie sicher, dass für die Elemente innerhalb der Spalten korrekte Innen- und Außenabstände festgelegt wurden (es sei denn, Sie kümmern sich bereits bei der Definition der Spaltenbreiten darum). Wenn Sie das nicht tun, kann es passieren, dass die Spalten zusammen breiter sind als 100%, wodurch eine oder mehrere Spalten untereinander verschoben werden können.

Anstelle von `rem` können Sie hier auch `em` als Längeneinheit verwenden. Das Problem ist allerdings der zusätzliche Rechenaufwand, da die Angabe `1em` abhängig vom Wert für `font-size` des jeweiligen Elements eine unterschiedliche Bedeutung haben kann.

Siehe auch

Das Rezept 13.16 zur Anwendung von Reset-Regeln; das Rezept 2.23 zur Verwendung der Eigenschaft `float`; das Rezept 2.1 zu den CSS-Längeneinheiten und ihren Vor- und Nachteilen.

11.4 Spalten mit Floats in beliebiger Reihenfolge darstellen

Problem

Es soll ein flexibles Layout erstellt werden, bei dem sich die Reihenfolge der Spalten nicht am HTML-Quellcode orientiert. Auf diese Weise kann der Hauptinhalt möglichst nah am Anfang des Dokuments stehen, gleichzeitig aber in der Reihenfolge der Spalten eine beliebige Position einnehmen.

Lösung

Definieren Sie Container für die einzelnen Teile der Seite (Kopfteil, Navigation, Hauptteil usw.) und definieren Sie diese als Floats. Durch den geschickten Einsatz von Außenabständen können Sie die einzelnen Spalten nach Bedarf verschieben.



Wir verwenden in diesem Beispiel die passenden HTML5-Elemente für die Kennzeichnung von Header, Navigation und Footer. Das Rezept funktioniert aber genauso gut, wenn Sie stattdessen `div`-Elemente benutzen.

Der HTML-Code könnte etwa so aussehen:

```
<header>
  <h1>Kopfteil</h1>
</header>
```

```

<div id="inhalt" class="spalte">
  <h1>Hauptinhalt</h1>
  <p>Lorem ipsum dolor sit amet, consectetur adipiscing elit...</p>
</div>

<nav class="spalte">
  <h2>Navigation</h2>
  <ul>
    <li><a href="#">Einleitung</a></li>
    <li><a href="#">Kapitel</a></li>
    <li><a href="#">Anhänge</a></li>
    <li><a href="#">Index</a></li>
  </ul>
</nav>

<div id="informationen" class="spalte">
  <h2>Informationen</h2>
  <p>Proin viverra, diam nec tempor cursus...</p>
</div>

<footer>
  Copyright © 2012 by The Swedish Chef ®
</footer>

```

Nachdem die Struktur der Seite steht, können wir das Stylesheet aufbauen. Definieren Sie zunächst die drei Spaltencontainer als Floats und legen Sie jeweils die gewünschte Breite fest:

```

#inhalt, nav, #informationen {
  float: left;
}

#inhalt      { width: 65%; }
nav          { width: 15%; }
#informationen { width: 20%; }

```

Damit der Footer auf jeden Fall unterhalb der Spalten bleibt, wird das Umfließen durch die Eigenschaft `clear` ausdrücklich unterbunden. Um zwischen den Spalten und dem Footer etwas Platz zu schaffen, erhält er außerdem etwas oberen Außenabstand:

```

footer {
  clear: both;
  margin-top: 1em;
}

```

Sowohl Header als auch Footer sollen ober- und unterhalb mit einer schwarzen Linie versehen werden. Der Text soll horizontal zentriert werden:

```

header, footer {
  border: 1px solid black;
  border-left: 0;
  border-right: 0;
  text-align: center;
}

```

Damit ist das Basislayout bereits abgeschlossen. Das Ergebnis sieht jetzt aus, wie in Abbildung 11-7 gezeigt.

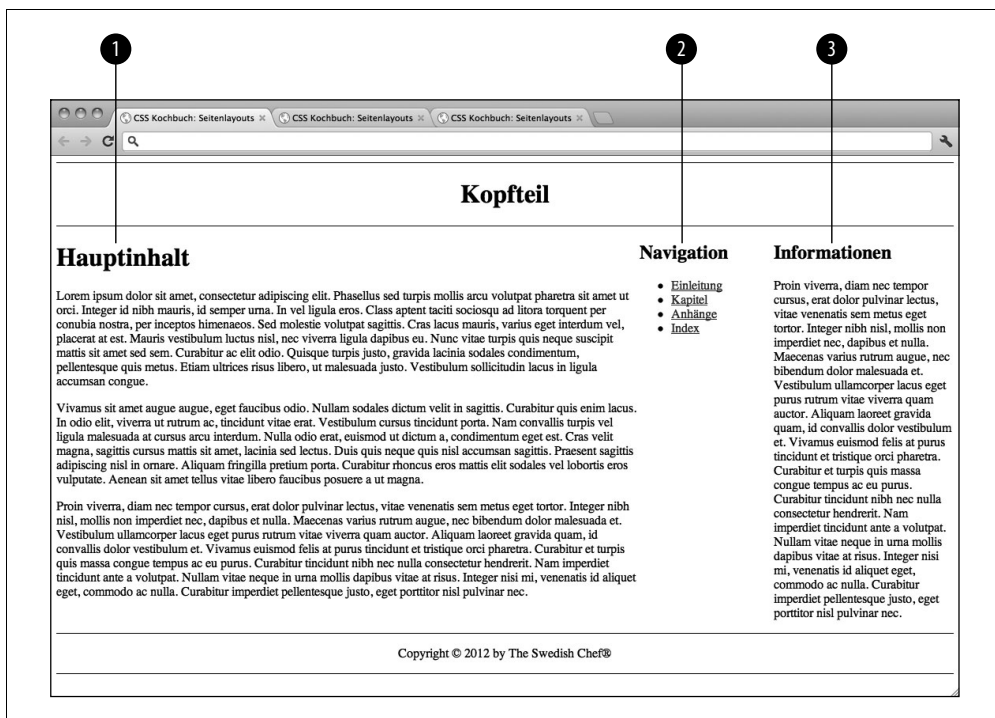


Abbildung 11-7: Die Spalten werden in der Quellcodereihenfolge dargestellt

Die Reihenfolge der Spalten ändern

Im Moment werden die Spalten noch in der Reihenfolge dargestellt, in der sie im HTML-Code definiert wurden: Zuerst kommt der Hauptinhalt ❶, darauf folgen die Navigation ❷ und ein Element mit zusätzlichen Informationen ❸.

Um diese Reihenfolge zu ändern, müssen Sie den Elementen unterschiedliche Außenabstände geben. Im Folgenden sehen Sie die nötigen Angaben für die jeweilige Reihenfolge.

1-2-3 (Hauptteil, Navigation, Informationen):

Diese Anordnung entspricht der Quellcode-Reihenfolge. Daher sind hier keine Anpassungen im Stylesheet nötig.

1-3-2 (Hauptteil, Informationen, Navigation):

```
nav { margin-left: 20%; }
#informationen { margin-left: -35%; }
```

2-1-3 (Navigation, Hauptteil, Informationen):

```
#inhalt { margin-left: 15%; }
nav { margin-left: -80%; }
```

2-3-1 (Navigation, Informationen, Hauptteil):

```
#inhalt { margin-left: 35%; }
nav { margin-left: -100%; }
#informationen { margin-left: -85%; }
```


3-2-1 (Informationen, Navigation, Hauptteil):

```
#inhalt      { margin-left: 35%; }  
nav          { margin-left: -80%; }  
#informationen { margin-left: -100%; }
```

3-1-2 (Informationen, Hauptteil, Navigation):

```
#inhalt { margin-left: 20%; }  
nav     { margin-left: -100%; }
```

Die verschiedenen Optionen zur Änderung der Reihenfolge können Sie Abbildung 11-8 entnehmen.

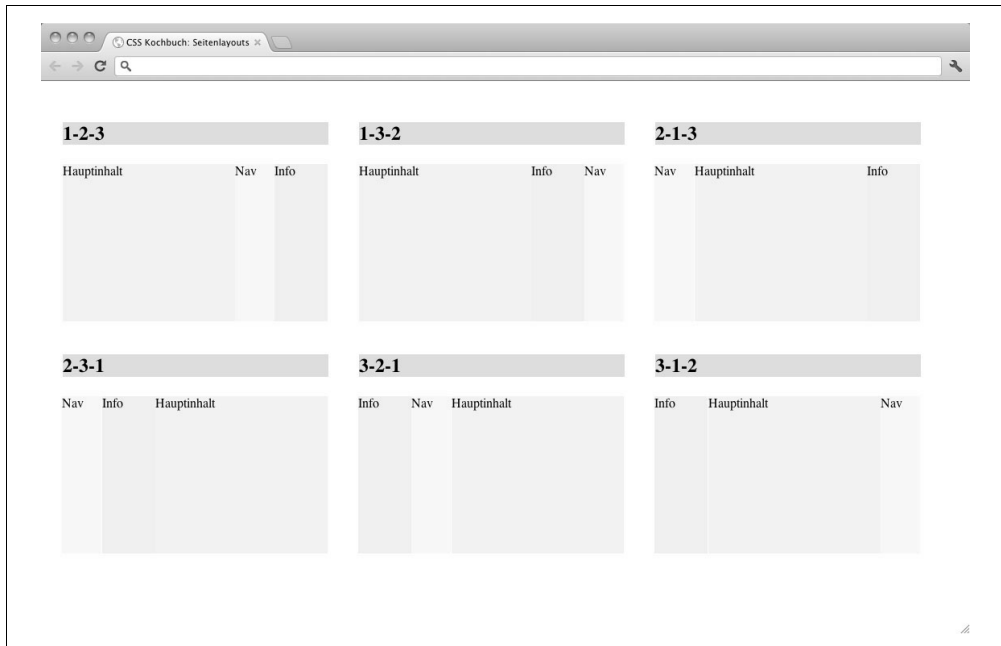


Abbildung 11-8: Sechs Möglichkeiten, die Spalten anzuordnen

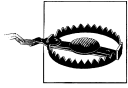
Diskussion

Bei diesem Layout schaffen Floats die Grundlage für das Verschieben der einzelnen Spalten. Die Umstellung selbst wird durch die Definition passender Außenabstände erreicht.

Da Floats sich ohne weitere Angaben über die gesamte Breite des Ansichtsbereichs erstrecken, werden die Ergebnisse erst sichtbar, wenn Sie ihnen eine Breite zuweisen. Die wird außerdem benötigt, um die Werte für `margin-left` zu berechnen.

Insgesamt sind bei dieser Vorgehensweise sechs verschiedene Anordnungen möglich, wobei die einfachste (1-2-3) bereits von der Reihenfolge der Spalten im HTML-Code vorgegeben wird. Es ist der einfachste Fall, bei dem Sie nur die Breite für die einzelnen Floats angeben müssen.

In allen anderen Fällen müssen Sie die Außenabstände der Elemente, wie in der Lösung gezeigt, anpassen.



Durch die Verwendung von Prozentwerten kann es zu Rundungsfehlern kommen, wodurch einzelne Spalten möglicherweise nach unten verschoben werden. Diesem Problem können Sie begegnen, indem Sie das umgebende Element (in unserem Fall `body`) mit ausreichendem Innenabstand (`padding`) versorgen.

Leerraum schaffen

Ohne weitere Angaben grenzen die Floats direkt aneinander. Um zwischen den Spalten etwas Platz zu schaffen, kann die Eigenschaft `margin` jedoch nicht verwendet werden. Sie wird bereits gebraucht, um die Spalten zu verschieben.

Nach Möglichkeit wollen wir keine zusätzlichen HTML-Elemente einfügen. Daher weichen wir auf die Eigenschaft `padding` aus. Hierbei ist zu beachten, dass `padding` gemäß dem CSS-Boxmodell zur Gesamtbreite (hier ist nicht der Wert von `width` gemeint) des Elements beiträgt. Wenn Sie die Spalten mit Innenabstand versehen, können sie nicht mehr nebeneinander dargestellt werden.

Um das zu verhindern, müssen die Breiten der jeweiligen Elemente angepasst werden, was wiederum Auswirkungen auf die Werte für `margin-left` hat. Als Beispiel verwenden wir hier die Reihenfolge 3-1-2 (Navigation, Hauptteil, Informationen). Jede Spalte soll einen Innenabstand von 1% erhalten:

```
#inhalt, nav, #informationen {  
  float: left;  
  padding: 1%;  
}
```

Im nächsten Schritt passen wir die Breite der Spalten an die neuen Verhältnisse an, indem wir den Wert für `width` jeweils um 2% reduzieren:

```
#inhalt      { width: 63%; }  
nav          { width: 13%; }  
#informationen { width: 18%; }
```

Die Werte für die Verschiebung per `margin-left` müssen hierbei nicht angepasst werden. Sie ergeben sich aus der Summe der Breitenangabe und den Werten für `padding`. Inklusive der Innenabstände sieht unser Layout nun aus wie das in Abbildung 11-9.

Das hier gezeigte Basislayout können Sie nun nach Belieben mit weiteren Stildefinitionen versehen.

Siehe auch

Das Rezept 2.23 zur Verwendung von Floats; das Rezept 1.16 zum CSS-Boxmodell; das Rezept 13.8 mit einer Möglichkeit, die Breitenberechnung des Boxmodells selbst zu steuern; die Rezepte 2.1 und 2.2 zur Verwendung von Längeneinheiten; das Rezept 11.5 zur Definition von »Faux Columns«; das Rezept 11.6 zum »heiligen Gral« des mehrspaltigen Layouts.

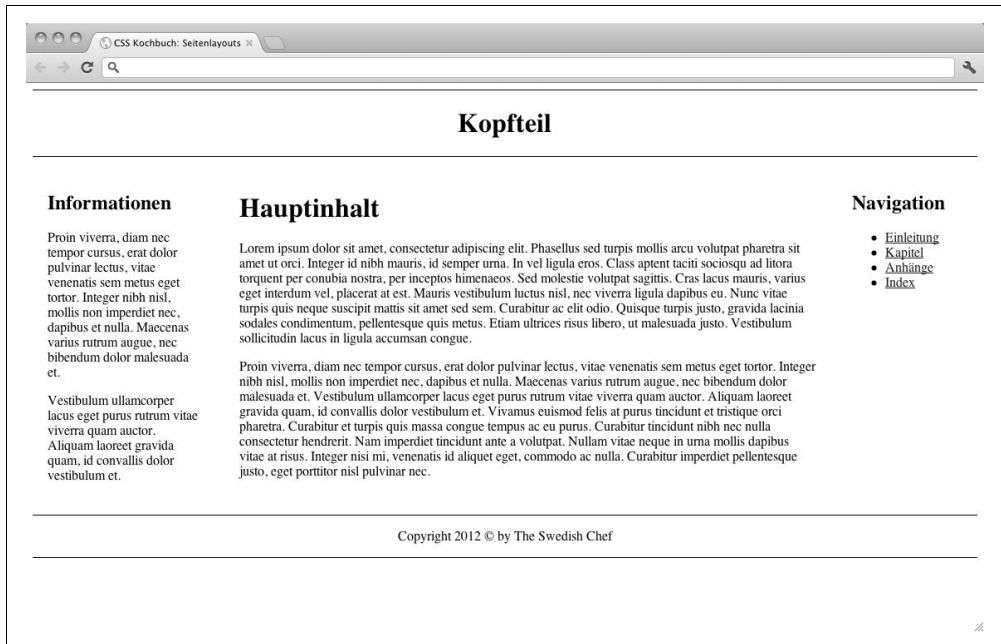


Abbildung 11-9: Die Spalten wurden mit Innenabständen versehen

11.5 Faux Columns einsetzen

Problem

Es wurde ein mehrspaltiges Layout mit Floats realisiert. Trotz unterschiedlich hoher Inhalte sollen die Spalten immer gleich hoch erscheinen.

Lösung 1

Verwenden Sie die klassische »Faux Columns«-Technik. Definieren Sie hierfür bei Bedarf ein Containerelement, für das ein entsprechend präpariertes Hintergrundbild definiert wurde. Das Hintergrundbild für den Container können Sie in einem beliebigen Bildbearbeitungsprogramm erstellen. Wichtig ist hierbei, dass die Breiten der verschiedenfarbigen Bereiche den Spaltenbreiten entsprechen. Danach können Sie das Bild als Hintergrund in Ihr Stylesheet einbinden:

```
#container {
    background-image: url('bilder/faux.gif');
    background-repeat: repeat-y;
    background-position: center top;
}
```

Die Breite des Hintergrundbilds muss mindestens der Gesamtbreite aller Spalten inklusive möglicher Rahmen, Innen- und Außenabstände entsprechen. Die Höhe ist dabei weniger wichtig, da das Bild horizontal (entlang der y-Achse) wiederholt wird. Das Ergebnis sehen Sie in Abbildung 11-10.

Anhand von `background-position` wird das Bild an der Oberkante des Containerelements zentriert ausgerichtet.

Lösung 2

Nutzen Sie die Möglichkeit, mit CSS3 mehrere Hintergrundbilder für ein Element zu definieren. Anstelle von Hintergrundbildern verwenden wir hier jedoch Farbverläufe, deren Anfangs- und Endwert gleich sind:

```
body {  
  background-image:  
    linear-gradient(#f00, #f00),  
    linear-gradient(#0f0, #0f0),  
    linear-gradient(#00f, #00f);  
  
  background-position: 0px 0px, -200px 0px, 100% 0px;  
  background-size: 150px 100%, auto 100%, 200px 100%;  
  background-repeat: no-repeat, no-repeat, no-repeat;  
}
```

Zunächst definieren wir sozusagen als Hintergrundbilder drei Farbverläufe mit der Funktion `linear-gradient`. Hierbei wird als Start- und Endwert jeweils die gleiche Farbe verwendet.

Da bei der Definition von Hintergründen das zuerst definierte Bild die danach angegebenen überlagert, müssen wir hier ein wenig um die Ecke denken. Am besten lesen Sie die Werte für `background-position` von rechts nach links. Das dritte Wertepaar platziert das rechte Hintergrundbild am rechten Rand und an der Oberkante des Ansichtsbereichs. Der zweite Wert definiert den Hintergrund für die Hauptspalte. Damit dieser die rechte Spalte nicht überlagert, positionieren wir ihn durch den negativen Wert für die horizontale Platzierung 200 Pixel vom rechten Rand entfernt. Der zuerst definierte Farbverlauf wird hingegen in der linken oberen Ecke verankert.

Mithilfe der Eigenschaft `background-size` können Sie angeben, welche Dimensionen die einzelnen Hintergrundbilder (bzw. Farbverläufe) haben sollen. Hierbei gibt der erste Wert die Breite an, der zweite steht für die Höhe. Die Werte entsprechen der Breite der jeweiligen Spalte, für die mittlere Spalte verwenden wir dagegen den Wert `auto`, wodurch sich die Breite automatisch dem verfügbaren Platz anpasst.



Wenn Sie für die Spalten Rahmen, Innen- oder Außenabstände definieren, müssen Sie diese bei der Definition der Hintergründe entsprechend berücksichtigen.

Da sich die Hintergründe über die gesamte Höhe des umgebenden Elements erstrecken sollen, erhalten alle drei den zweiten Wert 100%.

Abschließend wird festgelegt, dass die Hintergründe nicht wiederholt werden sollen. Anderenfalls würden sich alle drei Hintergründe über die gesamte Breite des Ansichtsbereichs erstrecken. Zu sehen wäre dann nur der erste definierte Farbverlauf. Das fertige Ergebnis sehen Sie in Abbildung 11-10.

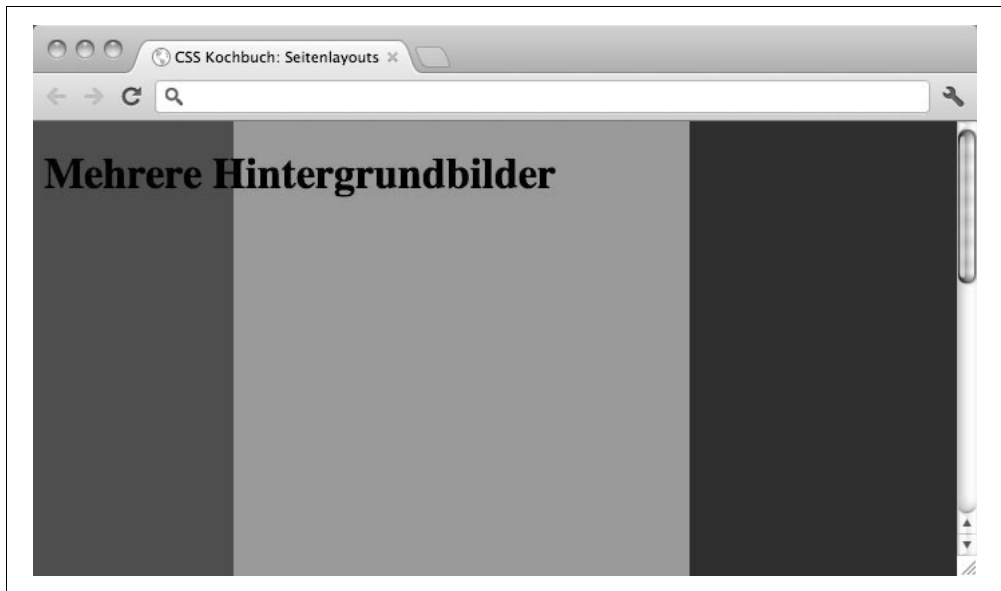


Abbildung 11-10: Mithilfe mehrerer Hintergrundbilder wird der Eindruck erweckt, alle drei Spalten hätten die gleiche Höhe

Um das Codebeispiel nicht unnötig zu verlängern, haben wir hier nur die offizielle Schreibweise für die Erzeugung von Farbverläufen angegeben. Bei Bedarf sollten Sie diese um die entsprechenden Präfixschreibweisen (-webkit, -moz usw.) ergänzen.

Diskussion

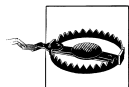
In den wenigsten Fällen ist der Inhalt aller Spalten gleich hoch. Ohne weitere Maßnahmen werden die Hintergründe ebenfalls nur so hoch dargestellt, wie die Spalten tatsächlich sind. Die hier gezeigten Techniken »fälschen« die Spalten durch die Definition von Hintergrundbildern für ein umgebendes Element.

Tatsächlich muss hier auf den Einsatz von Bildern zurückgegriffen werden, da anhand der verschiedenen background-Eigenschaften jeweils nur eine Hintergrundfarbe definiert werden kann. Ein Bild können Sie dagegen nach eigenen Wünschen gestalten.

Allerdings sind Bilder auch relativ unflexibel. Sobald sich die Spaltenbreite ändert oder Sie Rahmen, Innen- oder Außenabstände definieren wollen, muss das Hintergrundbild möglicherweise neu erstellt werden. Die in Lösung 1 gezeigte Technik ist daher ohne weitere Anpassungen in erster Linie für Layouts mit festen Breiten geeignet. Soll ein flexibles Layout gestalten werden, das sich dem verfügbaren Platz ganz oder teilweise anpasst, kommen Sie um zusätzlichen Markup-Code kaum herum.

Lösung 2 stellt ein Verfahren vor, das gänzlich ohne Bilddateien auskommt. Es nutzt die Möglichkeit, Farbverläufe anstelle von Bildern zu verwenden. Sollen die Hintergründe einfarbig sein, reicht es, als Anfangs- und Endfarbe jeweils den gleichen Wert anzugeben. Es sind aber auch komplexere Anwendungen denkbar.

Durch die richtigen Angaben für die Eigenschaften `background-position` und `background-size` können Sie im Prinzip beliebig viele Spalten mit eigenen Hintergründen versehen. Allerdings müssen Sie hierbei beachten, dass das zuerst definierte Hintergrundbild (bzw. der Farbverlauf) die danach angegebenen überlagert.



Die in Lösung 2 gezeigte Technik basiert auf Eigenschaften, die zwar von den meisten aktuellen Browsern in der Präfixschreibweise unterstützt werden, sich aber möglicherweise noch ändern können. Die dazugehörige Spezifikation befindet sich, während dieses Buch geschrieben wird, noch im Stadium eines Arbeitsentwurfs (Working Draft). Sie sollten dieses Verfahren daher auf jeden Fall sorgfältig testen.

Damit der Hintergrund für die erste Spalte die anderen beiden nicht überlagert, erhält er per `background-size` eine feste Breite. Das ist für den zweiten Hintergrund nicht möglich. Schließlich soll er sich flexibel an die Breite des Ansichtsbereichs anpassen.

Durch die Verschiebung nach links mittels eines negativen Werts für den ersten Parameter von `background-position` können wir Platz für den Hintergrund der dritten Spalte schaffen. Tatsächlich erstreckt sich der Hintergrund bis zur linken Kante des Ansichtsbereichs. Allerdings wird er hier vom ersten Hintergrund verdeckt. Trotzdem kann er sich bei Bedarf nach rechts ausdehnen.

Der Hintergrund für die linke Spalte wird an der rechten Kante des Ansichtsbereichs ausgerichtet und erhält ebenfalls eine feste Breite, die den Maßen der linken Seitenleiste entspricht. Durch die Verschiebung des Hintergrunds für die Hauptspalte um genau diesen Wert nach links bleibt der Hintergrund der linken Spalte ebenfalls sichtbar.

Siehe auch

Den ursprünglichen Artikel über »Faux Columns« auf der Website von *A List Apart* unter <http://www.alistapart.com/comments/fauxcolumns/>; eine umfassende Liste mit verschiedenen Methoden zur Erstellung von Faux Columns mit flexibler Breite unter <http://css-tricks.com/fluid-width-equal-height-columns/>; das Rezept 2.12 zur Definition mehrerer Hintergrundbilder; das Rezept 12.4 zur Definition von Farbverläufen.

11.6 Den »heiligen Gral« verstehen

Problem

Sie möchten den »heiligen Gral« finden. Das bedeutet in diesem Zusammenhang: ein dreispaltiges Layout mit zwei Seitenleisten mit fester Breite – eine zum Beispiel für die Navigation, die andere für Dinge wie die Liste Ihrer letzten Blogartikel oder Ähnliches – und einer Hauptspalte für den eigentlichen Inhalt, die sich an die Breite des verfügbaren Ansichtsbereichs anpasst.

Hierbei soll es möglich sein, dass:

- die Hauptspalte im Quellcode vor den Seitenleisten definiert wird
- jede Spalte eine beliebige Höhe hat (siehe Rezept 11.5)
- höchstens ein zusätzliches `div`-Element im Markup-Code verwendet wird
- möglichst einfaches CSS mit nur wenigen browserspezifischen Anpassungen verwendet wird

Lösung

Umgeben Sie die Spalten mit einem `div`-Element und definieren Sie die einzelnen Spalten als Floats. Damit wir dies nicht für jede Spalte einzeln tun müssen, verwenden wir für alle drei ein gemeinsames `class`-Attribut:

```
<div id="container">
  <div id="hauptteil" class="spalte">
    <h1>Hauptteil</h1>
    [mehr Inhalt]
  </div>

  <div id="seitenleiste1" class="spalte">
    <h2>Seitenleiste 1</h2>
    [mehr Inhalt]
  </div>

  <div id="seitenleiste2" class="spalte">
    <h2>Seitenleiste 2</h2>
    [mehr Inhalt]
  </div>
</div>
```

Mithilfe der Eigenschaften `padding-left` bzw. `padding-right` für den Container schaffen Sie links und rechts der Hauptspalte den nötigen Platz für die beiden Seitenleisten. Danach versehen Sie die Eigenschaften `margin-left` für die linke und `margin-right` für die rechte Seitenleiste mit einem negativen Wert, um sie an ihren Platz zu verschieben. Das Resultat sehen Sie in Abbildung 11-11.

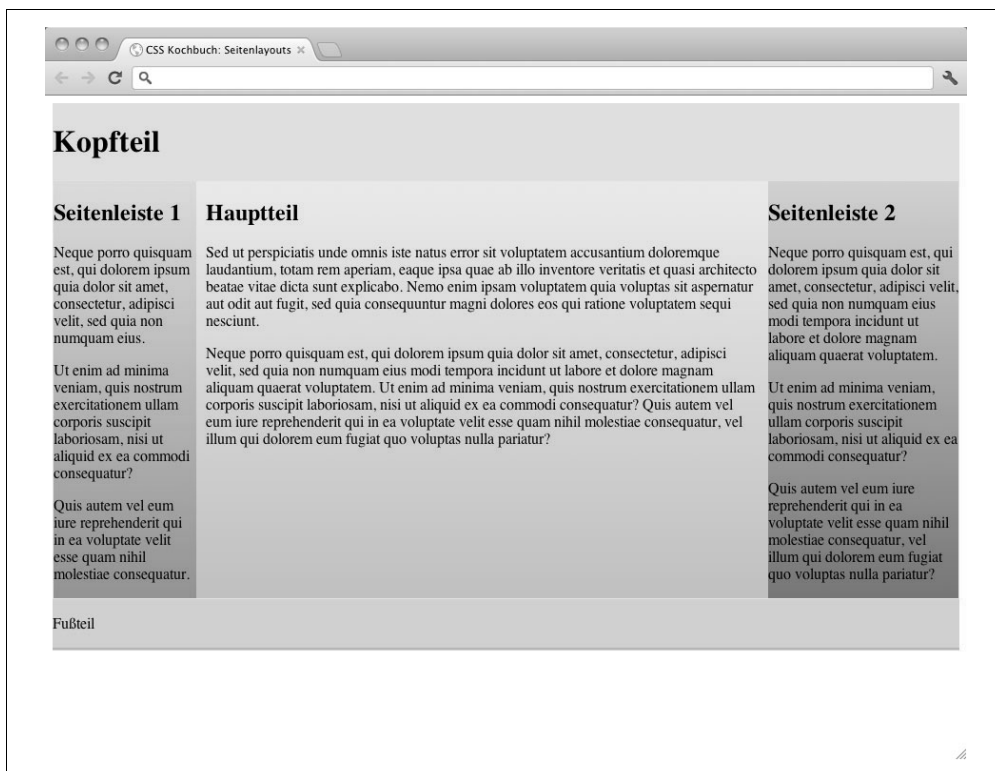


Abbildung 11-11: Der »heilige Gral« des Seitenlayouts: korrekte Quellcodereihenfolge, Seitenleisten mit fester Breite und eine flexible Hauptspalte

Hier der CSS-Code:

```
#container {
  padding-left: 150px;
  padding-right: 200px;
}
```

Die Innenabstände des Containers entsprechen den Breiten der Seitenleisten (links 150 Pixel, rechts 200 Pixel).

```
#container .spalte {
  float: left;
  position: relative;
}
```

Da wir zuvor die einzelnen Klassen mit dem gleichen class-Attribut versorgt haben, können wir nun alle drei auf einmal als links angeordnetes Float definieren. Die relative Positionierung wird benötigt, um die linke Spalte (zumindest in diesem Beispiel) korrekt platzieren zu können, wie wir gleich sehen werden.

```
#hauptteil {
  width: 100%;
}
```


An dieser Stelle ist es gut, sich noch einmal das Boxmodell (Rezept 1.16) vor Augen zu führen. Ein Prozentwert für die Eigenschaft `width` bezieht sich auf die Breite des Inhaltsbereichs des Elternelements. Damit ist der Hauptteil so breit wie das Containerelement minus dessen seitliche Außenabstände.

```
#seitenleiste1 {  
    margin-left: -100%;  
    width:      150px;  
    right:      150px;  
}
```

Gemäß ihrer Position im HTML-Code würde die linke Seitenleiste (`#seitenleiste1`) neben oder unterhalb der Hauptspalte stehen. Durch die Verwendung von `margin-left` mit einem negativen Wert können wir die Seitenleiste nach links verschieben. Ohne weitere Angaben würden die Seitenleiste und die Hauptspalte nun voreinander angeordnet. Da wir zuvor alle Spalten relativ positioniert haben (siehe oben), können wir nun die Eigenschaft `right` verwenden, damit die Spalte um das fehlende Stück (150 Pixel) nach links geschoben wird.

Die zweite Seitenleiste soll 200 Pixel breit sein. Hier reicht es aus, einen entsprechenden Wert für `margin-right` anzugeben, um auch die rechte Spalte korrekt zu platzieren.

```
#seitenleiste2 {  
    width:      200px;  
    margin-right: -200px;  
}
```

Diskussion

Diese Methode wurde erstmals 2006 auf der Website *A List Apart* beschrieben. Bis dahin schien es nahezu unmöglich, die verschiedenen Anforderungen (sinnvolle Quellcodereihenfolge, flexible Hauptspalte, Seitenleisten mit fester Breite) zu erfüllen.

Die Aufgabenstellung basiert auf einem häufig verwendeten Standardlayout für Desktopbrowser. Der Hauptteil steht in der Mitte und wird zu beiden Seiten von je einer Seitenleiste flankiert. Das war zwar auch schon vorher möglich, bedurfte aber einer Reihe zusätzlicher `div`-Elemente. Sollte die Position der Spalten verändert werden, waren Änderungen am HTML-Code unumgänglich.

Wir haben in der Lösung die »Rohfassung« gezeigt. Ohne weitere Angaben kann es jedoch passieren, dass die Spalten auf kleineren Bildschirmen untereinander verschoben werden. Diesem Problem können wir begegnen, indem wir das `body`-Element mit einer minimalen Breite versehen. Der Wert entspricht der doppelten Breite der linken Spalte plus der Breite der rechten Spalte:

```
body {  
    min-width: 550px;  
}
```

Innenabstände für die Hauptspalte

Soll die Hauptspalte zusätzlich mit Innenabständen versehen werden, muss der Code ein wenig abgeändert werden:

```
#hauptteil {  
  width:      100%;  
  padding-left: 10px;  
  padding-right: 10px;  
}
```

Als Folge muss auch die Breite des Containers entsprechend erweitert werden:

```
#container {  
  padding-left: 150px;  
  padding-right: 220px;  
}
```

Da die Platzierung der Seitenleisten direkt damit zusammenhängt, wie viel Platz insgesamt für die Hauptspalte benötigt wird, müssen auch hier noch ein paar Anpassungen vorgenommen werden:

```
#seitenleiste1 {  
  margin-left: -100%;  
  width:      150px;  
  right:      170px;  
}  
  
#seitenleiste2 {  
  width:      200px;  
  margin-right: -220px;  
}
```

Für die linke Seitenleiste muss der Wert der Eigenschaft `right`, für die rechte Seitenleiste der Wert von `margin-right` entsprechend den Innenabständen des Hauptteils vergrößert werden.

Siehe auch

Den Originalartikel von Mathew Levine in *A List Apart* unter <http://www.alistapart.com/articles/holygrail>; den Artikel *In Search of the One True Layout* unter <http://www.positioniseverything.net/articles/onetruelayout/>; das Rezept 1.16 zum Boxmodell; die Grundrezepte 2.5 und 2.6 zur Verwendung der Eigenschaften `margin` und `padding`.

11.7 Den Fußteil an der Unterkante des Browserfensters verankern

Problem

Sie möchten einen Fußteil (Footer) fest an der Unterkante des Ansichtsbereichs verankern, wie in Abbildung 11-12 gezeigt wird.

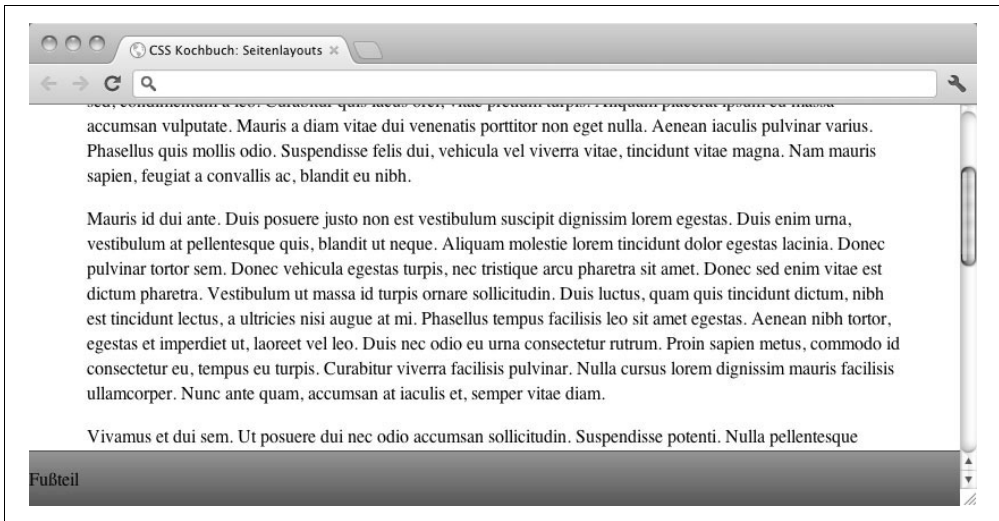


Abbildung 11-12: Der Fußteil wurde an der Unterseite des Browserfensters verankert

Lösung

Umgeben Sie den Fußteil mit einem eigenen `div`-Element, wie hier gezeigt:

[... weitere Inhalte hier ...]

```
<div id='fussteil'>
Hallo, ich bin der Fußteil.
</div>

</body>
</html>
```

Danach können Sie den Fußteil per `position: fixed` im Ansichtsbereich verankern:

```
#fussteil {
    position: fixed;
    bottom: 0;

    height: 3em;
}

#container {
    margin-bottom: 3em;
}
```

Die Deklaration `bottom: 0` gibt an, dass die Unterseite des Fußteils bündig mit der Unterseite des Ansichtsbereichs dargestellt werden soll. Außerdem erhält der Fußteil eine feste Höhe (`height: 3em`). Diesen Wert verwenden wir in der folgenden Regel als Wert für `margin-bottom`, damit der Fußteil den unteren Teil des Containers nicht verdeckt.

Diskussion

Standardmäßig hat die Eigenschaft `position` den Wert `static`. Das bedeutet, Elemente werden entsprechend ihrer Position im HTML-Quelltext auf der Seite angeordnet und beim Scrollen der Seite mitbewegt. Wird dagegen der Wert `fixed` verwendet, wird das Element komplett vom Dokumentenfluss ausgenommen. Wie bei Verwendung des Werts `absolute` kann das Element anhand der Eigenschaften `top`, `left`, `bottom` und `right` frei platziert werden. Im Gegensatz zu `position: absolute` wird das Element bei `position: fixed` jedoch nicht mitgescrollt.

Ohne einen unteren Außenabstand für das Containerelement würde dessen unterer Teil vom Fußteil verdeckt werden. Durch die Angabe einer Höhe für den Fußteil und eines entsprechenden Werts für `margin-bottom` des Containers wird dieses Problem behoben.

Siehe auch

Das Rezept 11.9 zur Erstellung eines »Sticky Footers«; das Rezept 2.20 zur festen Positionierung von Elementen; das Rezept 2.5 zur Eigenschaft `margin-bottom`.

11.8 Layouts mit einem »Sticky Footer« versehen

Problem

Sie möchten Ihr Layout mit einem Fußteil versehen, der am unteren Teil des Ansichtsberreichs »haftet«, auch wenn der übrige Inhalt nicht die gesamte Höhe des Browserfensters einnimmt. Benötigt der Inhalt dagegen mehr Platz, soll der Fußteil mit nach unten gescrollt werden. Ein Beispiel sehen Sie in Abbildung 11-13.



Abbildung 11-13: Die Seite wurde mit einem Sticky Footer versehen

Lösung

Umgeben Sie die Inhalte der Seite und den Fußteil mit eigenen div-Elementen:

```
<body>
  <div id='container'>
    <div id='inhalte'>
      <!-- Hier stehen die Inhalte der Seite -->
    </div>
  </div>

  <div id='fussteil'>
    <!-- Inhalt des Fußteils -->
  </div>
</body>
```

Im Stylesheet geben Sie den Elementen `html` und `body` jeweils eine Höhe von 100%, für das Containerelement wird festgelegt, dass es mindestens 100% der Höhe des umgebenden Blocks (des `body`-Elements) haben soll. Außerdem werden die Außen- und Innenabstände für diese Elemente entfernt:

```
body, html {
  height: 100%;
  margin: 0;
  padding: 0;
}

#container { min-height: 100%; }
```

Im nächsten Schritt wird für den Fußteil ausdrücklich eine Höhe festgelegt. Da er ohne weitere Angabe unterhalb des Ansichtsbereichs stehen würde, verwenden wir die Eigenschaft `margin-top` mit einem negativen Wert, um ihn wieder nach oben zu schieben:

```
#fussteil {
  height: 150px;
  margin-top: -150px;

  clear: both;
}
```

Hier erhält der Fußteil eine Höhe von 150 Pixeln. Dieser Wert wird in negativer Form für die Eigenschaft `margin-top` wiederholt. Die Angabe von `clear: both` sorgt dafür, dass Lösung auch mit Float-basierten Layouts, die wie weiter oben in diesem Kapitel beschrieben, funktioniert.

Damit der Hauptinhalt nicht verdeckt wird, müssen wir an seiner Unterseite entsprechend der Höhe des Fußteils Platz schaffen. Wir benutzen hierfür die Eigenschaft `padding-bottom`, da `margin-bottom` den Fußteil nach unten verschieben würde. Der Wert entspricht sinnvollerweise der Höhe des Fußteils:

```
#inhalte {
  padding-bottom: 150px;
}
```

Damit ist die Grundversion des »Sticky Footer« fertig. Das gesamte Stylesheet sieht nun so aus:

```
html, body {
  height: 100%;
  margin: 0;
  padding: 0;
}

#container {
  min-height: 100%;
}

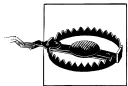
#inhalte {
  padding-bottom: 150px;
  overflow: auto;
}

#fussteil {
  height: 150px;
  margin-top: -150px;
  clear: both;
}
```

Diskussion

Das Geheimnis dieser Lösung besteht in dem geschickten Einsatz der Eigenschaften `height`, `min-height` und `margin`. Zunächst wird festgelegt, dass die Seite selbst wie auch das Containerelement den Ansichtsbereich komplett ausfüllen sollen. Da der Fußteil im Quelltext nach dem Container definiert wird, wird er ohne weitere Angaben außerhalb des Ansichtsbereichs dargestellt. Um das zu verhindern, muss der Fußteil um die angegebene Höhe (150 Pixel) wieder nach oben verschoben werden. Dafür kommt hier die Eigenschaft `margin-top` mit einem negativen Wert zum Einsatz, der exakt der Höhe des Fußteils entspricht.

Jetzt ist der Fußteil zwar sichtbar, verdeckt aber die Unterkante des Hauptinhalts. Wir können den verdeckten Teil nun wieder sichtbar machen, indem wir für das letzte Element innerhalb von `#container` (in unserem Beispiel ist dies `#inhalte`) einen passenden Wert für die Eigenschaft `padding-bottom` festlegen.



Diese Lösung hat einen kleinen Haken: Wenn Sie für die enthaltenen Elemente vertikale Außenabstände definieren, wirken sich diese auf die Platzierung des Fußteils aus. Brauchen Sie ober- und unterhalb der Elemente innerhalb von `#inhalte` Platz, müssen Sie hierfür mit der Eigenschaft `padding` arbeiten.

Die Eigenheiten von IE8 ausbügeln

Der Internet Explorer 8 hat Schwierigkeiten, die Höhe des #container-Elements anzupassen, wenn die Größe des Browserfensters verändert wird. Eine flüssigere Darstellung erreichen Sie, wenn Sie anhand von Conditional Comments die Regeln ein wenig anpassen. Erweitern Sie hierbei den Kopfteil Ihrer Seite um den folgenden Code:

```
<!--[if IE 8]>
<style type="text/css">
  #container {
    display:table;
    height: 100%;
  }
</style>
<![endif]-->
```

Diese Lösung basiert auf dem *Sticky Footer*-Stylesheet von Steve Hatcher. Im Gegensatz zu anderen Herangehensweisen kommt sie ohne zusätzliche HTML-Elemente oder einen »Clearfix-Hack« aus. Eine der ersten Erwähnungen des »Sticky Footer«-Konzepts findet sich auf der Website von Brian Fait (Details im Abschnitt »Siehe auch«).

Siehe auch

Das Rezept 11.8 zum Verankern des Fußteils an der Unterseite des Ansichtsbereichs; das Rezept 13.9 zur Clearfix-Methode; die Website, auf der die hier gezeigte Lösung basiert, unter <http://www.cssstickyfooter.com>; den Artikel *Exploring Footers* unter <http://www.alistapart.com/articles/footers/>; die ursprüngliche »Sticky Footer«-Methode von Ryan Fait unter <http://ryanfait.com/resources/footer-stick-to-bottom-of-page/>.

11.9 Grid-System benutzen

Problem

Sie möchten Ihr Seitenlayout an einem Raster aus Gitternetzlinien ausrichten, um die Elemente der Seite ohne viel Aufwand gleichmäßig anordnen zu können, wie in Abbildung 11-14 gezeigt.

Lösung

Verwenden Sie eines der vielen Grid-Frameworks, um das Layout Ihrer Seite zu steuern. Wir demonstrieren das Konzept am Beispiel des minimalistischen *1kb CSS Grid* von Tyler Tate:

```
.raster_1 { width: 60px; }
.raster_2 { width: 140px; }
.raster_3 { width: 220px; }
.raster_4 { width: 300px; }
.raster_5 { width: 380px; }
```

```

.raster_6 { width: 460px; }
.raster_7 { width: 540px; }
.raster_8 { width: 620px; }
.raster_9 { width: 700px; }
.raster_10 { width: 780px; }
.raster_11 { width: 860px; }
.raster_12 { width: 940px; }

.spalte {
  margin: 0 10px;
  overflow: hidden;
  float: left;
  display: inline;
}
.zeile {
  width: 960px;
  margin: 0 auto;
  overflow: hidden;
}
.zeile .zeile {
  margin: 0 -10px;
  width: auto;
  display: inline-block;
}

```

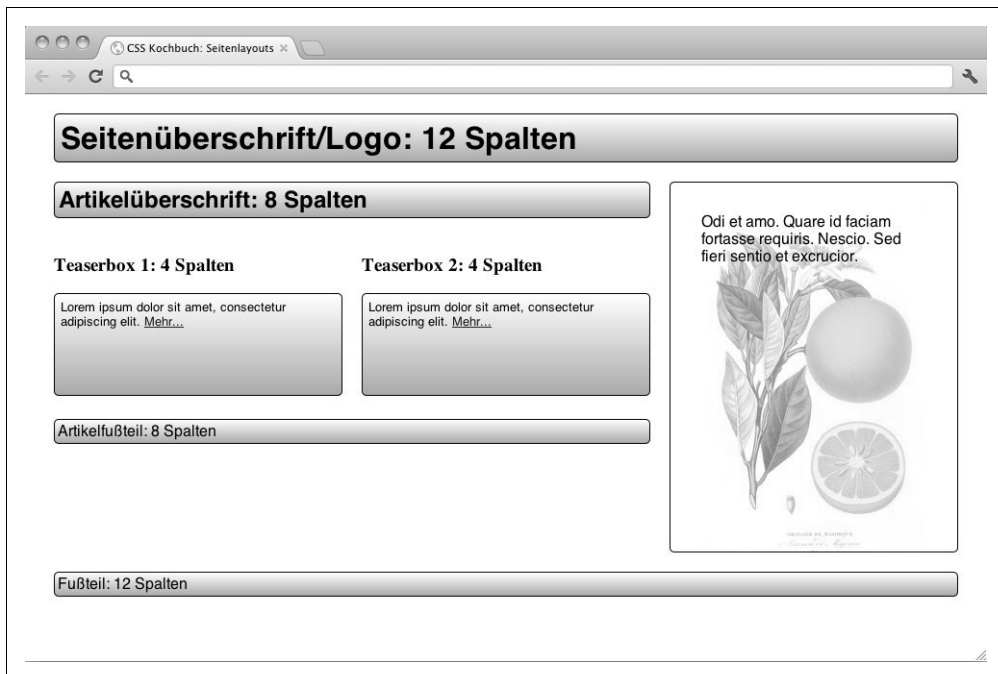


Abbildung 11-14: Ein Seitenlayout, das mit einem Grid-Framework erzeugt wurde

Die oben stehenden CSS-Regeln entsprechen dem vollständigen Code des *1kb CSS Grid*. Wir haben hier lediglich die Formatierung ein wenig lesbarer gemacht und die Klassennamen eingedeutscht, damit sie besser verständlich sind.

Zunächst wird festgelegt, wie breit einzelne bzw. mehrere Spalten sein sollen. Es folgen die Regeln zur Darstellung und Strukturierung der Seite. Wie Sie sehen, haben die Spalten jeweils eine feste Breite. Um sie nebeneinander anzuordnen, wird hier die Eigenschaft `float` verwendet.

Um dieses Framework auf Ihre Seite anzuwenden, müssen Sie den HTML-Code entsprechend strukturieren und mit den passenden CSS-Klassen versehen, wie hier gezeigt:

```
<div class="zeile" id="kopfteil">
  <div class="spalte raster_12">
    <h1>Seitenüberschrift/Logo: 12 Spalten</h1>
  </div>
</div>
<div class="zeile" id="hauptteil">
  <div class="spalte raster_8">
    <h2>Artikelüberschrift: 8 Spalten</h2>
    <div class="zeile" id="teaserboxen">
      <div class="spalte raster_4">
        <h3>Teaserbox 1: 4 Spalten</h3>
        <section>
          <p>Hier steht der Text</p>
        </section>
      </div>
      <div class="spalte raster_4">
        <h3>Teaserbox 2: 4 Spalten</h3>
        <section>
          <p>Hier steht der Text</p>
        </section>
      </div>
    </div>
    <div class="zeile" id="artikelfussteil">
      <div class="spalte raster_8">
        <p>Artikelfußteil: 8 Spalten</p>
      </div>
    </div>
    <div class="spalte raster_4" id="seitenleiste">
      <p>Hier steht der Text</p>
    </div>
  </div>
  <div class="zeile" id="fussteil">
    <div class="spalte raster_12">
      <p>Fußteil: 12 Spalten</p>
    </div>
  </div>
</div>
```

Der obige Code erzeugt das Layout, das Sie in Abbildung 11-14 sehen.



Im Abschnitt »Siehe auch« finden Sie einen Link zur englischsprachigen Anleitung für das *1kb CSS Grid*. Hier wird auch erklärt, wie Sie mithilfe von Prozentwerten ein flexibles Raster erzeugen können, das sich dynamisch an die verfügbare Breite des Ansichtsbereichs anpasst.

Für die Erstellung des Seitenlayouts haben wir keine Zeile CSS-Code selbst schreiben müssen. Das hat das Framework für uns erledigt. Wir haben lediglich die Seitenstruktur im HTML-Code angelegt und bei der Gestaltung der einzelnen Elemente ein wenig nachgeholfen.

Diskussion

Layoutraster unterteilen die Seite in mehrere gleich breite Spalten und Zwischenräume. Am häufigsten kommen dabei Systeme mit zwölf Spalten zum Einsatz. Im Gegensatz zu den mehrspaltigen Layoutsystemen alter Schule wird dabei aber nicht jede Spalte mit Inhalt versehen. Vielmehr funktionieren die Spalten als eine Art vertikale Maßeinheit, eben als Raster.



Wir verwenden hier den Begriff »Spalte« nicht im Sinne einer Layoutspalte, wie in anderen Rezepten dieses Kapitels. Im Zusammenhang mit Grid-Systemen hat das Wort eher die Bedeutung einer Längeneinheit.

Die Anzahl der Spalten hat praktische Gründe. Da zwölf sowohl durch vier wie auch durch drei teilbar ist, können hiermit diverse verschiedener Layouts umgesetzt werden. (Es gibt allerdings auch Varianten, die mit 18 oder mehr Spalten arbeiten.)

Die Breite der Spalten und Abstände ist nicht eindeutig festgelegt. Sie orientiert sich allerdings oft an mathematisch-ästhetischen Prinzipien wie dem Goldenen Schnitt, der Drittelregel oder Fibonacci-Folgen.

Weitere Frameworks

Im Laufe der letzten Jahre wurde eine Vielzahl verschiedener Frameworks entwickelt, die die Arbeit mit Layoutrastern im Webdesign erleichtern sollen. Die Lösungen reichen von minimalistischen Systemen wie dem hier gezeigten *1kb CSS Grid* bis hin zu Komplettlösungen, die auch typografische Aspekte, Formulargestaltung und andere Bereiche des Designs mit CSS einbeziehen, wie das von Google entwickelte *Blueprint CSS*.

Andere Systeme wie *960.gs* präsentieren sich mit einem umfangreichen Satz an Templates für Layout und Bildbearbeitungsprogramme, mit denen die Erstellung von Mock-ups und deren Übertragung auf das passende CSS-Raster erleichtert wird.

Das *1140 CSS Grid* ist auf eine Bildschirmbreite von 1.280 Pixeln »optimiert«. Ist der Ansichtsbereich kleiner, passt sich das Layoutraster dynamisch an den vorhandenen Platz an. Das soll selbst für Mobiltelefone (siehe Kapitel 10) noch funktionieren.

Der Nachteil bei der Verwendung von Grid-Systemen wird von vielen Puristen darin gesehen, dass es für ihre Benutzung nötig ist, den HTML-Code auf bestimmte Weise zu strukturieren, damit eine Aufteilung in Spalten bzw. Layoutzeilen überhaupt möglich ist.

Je nachdem, ob Ihre Seiten statisch von einem Webserver ausgeliefert werden oder als Template eines Content-Management-Systems wie WordPress, TYPO3 oder Joomla! angelegt sind, sollten Sie daher überlegen, ob Ihnen eine strikte Trennung von Quellcode und Gestaltung wichtiger ist oder ob Sie lieber nach den Prinzipien des *Rapid Prototyping* arbeiten, um möglichst schnell zu einem Ergebnis zu kommen.

Siehe auch

Die Homepage des *1kb CSS Grid* von Tyler Tate unter <http://1kbgrid.com/>; eine dreiteilige Anleitung zur Verwendung des 1kb CSS Grid unter <http://www.usabilitypost.com/2009/06/06/the-1kb-css-grid-part-1/>; das *Blueprint-CSS-Framework* unter <http://blueprintcss.org/>; das *960 Grid System* unter <http://960.gs/>; das *1140 CSS Grid* unter <http://cssgrid.net/>; das Grundrezept 2.23 zur Verwendung von Floats; das Rezept 11.3 zur Erstellung von Layouts mit Floats; das Rezept 1.13 zur Definition mehrerer CSS-Klassen für das gleiche HTML-Element; das Rezept 12.13 zur Verwendung eines Grundlinienrasters; den Artikel *Five Simple Steps to Designing Grid Systems* mit einer umfassenden Abhandlung zur Arbeit mit Layoutrastern von Mark Boulton unter <http://www.markboulton.co.uk/journal/comments/five-simple-steps-to-designing-grid-systems-preface>; das Rezept 11.11 mit weiteren Alternativen zu bestehenden Grid-Systemen.

11.10 Inhalte mit CSS3 auf mehrere Spalten verteilen

Problem

Inhalte sollen gleichmäßig auf mehrere Spalten verteilt werden, wie in Abbildung 11-15 gezeigt.

Lösung

Umgeben Sie den Inhalt, der mehrspaltig angezeigt werden soll, mit einem `div`-Element. Mit den entsprechenden CSS3-Eigenschaften können Sie anschließend die Anzahl der Spalten, ihre Breite, die Abstände dazwischen und vieles mehr steuern.

```
div.spalten {
    /* Safari und Chrome */
    -webkit-column-count: 3;
    -webkit-column-gap: 2em;
    -webkit-column-rule: 0.1em solid green;

    /* Firefox und Freunde */
    -moz-column-count: 3;
    -moz-column-gap: 2em;
    -moz-column-rule: 0.1em solid green;
```

```

/* alle anderen */
column-count:      3;
column-gap:        2em;
column-rule:       0.1em solid green;
}

img {
  float:           left;
  margin-right:    1em;
}

```

Der HTML-Code könnte etwa so aussehen:

```

<div class="spalten">
  <h2>Überschrift</h2>
  <p>Lorem ipsum dolor sit amet...</p>
  <p>Phasellus vel turpis...</p>
  <h2>Noch mehr</h2>
  <p>Proin laoreet nulla...</p>
</div>

```

Das oben stehende Stylesheet erzeugt drei Spalten, deren Breite vom Browser automatisch berechnet wird. Zwischen den Spalten soll ein Abstand von 2em. Um die Spalten visuell deutlicher zu trennen, definieren wir in den Zwischenräumen außerdem eine Trennlinie. Die letzte Regel sorgt dafür, dass das Bild im dritten Absatz vom übrigen Text umflossen wird.



Abbildung 11-15: Anhand der CSS3-Eigenschaften `column-count`, `column-gap` und `column-rule` wurde ein einfaches mehrspaltiges Layout erstellt

Diskussion

Die Verwendung der CSS3-Eigenschaften für die Spaltendarstellung kann Webdesignern eine Menge Arbeit ersparen. Allerdings wird beim Schreiben dieses Buchs nur eine kleine Untermenge der Eigenschaften von aktuellen Browsern zuverlässig unterstützt. Während alle aktuellen Versionen von Firefox, Safari und Chrome mit der Spaltendarstellung umgehen können, wurde sie in Opera erst in der Version 11.10 hinzugefügt. Der Internet Explorer soll die Mehrspaltigkeit ab der Version 10 unterstützen.

Da diese Technik sehr vielversprechend ist, haben wir uns entschieden, hier trotz der noch unvollständigen Umsetzung zumindest drei der wichtigsten Eigenschaften vorzustellen.

Safari, Chrome und Firefox unterstützten die Eigenschaften, während dieses Buch geschrieben wurde, nur in der Präfixschreibweise. Deshalb müssen wir die Angaben hier mehrmals vornehmen. Die Eigenschaft `column-count` gibt an, wie viele Spalten erzeugt werden sollen, während `column-gap` den Abstand zwischen den Spalten definiert. Der Wert für `column-gap` wird in einer beliebigen in CSS gültigen Längeneinheit angegeben. Mit `column-rule` können Sie zudem eine Trennlinie zwischen den Spalten ziehen. Die Werte und das Verhalten von `column-rule` entsprechen denen der Kurzschrifteigenschaft `border`. Prozentwerte sind für `column-gap` und `column-rule` jedoch nicht erlaubt.

Die übrigen Eigenschaften

Neben den hier gezeigten drei Spalteneigenschaften definiert das *Multicol*-Modul der CSS3-Spezifikation noch eine ganze Reihe weiterer Einstellungsmöglichkeiten. Hierzu gehört besonders die Eigenschaft `column-width`, mit der Sie angeben können, wie breit die einzelnen Spalten sein sollen. Die Anzahl der Spalten wird in diesem Fall vom Browser berechnet. Außerdem soll es irgendwann möglich sein, selbst zu steuern, vor oder nach welchem Element eine neue Spalte beginnt (z.B. vor Überschriften).

Floats in den Spalten platzieren

Wenn Sie innerhalb des `div`-Elements, das die Spalten enthält, Floats (z.B. ein Bild, das vom Text umflossen werden soll) verwenden möchten, dürfen diese keine direkten Nachkommen des Spaltencontainers sein. Ansonsten kann es bei Webkit-Browsern wie Safari und Chrome zu Problemen kommen. Aus diesem Grund haben wir das Bild in unserer Lösung innerhalb des Absatzes platziert:

```
<p>Proin laoreet nulla...</p>
```

Internet Explorer

Der Internet Explorer soll die mehrspaltige Darstellung mit CSS ab der Version 10 unterstützen. Frühere Versionen können immerhin mithilfe des JavaScript-Frameworks jQuery zum Mitspielen gebracht werden (vgl. dazu den Link unter »Siehe auch«).

Siehe auch

Das CSS3-Modul für mehrspaltige Layouts unter <http://www.w3.org/TR/css3-multicol/>; die Quirksmode-Website zur mehrspaltigen Darstellung von Inhalten unter <http://www.quirksmode.org/css/multicolumn.html>; den Artikel von Adam Wulf zur Verwendung von jQuery für die Erzeugung mehrspaltiger Layouts unter <http://welcome.totheinternet.net/2008/07/22/multi-column-layout-with-css-and-jquery/>; den Webkrauts-Artikel zu mehrspaltigen Layouts mit CSS3 unter <http://www.webkrauts.de/2011/12/08/css-3-im-praxis-test-multi-column-layout/>.

11.11 Ausblick

Problem

Sie möchten wissen, welche Alternativen es zu den hier beschriebenen Layouttechniken gibt bzw. geben wird.

Lösung

Aktuell arbeitet das World Wide Web Consortium an mehreren CSS3-Modulen, die das Layout von Webseiten erleichtern sollen. Wir geben hier auszugsweise die übersetzten Beschreibungen der Module aus der W3C-Spezifikation wieder:¹

Flexible Box Layout

Das *Flexible Box Layout*-Modul, kurz »Flexbox« genannt, erweitert die Eigenschaft `display` um die Werte `flexbox` und `inline-flexbox`.

Anhand neuer Eigenschaften können die Kindelemente einer Box horizontal oder vertikal angeordnet und gleichmäßig verteilt werden. Nicht benutzter Leerraum kann einem oder mehreren Kindelementen zugewiesen werden.

Durch die Verschachtelung mehrerer als Flexbox definierter Elemente können Seitenlayouts unterschiedlicher Komplexität erstellt werden.

CSS Template Layout

Dieses Modul enthält Merkmale zur Beschreibung von Layouts auf einer abstrakten Ebene. Sie sollen bei der Positionierung von vordefinierten Seitenelementen (Widgets) in einer grafischen Benutzerschnittstelle oder einem Layoutraster helfen.

¹ Im Original können Sie das hier nachlesen: <http://www.w3.org/Consortium/Legal/IPR-FAQ-20000620#attribdoclic>.

Hierbei definiert die Schablone einen oder mehrere Bereiche (Slots). Die Inhalte können dabei auf mehrere Slots verteilt werden. Durch die Verkettung mehrerer Slots können komplexe Formen erstellt werden.

Es wird außerdem ein spezieller CSS-Selektor `::slot` definiert, mit dem Elemente abhängig von dem Slot, in dem sie sich befinden, gestaltet werden können.

Grid Positioning

Dieses Modul beschreibt die Positionierung von Elementen anhand eines Layoutrasters ähnlich der Grid-Systeme, die in diesem Buch beschrieben werden.

Grid Layout

Das *Grid Layout*-Modul ist vorwiegend für die Entwicklung von Webapplikationen gedacht und soll eine Reihe verschiedener Layouts ermöglichen. Der vorhandene Platz kann in verschiedene Bereiche (einer Applikation) unterteilt werden. Es können aber auch Beziehungen der Elemente, z.B. in Bezug auf Größe und Position, definiert werden.

Hierbei können die Elemente in einem Raster aus Spalten und Zeilen angeordnet werden. Im Gegensatz zu anderen Grid-Systemen (oder gar Tabellen) muss hierfür jedoch keine zusätzliche Struktur im HTML-Code der Seite angelegt werden. Hierdurch ist eine Vielzahl verschiedener Layouts möglich.

Die Layouttechniken sind dabei nicht von der HTML-Quelltextreihenfolge abhängig. Durch die Kombination von Media Queries (siehe Rezept 10.1) mit den CSS-Eigenschaften zur Kontrolle des Rasters (und dessen Nachfahren) kann das Layout an eine Vielzahl verschiedener Geräte, Ausrichtungen und den verfügbaren Platz angepasst werden. Hierbei ist eine Änderung der semantischen Struktur des Inhalts nicht nötig.

Diskussion

Von den hier genannten Modulen hat das Flexbox-Modul die größte Unterstützung. Es wird von Firefox, Safari, Chrome, iOS Safari und dem Android-Browser in allen aktuellen Versionen durch die jeweilige Präfixschreibweise (`-webkit-` bzw. `-moz-`) unterstützt. Hierbei ist Google Chrome aktuell der einzige Browser, der die Spezifikation ab Version 17 vollständig umsetzt.

Während Opera bisher keinerlei Unterstützung für das Flexbox-Modul bietet, sollen seine Möglichkeiten im Internet Explorer zumindest teilweise ab der Version 10 (mit dem Präfix `-ms-`) nutzbar sein. Mit dem Polyfill *flexiejs* ist es möglich, auch anderen Browsern die Fähigkeiten des Flexbox-Moduls beizubringen.

Auch wenn das Flexbox-Modul in seiner Entwicklung schon recht weit fortgeschritten ist, ist sein Einsatz auf einer Produktions-Website im Moment noch nicht empfehlenswert.

Die anderen drei Module befinden sich noch in einem sehr frühen Entwicklungsstadium. Die Browserunterstützung ist entweder dürftig oder (noch) nicht vorhanden.

Siehe auch

Einen Artikel des Smashing Magazine zum Flexbox-Modul unter <http://coding.smashing-magazine.com/2011/09/19/css3-flexible-box-layout-explained/>; die Website für das Flexbox-Polyfill *flexiejs* unter <http://flexiejs.com>; die Spezifikationen für die hier genannten Module unter <http://www.w3.org/TR/css3-flexbox/> (Flexbox), <http://www.w3.org/TR/css3-layout> (Template Layout), <http://www.w3.org/TR/css3-grid/> (Grid Positioning) und <http://www.w3.org/TR/css3-grid-layout/> (Grid Layout).

Einführung

Obwohl Webworker viel Zeit damit verbringen, Browserprobleme zu lösen und sich die neuesten Tricks anzueignen, sollten wir nicht vergessen, dass wir in erster Linie Designer sind. Stylesheets helfen Ihnen dabei, Ihre Designideen Realität werden zu lassen. Gutes Design kann Sie dabei unterstützen, die Aufmerksamkeit der Benutzer Ihrer Website zu gewinnen und zu halten.

Auch wenn das Verständnis der Syntax und die Kenntnis der Spezifikationen wichtig sind, sollten Sie bedenken, dass der CSS-Code immer nur das Design implementiert. Im Kern kann man CSS als Visualisierungssprache verstehen. Daher ist es wichtig, dass Sie zumindest die Grundlagen kennen, mit denen Designprinzipien in CSS umgesetzt werden können.

In diesem Kapitel finden Sie eine Reihe von Techniken, mit denen Sie das Erscheinungsbild Ihrer Website auf verschiedenste Weise anpassen können. Hierbei behandeln wir Basistechniken wie die Definition von Rollover-Effekten und die Wahl eines Farbschemas genauso wie die angesagten CSS3-Techniken zur Definition von abgerundeten Ecken, Farbverläufen und Schattenwürfen.

Außerdem beschäftigen wir uns mit der Erstellung einfacher Animationen mit CSS3, der Erstellung modularer Skalen, der Definition eines Grundlinienrasters und der Erstellung von quasizufälligen Hintergrundmustern.

12.1 Rollover-Effekte ohne Bilder erzeugen

Problem

Sie möchten Ihre Webseiten mit Rollover-Effekte versehen, ohne hierfür JavaScript einsetzen zu müssen.

Lösung

Verwenden Sie die Pseudoklasse `:hover`, um beim Überfahren eines Elements seine Darstellung zu ändern. Am häufigsten werden Rollover-Effekte für Links definiert (siehe Abbildung 12-1).

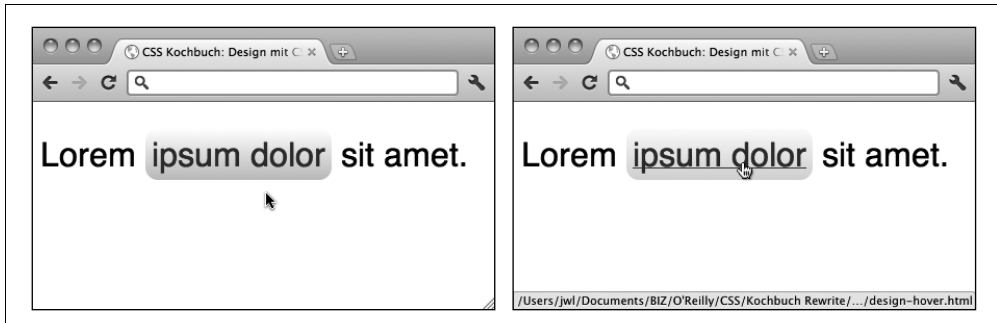


Abbildung 12-1: Ein Link mit Rollover-Effekt

```
a:link {
    text-decoration: none;
    color:          blue;
    background:     #eee;
}

a:hover {
    text-decoration: underline;
    background:     rgb(254,252,234);
}
```

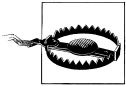
Hier haben wir zunächst die standardmäßige Linkunterstreichung entfernt. Damit die Links trotzdem vom übrigen Text unterschieden werden können, werden sie in blauer Schrift vor einem hellgrauen Hintergrund dargestellt.



Wenn Sie die Standarddarstellung (speziell die Unterstreichung) von Links ändern, sollten Sie auf jeden Fall dafür sorgen, dass sie sich deutlich vom übrigen Text unterscheiden, damit die Benutzbarkeit erhalten bleibt.

Die zweite Regel legt fest, wie der Link dargestellt werden soll, wenn sich der Mauszeiger darüber befindet. Der Link soll unterstrichen und mit einem hellgelben Hintergrund versehen werden.

Prinzipiell sind Ihnen bei der Gestaltung kaum Grenzen gesetzt. Für die Rollover-Effekte von Links können Sie sämtliche Eigenschaften verwenden, die für Inline-Elemente gültig sind. Für unsere Abbildung haben wir den Link beispielsweise mit einem Farbverlauf und abgerundeten Ecken versehen.



Auf den meisten Mobilgeräten mit berührungsempfindlichen Bildschirmen funktioniert die Pseudoklasse `:hover` nicht oder nur eingeschränkt.

Rollover-Effekte sind aber nicht auf Inline-Elemente oder Links beschränkt. Es ist durchaus denk- und auch machbar, beispielsweise ein bestimmtes `div`-Element hervorzuheben, wenn sich der Mauszeiger darüber befindet:

```
div.rollover {  
    opacity: 0.5;  
}  
  
div.rollover:hover {  
    opacity: 1;  
}
```

Hier werden mit der Klasse `.rollover` markierte `div`-Elemente halb transparent dargestellt. Befindet sich der Mauszeiger über dem Element, wird das Element hervorgehoben, indem der Wert für `opacity` auf den Wert 1 (lichtundurchlässig) gesetzt wird.

Diskussion

Rollover-Effekte gehören mittlerweile zum Standardrepertoire. War es noch vor wenigen Jahren nötig, den Effekt per JavaScript über das `onmouseover`-Event zu erzeugen, reicht es für die heute aktuellen Browser aus, einen speziellen Stil mit dem Pseudoelement `:hover` zu schreiben, wie oben gezeigt.

Wie in der Lösung bereits gesagt, können Sie im Prinzip alle Eigenschaften verwenden, die für die jeweilige Darstellungsrolle (Inline, Block usw.) gültig sind. Gerade Navigationslinks werden gern als Blockelemente definiert, um so den anklickbaren Bereich zu vergrößern. Ein angenehmer Nebeneffekt ist, dass Sie dadurch auch Eigenschaften wie `width`, `height`, `margin` usw. für die Gestaltung heranziehen können.

Theoretisch könnten Sie die Elemente also bis zur Unkenntlichkeit verändern. Seien Sie kreativ, aber denken Sie immer daran, dass der Effekt meistens einem bestimmten Zweck dient: die Usability einer Seite zu erhöhen.



Der Internet Explorer 7 unterstützt `:hover` für andere Elemente als Links nur, wenn er sich im *Standards*-Modus befindet. Hierfür muss das Dokument den richtigen Dokumenttyp (siehe Rezept 1.22) besitzen.

Siehe auch

Das Rezept 6.1 zur Definition von Stilen für verschiedene Linkzustände; das Rezept 1.22 zur Dokumenttyp-Deklaration; eine Demonstration der Fähigkeiten von HTML5 und CSS3 bei der Gestaltung von Rollover-Effekten unter <http://codepo8.github.com/3drollovers.css/>.

12.2 Ein Farbschema für eine Website erstellen

Problem

Sie möchten Ihre Website mit einem durchgängigen Farbschema versehen.

Lösung

Verwenden Sie eines der vielen Onlinewerkzeuge für die Auswahl eines passenden Farbschemas für Ihre Website. In Abbildung 12-2 sehen Sie das Prinzip am Beispiel der Website *colorschemedesigner.com*.

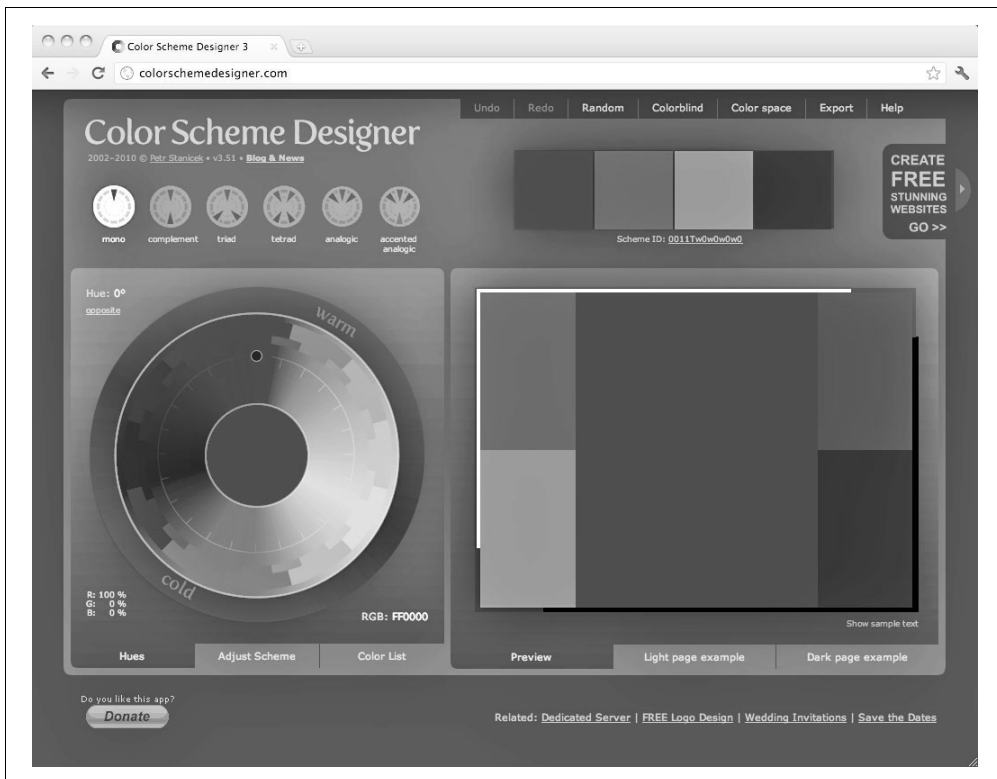


Abbildung 12-2: Ein Onlinegenerator zum Erzeugen von Farbschemata

In den meisten Fällen wählen Sie zunächst eine Grundfarbe. Darauf basierend werden dann mehr oder weniger automatisch dazu passende Farben ausgewählt.

Das Auswahlprinzip lässt sich ändern. In unserem Beispiel gibt es beispielsweise sechs verschiedene Möglichkeiten, die Farben zu kombinieren:

- *mono* – monochrom, es werden verschiedene Schattierungen der gleichen Farbe verwendet. Dieses Schema lebt hauptsächlich vom Hell-dunkel-Kontrast. Den Charakter erhält die Seite durch die gewählte Grundfarbe (z.B. Gelb = hell, warm, Blau = dunkel, kühl).
- *complement* – Komplementärfarben, es werden Schattierungen der gewählten Farbe und deren Gegenteil, z.B. Rot und Grün, benutzt. Hier entsteht der Kontrast weniger durch verschiedene Helligkeitswerte als mehr durch gegensätzliche Farben. Komplementärfarben assoziieren einerseits Gegensätze, gleichzeitig aber auch Harmonie, da sich die beiden Farben im Gleichgewicht zueinander befinden.
- *triad* – es werden drei Farben verwendet, die sich im Farbkreis möglichst weit voneinander entfernt befinden. Wird auch als »Dreiklang« bezeichnet. Ähnlich wie in der Musik ergänzen sich drei Farben zu einem harmonischen Ganzen.
- *tetrad* – es werden vier Farben verwendet, die sich im Farbkreis möglichst weit voneinander entfernt befinden. Ähnlich wie beim »Dreiklang« werden hier vier Farben zu einem »Vierklang« harmonisch ergänzt.
- *analogic* – es werden drei Farben benutzt, die im Farbkreis nur eine geringe Entfernung zueinander haben. Durch die Nähe der Farben zueinander wird der Eindruck von Harmonie und Konsistenz vermittelt. Hierbei beeinflusst die gewählte Grundfarbe den Eindruck besonders stark.
- *accented analogic* – wie *analogic*, aber mit einer zusätzlichen Komplementärfarbe zur gewählten Farbe. Ergänzt die Harmonie der analogen Farben um die Spannung der Komplementärfarben.

Unterhalb der Optionen für das Auswahlprinzip auf der linken Seite können Sie weitere Einstellungen vornehmen: Mit *Hues* wählen Sie die gewünschte Grundfarbe, unter *Adjust Theme* können Sie Dinge wie Sättigung/Helligkeit (*Theme Saturation/Brightness*) und Kontrast (*Scheme Contrast*) des Farbschemas einstellen, ein Klick auf *Color List* zeigt eine Liste der Farben in hexadezimaler Notation an.

Auf der rechten Seite wird im oberen Bereich eine Übersicht des Schemas eingeblendet. Darunter sehen Sie ein Beispiel für ein mögliches Layout unter Verwendung der gewählten Farben.

Ein wenig versteckt rechts unterhalb des Beispiellayouts gibt es außerdem einen Link, mit dem Sie Beispieltext aktivieren und wieder abschalten können.

Darunter befinden sich die Schalter, mit denen Sie die Seitenvorschau steuern können. Hier haben Sie die Option der Normalansicht (*Preview*) sowie einer helleren (*Light page example*) und einer dunkleren Variante (*Dark page example*).

Der Color Scheme Generator ist natürlich nicht das einzige Werkzeug zum Erstellen von Farbschemata und soll hier nur als Beispiel für die Funktionsweise dienen. Links zu weiteren ähnlichen Werkzeugen finden Sie im Abschnitt »Siehe auch«.

Diskussion

Die Auswahl der richtigen Farbpalette kann einen enormen Einfluss auf die Wirkung Ihrer Website haben. Soll es eher frühlingshaft bunt zugehen oder eher mystisch düster? Sie haben die Wahl.

Ähnlich wie bei der Auswahl der richtigen Schriften sollten Sie auch bei der Erstellung eines Farbschemas vorsichtig vorgehen. Zu viele verschiedene Grundfarben können leicht verwirren. Zwei bis drei Grundfarben, die nach Möglichkeit miteinander verwandt sind, reichen meistens aus.

Bevor Sie ein Farbschema für Ihre Website definieren, sollten Sie sich über die Aussage der Seiten Gedanken machen. An wen richtet sich Ihr Angebot? Es ist eher nicht sinnvoll, wenn ein Spielzeughersteller seine Seiten im Gothic-Stil mit viel Schwarz gestaltet¹ oder eine Anwaltskanzlei sich mit knalligen Bonbonfarben präsentiert.



Unter Umständen kann es hilfreich sein, mit einem CSS-Präprozessor zu arbeiten, um die Farbwerte nur einmal im Stylesheet ändern zu müssen.

Wenn Ihnen die Zugänglichkeit Ihrer Website wichtig ist, sollten Sie darauf achten, dass die Textfarben einen ausreichenden Kontrast zu den gewählten Hintergründen haben. Ansonsten kann es bei Menschen mit Einschränkungen passieren, dass sie Text und Hintergrund nicht mehr unterscheiden können. Mehr zu diesem Thema finden Sie im folgenden Rezept 12.3.

Siehe auch

Den *Color Scheme Designer* unter <http://colorschemedesigner.com/>; den Adobe-Farbwähler *Kuler* unter <http://kuler.adobe.com/>; eine Liste mit weiteren Onlinewerkzeugen zur Arbeit mit Farben und Farbschemata unter <http://www.cssjuice.com/25-popular-color-scheme-and-palette-generators/>; das Rezept 12.3 zum Thema Farbkontrast und Zugänglichkeit; das Rezept 13.17 zur Benutzung von CSS-Präprozessoren; den Wikipedia-Artikel zur Farbenlehre und zu diversen weiterführenden Ressourcen unter <http://de.wikipedia.org/wiki/Farbenlehre>.

¹ Obwohl es hier vermutlich auch auf das Sortiment ankommt.

12.3 Für ausreichenden Farbkontrast sorgen

Problem

Ihre Website soll auch für Benutzer zugänglich sein, die bestimmte Farben nicht oder nur schwer wahrnehmen können.

Lösung

Versehen Sie die inhaltlich wichtigen Dinge mit ausreichendem Kontrast. Verwenden Sie für Text immer die Eigenschaften `color` (Vordergrundfarbe, Schrift und Tabellen) und `background-color` (Hintergrundfarbe) gemeinsam. Das Kontrastverhältnis (der Unterschied der relativen Helligkeit) zwischen Text und Hintergrund sollte für normalen Text gemäß den *Web Content Accessibility Guidelines* (WCAG) mindestens 4,5:1 betragen.

Bei großem Text (ab einer Schriftgröße von 18 Punkt) reicht ein Kontrastverhältnis von 3:1. Ein Beispiel sehen Sie in Abbildung 12-3. Im Abschnitt »Siehe auch« finden Sie eine Auswahl von Werkzeugen, mit denen Sie das Kontrastverhältnis testen können.

Ein Beispiel:

```
p {  
  color:          #6a6a6a;  
  background-color: #fff;  
}
```

Das Kontrastverhältnis liegt bei 5,4:1:

```
h1 {  
  font-size:      1.8em;  
  color:          #8a8a8a;  
  background-color: #fff;  
}
```

Hier ist das Kontrastverhältnis geringer. Es liegt gerade noch bei 3,5:1. Das ist bei einer Schriftgröße von 1.8em zwar immer noch ausreichend, wäre für normalen Fließtext aber nicht mehr genug.



Wir haben für die Berechnung der Kontrastverhältnisse den *Webaim Color Contrast Checker* verwendet. Einen Link finden Sie im Abschnitt »Siehe auch«.

Diskussion

Ungefähr 8 bis 10% der männlichen Bevölkerung sind von der Rot-Grün-Schwäche oder -Blindheit betroffen. Bei den Frauen sind es 0,8%. Diese Menschen können Rot- und Grüntöne mit ähnlicher oder gleicher Helligkeit kaum unterscheiden. Das kann beim Lesen, aber auch beim Erkennen grafischer Elemente zu Unterscheidungsschwierigkeiten führen.

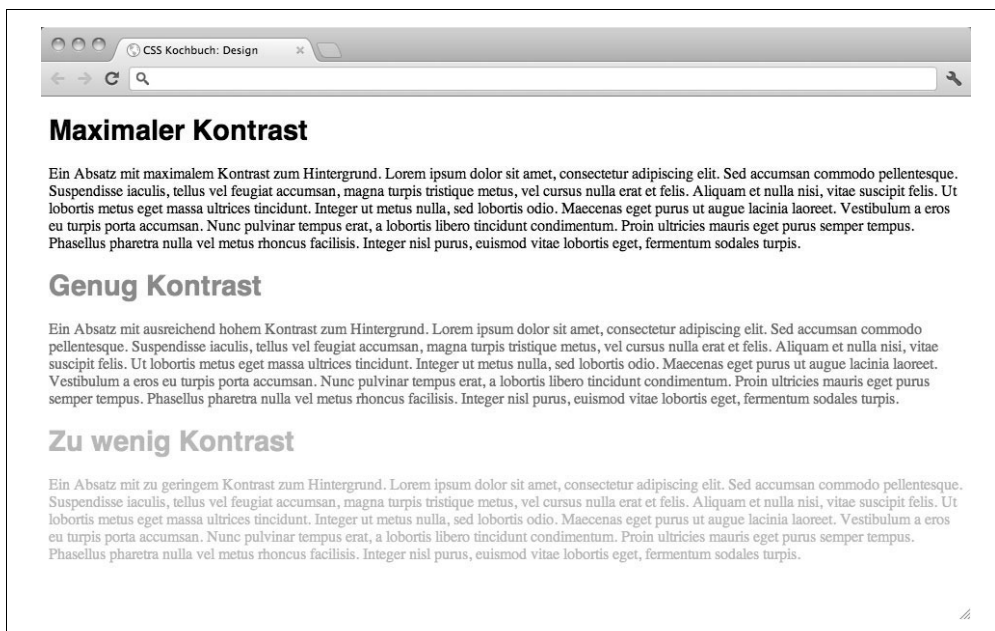


Abbildung 12-3: Überschriften und Absätze in verschiedenen Kontrastverhältnissen zum Hintergrund

Auch die Auswahl mehr oder weniger komplementärer Farben bringt nicht immer das gewünschte Ergebnis. Ausschlaggebend ist die unterschiedliche Helligkeit der Farben.

Laut den Bestimmungen der WCAG (Web Content Accessibility Guidelines) sollte der Kontrast mindestens ein Verhältnis von 4,5:1 haben. Zum Glück gibt es eine Reihe von Werkzeugen (Details im Abschnitt »Siehe auch«), die Ihnen die Berechnung des Kontrastverhältnisses abnehmen. Die genaue Berechnungsmethode wird auf der Website des W3C beschrieben.

Siehe auch

Die Wikipedia-Seite zum Thema Farbenblindheit unter <http://de.wikipedia.org/wiki/Farbenblindheit>; die Visicheck-Website, auf der Sie sehen können, wie Ihre Website für Menschen mit farblichen Einschränkungen dargestellt wird, unter <http://www.visicheck.com/>; die Richtlinien für barrierefreie Webinhalte des W3C unter <http://www.w3.org/Translations/WCAG20-de/>; die Erläuterungen des W3C zur Berechnung des Kontrastverhältnisses unter <http://www.w3.org/TR/WCAG20/#relativeluminancedef>; ein Werkzeug zum Überprüfen des Kontrastverhältnisses zweier hexadezimaler Farbwerte unter <http://webaim.org/resources/contrastchecker/>; ein Firefox-Add-on zur Berechnung des Farbkontrasts unter <https://addons.mozilla.org/de/firefox/addon/wcag-contrast-checker/>.

12.4 Farbverläufe mit CSS3 definieren

Problem

Sie möchten Farbverläufe für das Design Ihrer Website einsetzen. Dabei soll nach Möglichkeit auf die Verwendung von Grafiken verzichtet werden.

Lösung

Definieren Sie den Wert der Eigenschaft `background-image` anhand der neuen Farbverlaufs-funktionen aus dem CSS3-Modul *Image Values and Replaced Content*. Um den in Abbildung 12-4 gezeigten vertikalen Farbverlauf zu erzeugen, haben wir folgenden Code verwendet:

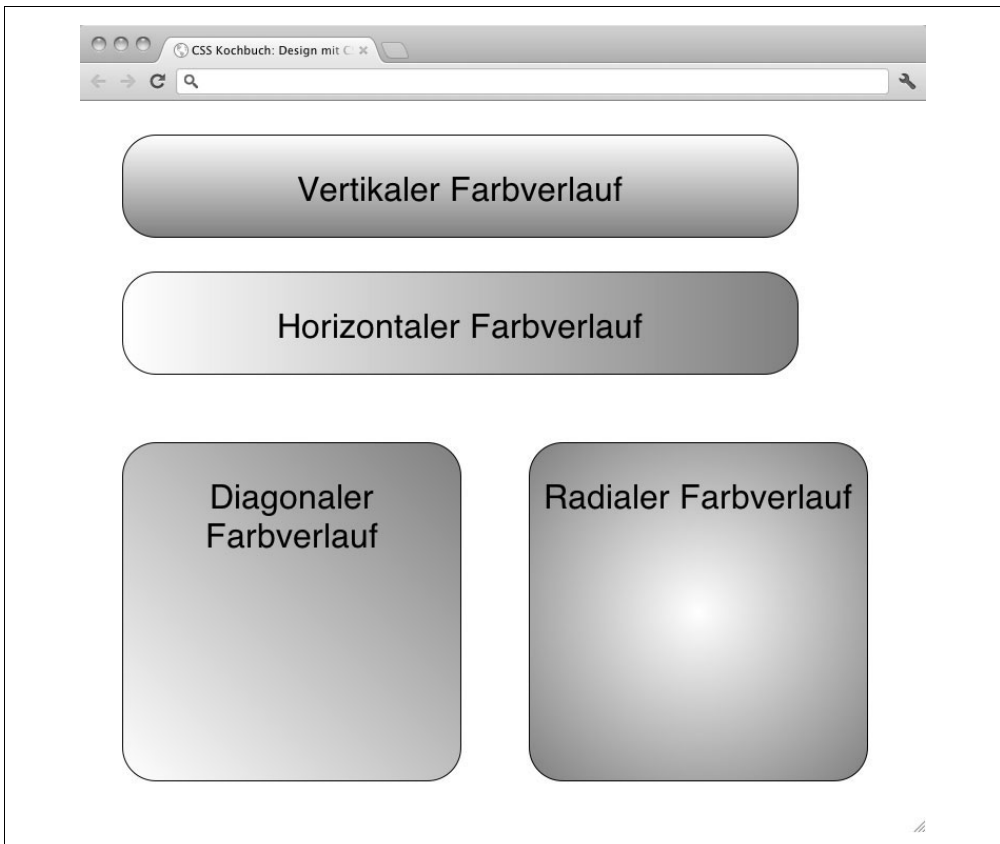


Abbildung 12-4: Farbverläufe mit CSS

```
.farbverlauf_vertikal {
  background: #fefcea;
  background: -moz-linear-gradient(top,#fefcea 0%, #f1da36 100%);
  background: -webkit-gradient(linear, left top, left bottom,
  color-stop(0%,#fefcea), color-stop(100%,#f1da36));
  background: -webkit-linear-gradient(top, #fefcea 0%,#f1da36 100%);
  background: -o-linear-gradient(top, #fefcea 0%,#f1da36 100%);
  background: -ms-linear-gradient(top, #fefcea 0%,#f1da36 100%);
  background: linear-gradient(to bottom, #fefcea 0%,#f1da36 100%);
  /* W3C (Stand: 12.1.2012) */
}
```

Die Regel beginnt mit der Definition einer einfachen Hintergrundfarbe für Browser, die keine CSS-Farbverläufe darstellen können. Es folgen die Deklarationen in der jeweiligen Präfixschreibweise für aktuelle Browser. Um auch ältere Browser aus der Webkit-Familie zu unterstützen, haben wir hier die ältere und die aktuelle Präfixschreibweise verwendet.

Die Regel endet mit der Deklaration gemäß der Spezifikation des W3C. Wie Sie sehen, unterscheidet sich die Schreibweise von den Präfixvarianten. Anstelle von `top` steht hier `to bottom`. Näheres hierzu finden Sie in der Diskussion.

Diskussion

Lange Zeit war es üblich, Farbverläufe in Webseiten über eingebundene Hintergrundbilder umzusetzen. Inzwischen verstehen aber fast alle aktuellen Browser außer Internet Explorer (bis zur Version 9) die eine oder andere Form der CSS3-Schreibweise.

Der Preis ist eine Reihe redundanter Deklarationen, die sich zum Teil sogar zu widersprechen scheinen, weil die Spezifikation noch sehr stark in Bewegung ist.² Falls Ihnen das zu aufwendig erscheint, können Sie natürlich auch weiterhin eine passende Grafik verwenden. Auf der anderen Seite sind gerade im mobilen Web ein paar Zeilen zusätzlicher CSS-Code oft leichter zu verschmerzen als zusätzliche HTTP-Anfragen.

Nach der Spezifikation können CSS3-Farbverläufe mit allen Eigenschaften verwendet werden, für die eine Grafik als Wert gültig ist, etwa bei der Eigenschaft `background-image` (bzw. der Kurzschrifteigenschaft `background`).



Wir verwenden hier absichtlich nicht die offizielle W3C-Schreibweise, weil diese aktuell noch diverse Änderungen unterworfen ist, während die hier gezeigte Präfixversion bereits jetzt von vielen Browsern verstanden wird. (In Zukunft werden Sie Ihre Regeln um die neue Schreibweise erweitern müssen.)

² Während der Arbeit an diesem Kapitel wurde beispielsweise festgelegt, dass der Winkel für radiale Farbverläufe im Uhrzeigersinn angegeben werden soll. Nach der bisherigen Version der Spezifikation wurde der Winkel entgegen dem Uhrzeigersinn definiert.

Generell wird zwischen linearen und radialen Farbverläufen unterschieden, die jeweils mit den Funktionen `linear-gradient()` bzw. `radial-gradient()` definiert werden. Hierbei fließt der Verlauf von einer Farbe zur nächsten.

Zusätzlich gibt es auch noch sich wiederholende Farbverläufe, die mit den Eigenschaften `repeating-linear-gradient()` bzw. `repeating-radial-gradient()` definiert werden.

Im Folgenden finden Sie eine Beschreibung der einzelnen Eigenschaften und ihrer Funktionsweise. Da die Möglichkeiten äußerst umfangreich sind und die Spezifikation noch nicht finalisiert ist, werden wir uns auf die Grundfunktionen beschränken. Bei Bedarf finden Sie im Abschnitt »Siehe auch« weitere Ressourcen zum Thema.

Lineare Farbverläufe

Lineare Farbverläufe haben wir bereits in der Lösung vorgestellt. In ihrer einfachsten Form erstrecken sie sich von einer Seite der umgebenden Box zur anderen (z.B. von links nach rechts oder von oben nach unten). Der Verlauf beginnt an der Oberkante und endet an ihrer Unterkante. In diesem Fall reicht es aus, die Start- und die Endfarbe des Verlaufs anzugeben, z.B.:

```
background: linear-gradient(yellow, orange);
```

Bei der Definition der Farben können Sie alle in CSS gültigen Angaben verwenden und auch miteinander kombinieren. Mit der `rgba()`-Notation sind also auch Verläufe mit einem transparenten Alphakanal möglich. Wenn Sie mehr als zwei Farben angeben, werden diese ohne weitere Angaben gleichmäßig auf den verfügbaren Platz verteilt.



Der Einfachheit halber haben wir bei diesen Beispielen auf die Präfixschreibweise verzichtet.

Die Richtung des Farbverlaufs können Sie angeben, indem Sie eines der Schlüsselwörter `top`, `right`, `bottom` oder `left` verwenden, wie hier gezeigt:

```
background: linear-gradient(bottom, yellow, orange);
```

Hier wurde die Richtung des Verlaufs umgekehrt, er verläuft jetzt von unten nach oben. Mit den Schlüsselwörtern ist es auch kein Problem, einen horizontalen Farbverlauf zu definieren:

```
background: linear-gradient(left, yellow, orange);
```

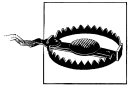
In diesem Beispiel beginnt der Verlauf auf der linken Seite in Gelb und endet rechts in Blau.

Um diagonale Farbverläufe zu erstellen, können Sie die Schlüsselwörter auch miteinander kombinieren, wie hier gezeigt:

```
background: linear-gradient(left top, yellow, orange);
```

Wollen Sie den Winkel selbst bestimmen, geben Sie ihn in Grad (deg) an.

```
background: linear-gradient(225deg, yellow, orange);
```



Wie bereits erwähnt, ist das Modul *Image Values and Replaced Content* der CSS3-Spezifikation noch im Fluss. Die hier beschriebenen Vorgehensweisen bilden ab, wie die Browser die Verlaufeigenschaften beim Entstellen dieses Buchs umsetzen (Stand: Februar 2012).

In Zukunft sollen die Schlüsselwörter die Zielrichtung angeben. Hierfür wird ihnen die Präposition *to* vorangestellt (z.B. `linear-gradient(to top, yellow, orange)`). Außerdem sollen die Winkelangaben nun im Uhrzeigersinn größer (0deg entspricht top, 90deg entspricht right, usw.) werden.

Entlang des Farbverlaufs können Sie sogenannte Farbstopps (Color Stops) angeben. So erstreckt sich der folgende Verlauf von der Oberkante des Elements bis zur Mitte:

```
background: linear-gradient(yellow 0%, orange 50%);
```

Farbstopps können in den gängigen CSS-Maßeinheiten angegeben werden. Bei einem Element mit einer Höhe von 3 em-Einheiten könnten Sie das obige Beispiel also auch so formulieren:

```
background: linear-gradient(yellow 0em, orange 1.5em);
```

Hierbei gibt es für die Anzahl der Farbstopps keine Beschränkungen. So sind auch komplexe Farbverläufe möglich, zum Beispiel:

```
background: linear-gradient(top, #4c4c4c 0%, #595959 12%, #666666 25%, #474747 39%, #2c2c2c 50%, #000000 51%, #111111 60%, #2b2b2b 76%, #1c1c1c 91%, #131313 100%);
```

Werden CSS3-Farbverläufe als Hintergrundbilder eingesetzt, können sie, wie in Rezept 2.12 beschrieben wird, kombiniert werden. Dadurch sind teilweise recht komplexe Muster und Anwendungen möglich. Beispiele finden Sie im Abschnitt »Siehe auch«.

Radiale Farbverläufe

Im Gegensatz zu linearen Verläufen gehen radiale Farbverläufe von einem bestimmten Punkt aus und erstrecken sich von dort aus in alle Richtungen. Zur Definition von radialen Farbverläufen gibt es die Funktion `radial-gradient()`, die in ihrer einfachsten Form wie `linear-gradient()` mit zwei Farbwerten auskommt:

```
background: radial-gradient(white, black);
```

Hiermit erzeugen Sie einen radialen Farbverlauf, der genau in der Mitte des Elements beginnt. Die angegebenen Farben werden gleichmäßig auf den verfügbaren Platz verteilt. Wie bei linearen Farbverläufen können Sie hier ebenfalls mehr als zwei Farben angeben. Auch in diesem Fall werden die Farben anteilig auf den verfügbaren Bereich verteilt.

Anhand der Werte für `background-position` können Sie einen anderen Mittelpunkt angeben, wie hier:

```
background: radial-gradient(right center, white, black);
```

oder auch:

```
background: radial-gradient(100% 50%, white, black);
```

Beide Deklarationen positionieren den Mittelpunkt des Verlaufs vertikal zentriert an der rechten Seite.

Ohne weiteren Angaben ergibt sich bei einem quadratischen Element (`height = width`) ein kreisförmiger Verlauf, bei rechteckigen Elementen entsteht dagegen eine elliptische Form. Die Form des Verlaufs können Sie aber auch explizit bestimmen, indem Sie eines der Schlüsselwörter `circle` oder `ellipse` angeben. Um auch bei rechteckigen Elementen einen kreisförmigen Verlauf zu erzeugen, können Sie schreiben:

```
background: radial-gradient(circle,yellow,orange);
```

Zusätzlich gibt es die Möglichkeit, anhand der Schlüsselwörter `closest-side`, `farthest-side`, `closest-corner` und `farthest-corner` anzugeben, bis wohin sich der Verlauf erstrecken soll. Hier ein Beispiel:

```
background: radial-gradient(2em 2em, circle closest-corner, white, black);
```

Diese Deklaration positioniert den Mittelpunkt des Verlaufs jeweils 2em von der linken oberen Ecke seines Elements entfernt. Er soll als Kreis dargestellt werden, bei dem sich der Farbstopps zu 100% in der vom Mittelpunkt nächstgelegenen Ecke befindet.

Die Definition von Farbstopps entspricht der von linearen Farbverläufen.

Sich wiederholende Farbverläufe

Mit sich wiederholenden Farbverläufen können Sie verschiedenartige Muster erzeugen. Analog zu linearen und radialen Verläufen gibt es hierfür die Funktionen `repeating-linear-gradient()` und `repeating-radial-gradient()`. Mit der folgenden Deklaration können Sie beispielsweise ein schräg gestreiftes Hintergrundmuster erzeugen:

```
background: repeating-linear-gradient(-45deg, red, red 5%, white 5%, white 10%);
```

Die Definition radial sich wiederholender Farbverläufe funktioniert analog zur Funktion `radial-gradient()`.

Die Spezifikation definiert noch eine Reihe weiterer Optionen. Diese sind aber noch sehr stark in Bewegung. Weitere Informationen zu den Anwendungsmöglichkeiten und zum Entwicklungsstand finden Sie im Abschnitt »Siehe auch«.

Browserunterstützung

Farbverläufe werden von den meisten aktuellen Browsern in der Präfixschreibweise unterstützt. Folgende Browser können mit CSS3-Farbverläufen umgehen: Firefox 3.6+, Safari 5.1 (4.0 und 5.0 eingeschränkt), Opera 11.6 (11.1 und 11.5 eingeschränkt), iOS Safari 5.0 (3.2 bis 4.3 eingeschränkt), Android Browser 4.0+ (2.1 bis 3.0 eingeschränkt). Die Einschränkungen bei Webkit-Browsern (Chrome, Safari, iOS Safari, Android Browser) beziehen sich auf doppelte Präfixschreibweise (siehe Lösung), Opera unterstützt in der eingeschränkten Version nur lineare Farbverläufe.

Im Internet Explorer werden Farbverläufe ab der Version 10 in der Präfixschreibweise unterstützt. Für Internet Explorer 9 und Opera 9 besteht die Möglichkeit, Verläufe als

SVG-Datei in einen Daten-URI (siehe Rezept 13.20) einzubinden. In früheren Versionen des Internet Explorer können einfache lineare Farbverläufe über die proprietäre `filter()`-Schreibweise definiert werden.

Nicht zuletzt wegen der vielen Präfixvarianten und der Eigenheiten des IE sollten Sie für Erstellung der Verlaufsdeklarationen nach Möglichkeit eines der Onlinewerkzeuge wie beispielsweise *Colorzilla* verwenden.

Siehe auch

Die CSS3-Spezifikation zu Farbverläufen unter <http://www.w3.org/TR/css3-images/#gradients>; das Onlinewerkzeug *Colorzilla* zum Erzeugen des für Farbläufe nötigen Codes unter <http://www.colorzilla.com/gradient-editor/>; einen alternativen Verlaufsgenerator unter <http://www.display-inline.fr/projects/css-gradient>; das Rezept 13.19 zur Verwendung von Präfixeigenschaften; das Mozilla Developer Network zu Farbverläufen unter https://developer.mozilla.org/de/Farbverläufe_in_CSS; die entsprechende Seite im Webkit-Blog unter <http://www.webkit.org/blog/1424/css3-gradients/>; eine Galerie verschiedener mit CSS3-Farbverläufen erzeugter Hintergrundmuster unter <http://lea.verou.me/css3patterns>.

12.5 Transparenzen erzeugen

Problem

Sie möchten bestimmte Elemente mit Transparenzen versehen. Nach Möglichkeit soll auf den Einsatz von Grafiken verzichtet werden.

Lösung 1

Verwenden Sie die CSS3-Eigenschaft `opacity`, um zu steuern, wie durchscheinend ein Element sein soll, zum Beispiel:

```
#navigation a:link { opacity: 0.5; }  
#navigation a:hover { opacity: 1; }
```

Zunächst werden Navigationslinks halb durchscheinend angezeigt. Sobald sich aber der Mauszeiger darüber befindet, wird die Transparenz verringert (der Wert von `opacity` erhöht) und der Link hervorgehoben. Das Ergebnis sehen Sie in Abbildung 12-5.

Lösung 2

Verwenden Sie die `rgba()`- oder `hsla()`-Farbnotation. Um beispielsweise ein Element mit einem halb transparenten Hintergrund zu versehen, ohne die enthaltenen Dinge (Schrift, aber auch Bilder usw.) zu beeinflussen, können Sie schreiben:

```
background-color: rgba(241,218,54,0.6);
```

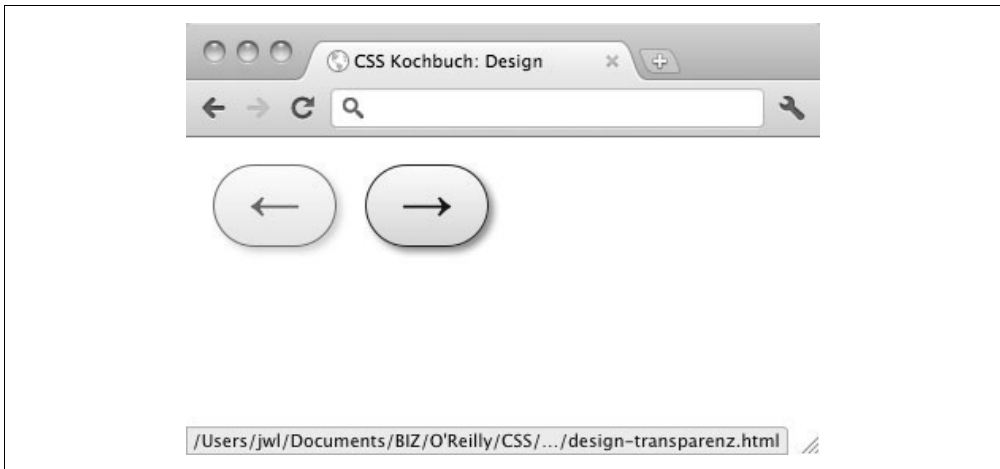


Abbildung 12-5: Durch einen passenden Wert für die Eigenschaft *opacity* wird der Navigationsbutton hervorgehoben, wenn sich der Cursor darüber befindet

Das Gleiche in der *hsla()*-Schreibweise sieht so aus:

```
background-color: hsla(53,87%,58%,0.6);
```

Durch diese Anweisung wird der Hintergrund der Überschrift in Abbildung 12-6 teilweise transparent. Auf die Schrift hat dies jedoch keinen Einfluss.



Abbildung 12-6: Eine Überschrift wurde mit einem teilweise durchscheinenden Hintergrund versehen

Die ersten drei Werte definieren dabei die Farbe. Der vierte Werte definiert einen sogenannten *Alphakanal*, der die Transparenz angibt. Die Schreibweise für den Alphakanal entspricht dem Wert für die Eigenschaft *opacity*.

Lösung 3

Verwenden Sie das Schlüsselwort `transparent` als Farbwert. Soll die Farbe eines Elements vollkommen transparent dargestellt werden, ist es nicht nötig, mit `opacity` oder `rgba()` zu hantieren. Stattdessen reicht es aus, anstelle einer Farbe das Schlüsselwort `transparent` zu verwenden, z.B.:

```
background-color: transparent;
```

Hiermit erzeugen Sie einen genauso vollkommen durchsichtigen Hintergrund für das Element, als hätten Sie die Hintergrundfarbe mit `rgba(0,0,0,0)` angegeben.

Das Schlüsselwort `transparent` wurde bereits in der ersten Version der CSS-Spezifikation definiert und wird von allen aktuellen Browsern unterstützt.

Lösung 4

Wenn alle Stricke reißen, können Sie auch transparente PNG-Dateien einsetzen, um den gewünschten Transparenzeffekt zu erzeugen. Hierbei erstellen Sie mit einem Bildbearbeitungsprogramm Ihrer Wahl eine Grafik mit der gewünschten Transparenz und sichern diese als PNG-Datei. Danach können Sie die Grafik als Hintergrundbild in Ihr Stylesheet einbinden, zum Beispiel:

```
background-image: url('bilder/halbtransparent.png');
```

Die Grafik muss nicht einmal sonderlich groß sein, weil sie standardmäßig horizontal und vertikal wiederholt wird.

Diskussion

Mit der Finalisierung des *Color*-Moduls der CSS3-Spezifikation ist es nun auch offiziell möglich, die Transparenz von Elementen und ihren Hintergründen ohne PNG-Grafiken umzusetzen. Die Browserunterstützung ist sehr hoch. Nur für den Internet Explorer in den Versionen 7 und 8 wird noch eine Fallback-Lösung gebraucht.

opacity

Anhand der Eigenschaft `opacity` können Sie festlegen, wie durchscheinend ein Element sein soll. Die Werte werden als Fließkommazahl zwischen 0 (`transparent`) und 1 (`opak`, nicht durchscheinend) angegeben. Der Standardwert ist 1.

Der Wert von `opacity` wirkt sich auf das gesamte Element aus, nicht nur auf den Hintergrund. Dies ist nicht das Gleiche wie die Vererbung eines Werts. Technisch gesehen, wird das Element zunächst außerhalb des sichtbaren Bereichs inklusive aller enthaltenen Elemente aufgebaut. Erst danach wird die Eigenschaft `opacity` angewendet und das Element dargestellt.

Normalerweise werden diese Methoden im Rahmen des Progressive Enhancement eingesetzt. Oft reicht es daher aus, für ältere Browser eine Fallback-Farbe anzugeben, wie hier gezeigt:

```
background-color: rgb( 241,218,54);  
background-color: rgba(241,218,54,0.6);
```

Die CSS3-Spezifikation enthält eine Reihe von Beispielen und Farbtafeln, die speziell die `hsla()`-Schreibweise sehr gut illustrieren.

Das Schlüsselwort transparent

War `transparent` in den Versionen 1 und 2.1 der CSS-Spezifikation nur für Rahmenfarben (`border-color`) zulässig, kann dieses Schlüsselwort in der aktuell gültigen CSS3-Spezifikation überall dort verwendet werden, wo ein Farbwert gültig ist.

Internet Explorer 7 und 8

Der Internet Explorer setzt das *Colors*-Modul der CSS3-Spezifikation erst in Version 9 vollständig um. Daher müssen Sie hier entweder auf die Transparenz verzichten oder sich verschiedener proprietärer Microsoft-Techniken bedienen.

Um analog zur Eigenschaft `opacity` ganze Elemente mehr oder weniger durchscheinend zu machen, können Sie die Microsoft-eigene Eigenschaft `filter` verwenden. Hierbei gibt es für Internet Explorer 7 und 8 unterschiedliche Schreibweisen. Da das Verfahren nicht standardkonform ist (aber trotzdem funktioniert), verpacken wir die Anweisungen in Conditional Comments:³

```
<!--[if lte IE 8]>  
  -ms-filter:"progid:DXImageTransform.Microsoft.Alpha(Opacity=50)";  
  filter: alpha(opacity=50);  
<![endif]-->
```

Wollen Sie dagegen nur die Hintergrundfarbe eines Elements mit Transparenz versehen, hilft der einfache Filter aus dem vorigen Beispiel nicht weiter. Hier müssen wir schon etwas tiefer in die Trickkiste greifen. Zwar kommt auch diesmal die `filter`-Eigenschaft zum Einsatz – allerdings benutzen wir hier einen Farbverlauf:

```
filter:progid:DXImageTransform.Microsoft.gradient(startColorstr=#26ffffff,  
endColorstr=#26ffffff);
```

Farbverläufe müssen nicht unbedingt unterschiedliche Farben haben. Dafür kann bei ihrer Definition mit der Eigenschaft `filter` ein transparenter Alphakanal angegeben werden. Die Farbe wird in hexadezimaler Schreibweise festgelegt:

```
#AARRGGBB
```

³ Beachten Sie die Reihenfolge der beiden Deklarationen. Geben Sie die Filter umgekehrt an, werden die Anweisungen vom IE8 im Kompatibilitätsmodus nicht erkannt.

Hierbei wird ein hexadezimaler Wert für die Transparenz (AA) festgelegt. Darauf folgt die gewünschte Farbe in hexadezimaler RGB-Form. Da die Umrechnung nicht gerade intuitiv ist, empfehlen wir, nach Möglichkeit ein Onlinewerkzeug zu verwenden.

Siehe auch

Das *Colors*-Modul der CSS3-Spezifikation unter <http://www.w3.org/TR/css3-color>; den deutschen Wikipedia-Eintrag zum HSL-Konzept unter <http://de.wikipedia.org/wiki/HSL-Farbraum>; den Quirksmode-Artikel zur Eigenschaft *opacity* unter <http://www.quirksmode.org/css/opacity.html>; den Artikel *How to Use rgba in IE* von Kilian Valkhof (mit Werkzeug zum Umrechnen von *rgba()* in die hexadezimale Schreibweise) unter <http://kilianvalkhof.com/2010/css-xhtml/how-to-use-rgba-in-ie/>; den Artikel *Cross-Browser rgba()* von Eric Eggert unter <http://yatil.de/Weblog/cross-browser-rgba>.

12.6 Abgerundete Ecken mit CSS3 definieren

Problem

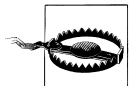
Sie möchten ein Element mit abgerundeten Ecken versehen, ohne hierfür mit Grafiken oder zusätzlichen nicht semantischen HTML-Elementen arbeiten zu müssen.

Lösung

Verwenden Sie die CSS3-Eigenschaft *border-radius*:

```
div.abgerundet {  
    -webkit-border-radius: 0.5em;  
    -moz-border-radius: 0.5em;  
    border-radius: 0.5em;  
}
```

Dies ist die einfachste Form. Die oben stehende Deklaration erzeugt für alle Ecken des Elements eine gleichmäßige Rundung mit einem Radius von 0.5em (siehe Abbildung 12-7).



Vergessen Sie nicht, dass in Stylesheets anstelle des deutschen Kommas (,) ein Punkt (.) verwendet werden muss.

Wir haben die Deklaration für ältere Versionen von Firefox und Safari außerdem in der entsprechenden Präfixschreibweise angegeben.

Sie können mit der Kurzschrift aber auch unterschiedliche Radien angeben:

```
border-radius: 0.5em 1em 0 0.3em;
```

Hier erhält die linke obere Ecke einen Radius von 0.5em, rechts oben ist es 1em, die linke untere Ecke wurde nicht abgerundet, während die Rundung rechts unten 0.3em beträgt. Das lässt sich auch mit Einzeleigenschaften wie folgt ausdrücken:

```
border-top-left-radius: 0.5em;  
border-top-right-radius: 1em;  
border-bottom-left-radius: 0;  
border-bottom-right-radius: 0.3em;
```

Um auch ältere, aber noch aktuelle Browser zu unterstützen, ist es nötig, die entsprechenden Präfixeigenschaften mit anzugeben. Details hierzu finden Sie in der Diskussion. Für den Internet Explorer in den Versionen 7 und 8 bietet sich in diesem Fall die Verwendung eines Polyfill wie CSS3 PIE (siehe Kapitel 13.18) an.

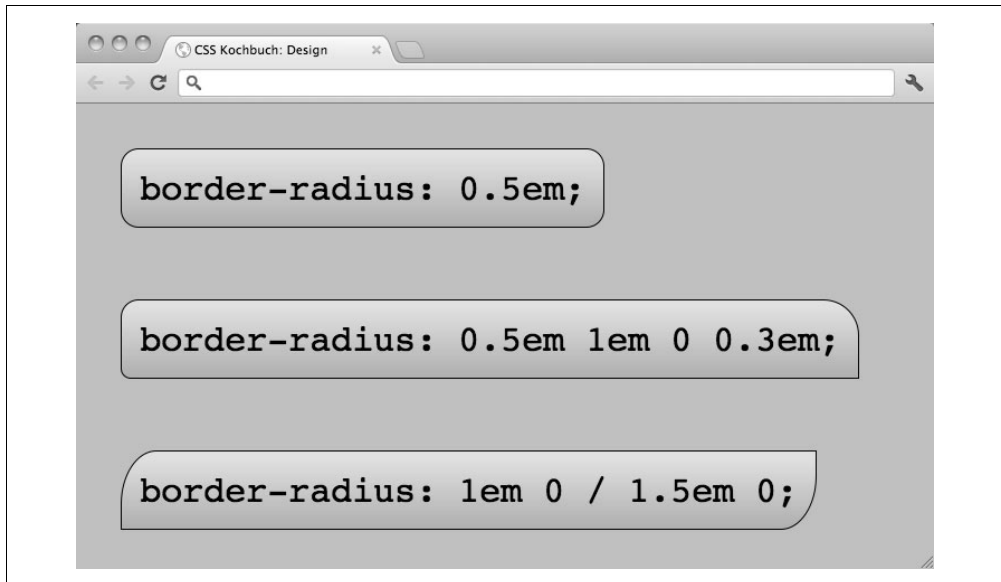


Abbildung 12-7: Elemente mit abgerundeten Ecken

Diskussion

Abgerundete Ecken haben sich in den letzten Jahren einen festen Platz im Werkzeugkasten der Webdesigner erobert. Mit der CSS3-Eigenschaft `border-radius` (bzw. den entsprechenden Präfixeigenschaften) ist es nur in äußerst seltenen Fällen notwendig, grafikbasierte Lösungen einzusetzen.

Üblicherweise sind abgerundete Ecken für die Funktionalität einer Website nicht unbedingt nötig und werden meist im Rahmen des Progressive Enhancement eingesetzt.

Inzwischen wird `border-radius` von den meisten aktuellen Browsern (IE9+, Firefox 4+, Chrome 5+, Safari 5+, Opera 10.5+) in der offiziellen Schreibweise unterstützt. Safari unterstützt die Verwendung von Prozentwerten erst ab Version 5.1.

Für ältere Browser (Firefox 2 bis 3.6, Chrome 4 und Safari 3.1 bis 4.0) hilft entweder die entsprechende Präfixversion oder ein Polyfill wie *CSS3 PIE* oder *ie-css3* weiter.

Bei der Verwendung der Einzeleigenschaften in der Präfixschreibweise unterscheidet sich Firefox (bis inklusive Version 3.6) von der offiziellen Syntax. Hier ein Beispiel für die Definition einer Rundung in der linken oberen Ecke:

```
.obenlinks {  
  -webkit-border-top-left-radius: 0.5em;  
  -moz-border-radius-topleft: 0.5em;  
  border-top-left-radius: 0.5em;  
}
```

Für die anderen Ecken heißen die Einzeleigenschaften entsprechend `-moz-border-radius-topright`, `-moz-border-radius-bottomleft` und `-moz-border-radius-bottomright`.

Firefox 4+, Chrome 5+, Safari 5+, Opera 10.5 und Internet Explorer (ab Version 9) unterstützen `border-radius` direkt ohne Präfix. Für die immer noch aktuellen Versionen 7 und 8 des IE empfehlen wir die Verwendung eines Polyfill wie *CSS3 PIE* oder *ie-css3*.

Hinweise zur Verwendung von Polyfills finden Sie in Rezept 13.18. Bei *CSS3 PIE* werden jedoch keine Einzeleigenschaften unterstützt. Wenn Sie die Rundung nur für bestimmte Ecken definieren wollen, müssen Sie hierfür ebenfalls die Kurzschriftschreibweise verwenden:

```
<!--[if (IE7)|(IE8)]>  
<style type="text/css">  
  .obenlinks-pie {  
    border-radius: 0.5em 0 0 0;  
    behaviour: url('pfad/zu/pie.htc');  
  }  
</style>  
<![endif]-->
```

(Damit der Code trotz der proprietären Eigenschaft `behaviour` validiert, haben wir ihn mit einem Conditional Comment umgeben.)

Die Möglichkeiten der verschiedenen `border-radius`-Eigenschaften sind aber noch nicht ausgeschöpft. So können Sie den Einzeleigenschaften auch zwei Werte mitgeben, wie hier:

```
.doppelwerte-einzel {  
  border-top-left-radius: 1em 1.5em;  
  border-bottom-right-radius: 1em 1.5em;  
}
```

Während die Schreibweise mit einem Wert den Radius eines Viertelkreises in der angegebenen Ecke definiert, wird bei der Zwei-Werte-Variante ein Vierteloval definiert. Hierbei steht der erste Wert für den horizontalen Radius, der zweite für den vertikalen. Dadurch ist die Rundung auf der linken Seite eineinhalb mal länger als an der Oberkante, wie in Abbildung 12-8 gezeigt wird.

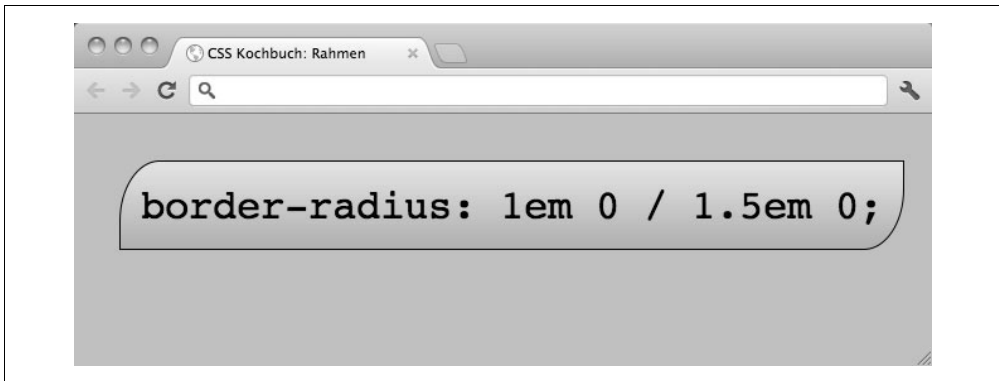


Abbildung 12-8: Durch die Angabe eines zweiten Werts können auch ovale Rundungen erzeugt werden

Bei der Verwendung der Kurzschrifteigenschaft `border-radius` werden zuerst die horizontalen Radien angegeben. Es folgen, durch einen Schrägstrich getrennt, die Definitionen der vertikalen Radien. Das obige Beispiel könnte man also auch so schreiben:

```
.doppelwerte {
  border-radius: 1em 0 / 1.5em 0;
}
```

Abgerundete Ecken für Bilder

Aktuell ist es noch nicht möglich, Bilder ohne zusätzliche Grafiken oder Markup-Code browserübergreifend mit abgerundeten Ecken zu versehen. In manchen Fällen können Sie sich behelfen, indem Sie das Bild mit einem `span`-Element umgeben, für das die abgerundeten Ecken definiert werden. Per `overflow: hidden` für das `span`-Element können die Ecken des Bilds versteckt werden:

```
span.abgerundet {
  border-radius: 10px;
  overflow:      hidden;
}
```

Anstatt die Eigenschaft `overflow` zu verwenden, können Sie den Wert von `border-radius` auch explizit an das Bild vererben:

```
span.abgerundet { border-radius: 10px; }
img              { border-radius: inherit; }
```

Siehe auch

Das *Backgrounds and Borders*-Modul der CSS3-Spezifikation unter <http://www.w3.org/TR/css3-background/>; eine Tabelle zur Unterstützung von `border-radius` in verschiedenen Browsern unter <http://muddledramblings.com/table-of-css3-border-radius-compliance/>; die *Can I Use*-Website zur Browserunterstützung von `border-radius` unter <http://caniuse.com/#search=border-radius>; das Rezept 13.19 zur Verwendung von Präfixeigenschaften; das Rezept 13.18 zu Polyfills; das Rezept 13.13 zu Conditional Comments; eine Liste ver-

schiedener Techniken zum Erstellen abgerundeter Ecken unter <http://www.cssjuice.com/25-rounded-corners-techniques-with-css/>; die Mozilla Developer Network-Website zu border-radius unter <https://developer.mozilla.org/de/CSS/border-radius>.

12.7 Schattenwürfe mit CSS3 erzeugen

Problem

Sie möchten ein Element mit einem Schattenwurf versehen.

Lösung

Verwenden Sie die CSS3-Eigenschaft box-shadow bzw. die passende Präfixschreibweise, zum Beispiel:

```
.schatten {  
  -webkit-box-shadow: 0.2em 0.2em 0.5em rgba(0,0,0,0.6);  
  -moz-box-shadow: 0.2em 0.2em 0.5em rgba(0,0,0,0.6);  
  box-shadow: 0.2em 0.2em 0.5em rgba(0,0,0,0.6);  
}
```

In dieser Regel legen die ersten beiden Teilwerte (0.2em 0.2em) die horizontale und vertikale Verschiebung des Schattens fest. Der dritte Wert (0.5em) definiert den Radius für die Weichzeichnung des Schattens, und der vierte Wert gibt die Farbe des Schattens und dessen Transparenz an. Das Ergebnis sehen Sie in Abbildung 12-9.

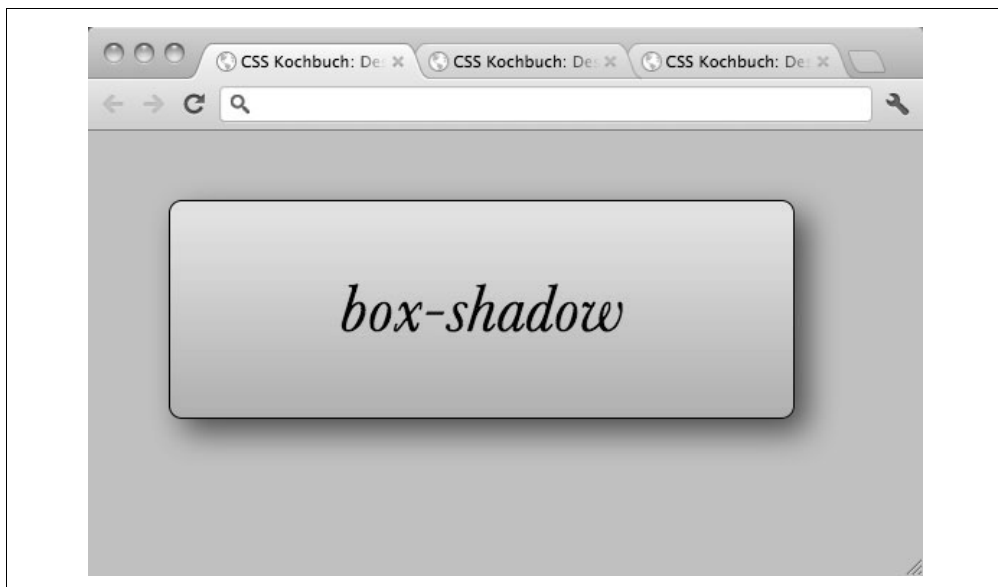


Abbildung 12-9: Mithilfe der Eigenschaft box-shadow erhält das Element einen Schattenwurf

Diskussion

Schattenwürfe mit `box-shadow` sind in allen aktuellen Browsern außer Internet Explorer 7 und 8 möglich. Für diese gibt es aber verschiedene Hilfslösungen, die in Rezept 12.8 behandelt werden. Opera unterstützt `box-shadow` ab der Version 10.5 ohne Präfix. Daher reicht es, die alternative Schreibweise nur für Gecko (Firefox & Co.) und Webkit (Chrome, Safari) anzugeben.

Laut der CSS3-Spezifikation besteht die Definition eines Schattens aus zwei bis vier Längenangaben und einer optionalen Farbangabe. Eine vollständige Deklaration könnte so aussehen:

```
box-shadow: [optional: inset] [horizontale Verschiebung], [vertikale Verschiebung],  
[optional: Weichzeichnungsradius], [optional: Ausdehnung], [optional: Farbangabe];
```

Die Werte für die horizontale und die vertikale Verschiebung müssen auf jeden Fall angegeben werden. Positive Längenangaben platzieren den Schatten rechts bzw. unter dem Element. Negative Werte verschieben den Schatten nach links bzw. oben.

Die übrigen Werte sind optional. Der Weichzeichnungsradius darf nicht negativ sein. Wird kein Wert angegeben, wird der Schatten nicht weichgezeichnet. Mit dem Wert für die Ausdehnung (*spread*) können Sie die Größe des Schattens steuern.

Auch die Farbangabe ist laut Spezifikation optional. Hierbei ist es den Browsern überlassen, welche Farbe in diesem Fall benutzt wird. Um konsistente Ergebnisse zu erhalten, sollten Sie daher auf jeden Fall einen Farbwert angeben.



Das Schlüsselwort `inset` und die Definition der Ausdehnung des Schattens mittels einer vierten Längenangabe werden in Safari ab der Version 5.0 (mit Präfix, ab 5.1 ohne Präfix) unterstützt.

Damit der Schattenwurf möglichst unabhängig vom Hintergrund des Elements oder der Seite funktioniert, verwenden wir hier die `rgba()`-Schreibweise, die von allen aktuellen Browsern (IE ab Version 9) unterstützt wird.

Wenn Sie abgerundete Ecken mit der Eigenschaft `border-radius` definieren, folgt der Schatten dieser Form.

Stellen Sie der Definition eines Schattens das Schlüsselwort `inset` voran, wird dieser nicht außerhalb, sondern innerhalb des Elements erzeugt. Das könnte beispielsweise so aussehen:

```
.inset {  
  box-shadow: inset 0.15em 0.15em 0.2em rgba(0,0,0,0.7);  
}
```

Diese Regel erzeugt einen Schatten innerhalb des Elements. Das Ergebnis sehen Sie in Abbildung 12-10.

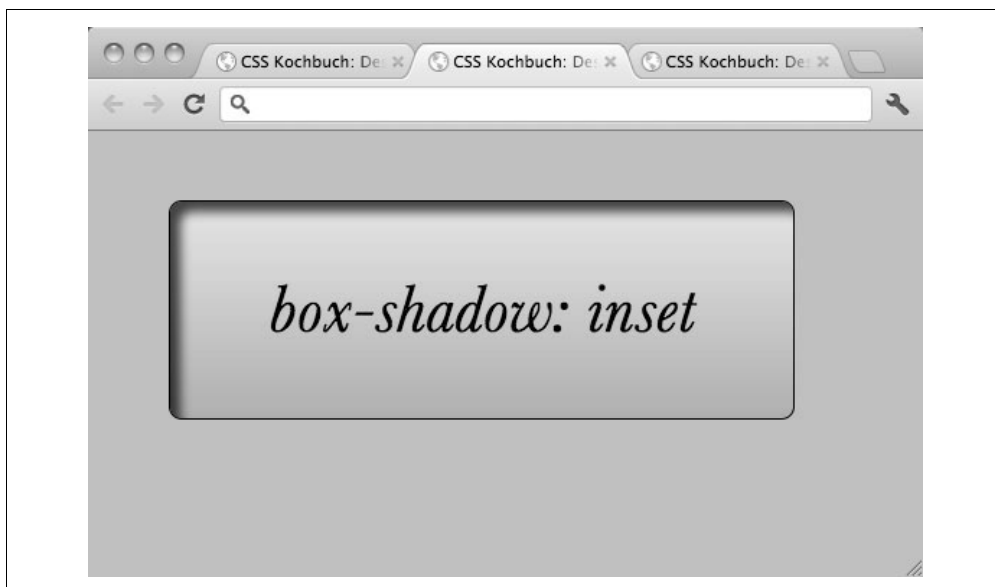
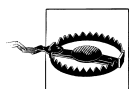


Abbildung 12-10: Durch das Schlüsselwort *inset* wird der Schattenwurf innerhalb des Elements dargestellt

Es ist auch möglich, für ein Element mehrere Schatten auf einmal zu definieren. Hierbei werden die einzelnen Schattendefinitionen durch Kommata voneinander getrennt angegeben, wie hier:

```
.mehrfach {
  box-shadow: 0.2em 0.2em 0.5em rgba(0,0,0,0.6),
             inset 0.15em 0.15em 0.2em rgba(0,0,0,0.7);
}
```

Diese Regel kombiniert den äußeren Schattenwurf aus der Lösung mit dem oben gezeigten inneren Schatten (siehe Abbildung 12-11).



Besonders große CSS-Schattenwürfe können die Browserdarstellung stark verlangsamen. Daher sollten Sie `box-shadow` mit Bedacht einsetzen.

Schattenwürfe beeinflussen das gesamte Layout des Elements. Wurde beispielsweise ein Schatten für ein Element mit einer Breite von 100% definiert, kann es passieren, dass bei der Verwendung von `box-shadow` Scrollbalken angezeigt werden. Unter Umständen müssen die Werte für `padding`, `margin`, `border-width`, `height` und `width` angepasst werden, um Nebenwirkungen zu verhindern.

Wollen Sie Schattenwürfe für ein Element ausdrücklich deaktivieren, verwenden Sie die Deklaration `box-shadow: none`.



Abbildung 12-11: Mehrere Schatten können in einer Deklaration kombiniert werden

Siehe auch

Das *Backgrounds and Borders*-Modul der CSS3-Spezifikation unter <http://www.w3.org/TR/css3-background/#the-box-shadow>; den Mozilla Developer Network-Artikel zur Eigenschaft `box-shadow` unter <https://developer.mozilla.org/de/CSS/box-shadow>; die *Can I Use*-Website zur Browserunterstützung für `box-shadow` unter <http://caniuse.com/#feat=css-boxshadow>; eine Seite mit verschiedenen Tests für die Browserunterstützung für `box-shadow` unter http://www.elektronotdienst-nuernberg.de/bugs/box-shadow_inset.html.

12.8 Realistische Schattenwürfe im Internet Explorer vor Version 9 erzeugen

Problem

Ein Bild soll auch in älteren Versionen des Internet Explorer (8 und darunter) mit einem ästhetisch ansprechenden Schattenwurf versehen werden.

Lösung

Fügen Sie mithilfe eines Conditional Comment eine Kopie des Bilds in den HTML-Code ein. Positionieren Sie die Bilder übereinander und definieren Sie dann mit der proprietären Eigenschaft `filter()` einen passenden Weichzeichnungseffekt für das unten liegende

Bild. Für die Positionierung ist es hilfreich, beide Kopien mit einem div-Container zu umgeben. Der HTML-Code hierfür könnte beispielsweise so aussehen:

```
<!--[if lte IE 8]><div class="ie-img-container">

<![endif]-->

<!--[if lte IE 8]></div><![endif]-->
```

Im Stylesheet werden die beiden Bilder nun überlagert. Das eingefügte Bild wird weichgezeichnet und als Schatten definiert. Hier die nötigen Regeln:

```
<!--[if lte IE 8]>
<style type="text/css">
div.ie-img-container {
    position: relative;
}

div.ie-img-container img {
    position: absolute; /* gilt für beide */
    top:    0;
    left:   0;
}

img.ie-schatten {
    filter: progid:DXImageTransform.Microsoft.Blur( PixelRadius=5,MakeShadow=true);
}
</style>
<![endif]-->
```

Zunächst schaffen wir im Stylesheet per `position: relative` den nötigen Positionierungskontext für die beiden Bilder. In der zweiten Regel können wir dann beide Bilder absolut positionieren und so überlagern. Da das Schattenbild in der Quellcodereihenfolge vor dem Original eingefügt wurde, wird es vom Original überlagert. Die Verwendung der Eigenschaft `z-index` ist daher nicht nötig.

In der dritten Regel kommt schließlich der Filter zum Einsatz. Die Filterfunktion `Blur` (Weichzeichnen) erhält zwei Parameter: Mit `PixelRadius` können Sie, ähnlich dem dritten Wert für die CSS3-Eigenschaft `box-shadow`, festlegen, wie stark der Schatten weichgezeichnet wird. Der Wert `true` für den Parameter `MakeShadow` sorgt dafür, dass anstelle des Bilds der gewünschte Schattenwurf erzeugt wird. Das Ergebnis sehen Sie in Abbildung 12-12.

Diskussion

Zwar besitzt der Internet Explorer seit Version 5.5 die Möglichkeit, Schattenwürfe über die Filterfunktionen *Shadow* und *DropShadow* zu definieren. Leider sind diese aber alles andere als schön und im Design daher meist fehl am Platz.

Die hier gezeigte Lösung basiert auf der Möglichkeit, ein Bild im Internet Explorer weichzeichnen zu lassen und per `MakeShadow=true` als Schatten zu definieren.



Abbildung 12-12: Ein Schattenwurf im Internet Explorer 8

Da die beiden Kopien des Bilds überlagert werden sollen, benötigen wir einen entsprechenden Positionierungskontext. Diesen schaffen wir in unserer Lösung explizit durch das eingefügte Container-div, das relativ positioniert wird (siehe Rezept 2.18). Danach ist es möglich, beide Versionen des Bilds mithilfe der absoluten Positionierung übereinander zu positionieren.

Hierbei nutzen wir die Quellcodereihenfolge. Das Bild, das uns als Schatten dienen soll, wird vor dem Original in den Quellcode eingefügt. Dadurch erscheint es in der Darstellung hinter dem Original.

(Möglicherweise reicht es aus, das umgebende Element relativ zu positionieren und als Kontext für die Positionierung zu benutzen. In diesem Fall wird das Container-div nicht gebraucht.)



Eventuell ist es möglich, die Kopie des Bilds und gegebenenfalls das Container-div automatisch (z.B. per PHP) einzufügen.

Normalerweise verwenden wir in diesem Buch Conditional Comments nur, um Stylesheets oder einzelne Stilregeln für bestimmte Versionen des Internet Explorer einzubinden. Conditional Comments sind aber nicht auf den head-Teil der Seite beschränkt. Das machen wir uns hier zunutze, um bei Bedarf die nötigen Elemente in den HTML-Code einzufügen. Andere Browser interpretieren diese Anweisungen als das, was sie eigentlich sind: einfache HTML-Kommentare. Nur für Internet Explorer bis inklusive Version 9 haben sie eine bestimmte Bedeutung.

Die Verwendung von Conditional Comments, speziell im body-Teil der Seite, ist umstritten, weil sie die Wartbarkeit von Webseiten-Templates verringern kann. Eine mögliche Alternative wäre der Einsatz von jQuery, um die Kopie des Bilds, das div-Element und die nötigen Stildefinitionen einzufügen.⁴

Damit der Filter funktioniert, muss für das Element *hasLayout* aktiv sein. Das ist bei Bildern prinzipiell der Fall. Für andere Elemente müssen Sie möglicherweise noch tätig werden.

Ab der Version 9 des Internet Explorer gilt der *Blur*-Filter offiziell als veraltet (*deprecated*). Stattdessen können Sie die CSS-Eigenschaft *box-shadow* verwenden.

Siehe auch

Das Rezept 2.19 zum Positionierungskontext; das Rezept 13.13 zur Verwendung von Conditional Comments; die Microsoft-Seite zum *Blur*-Filter unter [http://msdn.microsoft.com/de-de/library/ie/ms532979\(v=vs.85\).aspx](http://msdn.microsoft.com/de-de/library/ie/ms532979(v=vs.85).aspx); die Microsoft-Seite zum *Shadow*-Filter unter [http://msdn.microsoft.com/en-us/library/ie/ms533086\(v=vs.85\).aspx](http://msdn.microsoft.com/en-us/library/ie/ms533086(v=vs.85).aspx); das Rezept 13.15 zum Thema *hasLayout*.

12.9 Elemente umformen

Problem

Sie möchten die Elemente einer Seite verschieben, drehen, verzerren oder strecken.

Lösung

Verwenden Sie die CSS3-Eigenschaft *transform* in der Präfixschreibweise. Um ein Element zu verändern, stehen verschiedene Funktionen zur Verfügung. Soll das Element ausgehend von seiner ursprünglichen Position verschoben werden, verwenden Sie die Funktionen *translateX()* bzw. *translateY()*. Die folgende Deklaration verschiebt das Element mit der ID *#verschieben* um 5em nach links und um 4em nach unten:

⁴ Oder aber Sie deklarieren die Schattenwürfe für IE8 und darunter als Progressive Enhancement. In diesem Fall brauchen Sie überhaupt keine Änderungen vorzunehmen.

```
#verschieben {
  -webkit-transform: translateX(5em) translateY(4em);
  -moz-transform: translateX(5em) translateY(4em);
  -ms-transform: translateX(5em) translateY(4em);
  -o-transform: translateX(5em) translateY(4em);
  transform: translateX(5em) translateY(4em);
}
```

Da die Eigenschaft `transform` noch nicht finalisiert ist, muss für alle Browser die Präfixschreibweise eingesetzt werden. Trotzdem geben wir hier die W3C Schreibweise mit an. Die Funktion `translateX()` bewirkt eine Verschiebung auf der x-Achse (horizontal), während `translateY()` für die y-Achse (vertikal) zuständig ist.



Um Platz zu sparen, geben wir hier nicht in jedem Beispiel den kompletten Satz an Präfixeigenschaften an, sondern nur die Schreibweise ohne Präfix.

Die beiden `translate`-Funktionen können auch mit einer Kurzschriffunktion zusammengefasst werden. Das obige Beispiel würde dann so aussehen:

```
#verschieben {
  transform: translate(5em,4em);
}
```

Bei dieser Schreibweise ist der zweite Wert optional. Geben Sie keinen Wert für die y-Achse an, wird dieser automatisch auf null gesetzt.

Anhand der Funktionen `scaleX()` und `scaleY()` können Sie ein Element skalieren, sprich: vergrößern und verkleinern. Der Wert für diese Funktionen ist jeweils eine positive Fließkommazahl, die den Skalierungsfaktor angibt. Der Faktor 1 steht für »keine Größenänderung«. Werte größer als 1 vergrößern, Werte darunter verkleinern das Element. Um ein Element doppelt so breit und anderthalbmal so hoch wie das Original darzustellen, könnten Sie schreiben:

```
#skaliert {
  transform: scaleX(2) scaleY(1.5);
}
```

Beachten Sie, dass die Skalierung vom Mittelpunkt des Elements in alle Richtungen gleichmäßig ausgeführt wird. Steht das Element sehr weit am Rand der Seite, kann es passieren, dass es teilweise außerhalb des Ansichtsbereichs dargestellt wird.

Für die gleichmäßige Skalierung von Elementen gibt es außerdem die Kurzschriffunktion `scale()`, mit der Sie beide Werte auf einmal angeben können:

```
#skaliert {
  transform: scale(2, 1.5);
}
```

Diese Funktion bewirkt die gleiche Skalierung wie die oben stehende Kombination aus `scaleX()` und `scaleY()`. Soll das Element in beide Richtungen gleichmäßig skaliert werden, reicht es, einen Wert anzugeben:

```
#skaliert {
  transform: scale(1.5);
}
```

Die obige Deklaration sorgt dafür, dass das Element horizontal und vertikal um den Faktor 1.5em vergrößert wird.

Um ein Element zu drehen, verwenden Sie die Funktion `rotate()`. Der Parameter ist ein Winkel zwischen 0 und 360 Grad:

```
#rotiert {
  transform: rotate(45deg);
}
```

Diese Regel sorgt dafür, dass das Element um 45 Grad im Uhrzeigersinn gedreht wird. Um die gleiche Drehung in die andere Richtung auszuführen, geben Sie entweder eine entsprechend hohe Gradzahl (315deg) an, oder Sie nehmen einen negativen Wert:

```
#rotiert {
  transform: rotate(-45deg);
}
```

Wenn Sie ein Element verzerren wollen, können Sie hierfür die Funktionen `skewX()` und `skewY()` verwenden. Eine Verzerrung entlang der x-Achse erreichen Sie beispielsweise so (siehe Abbildung 12-13):

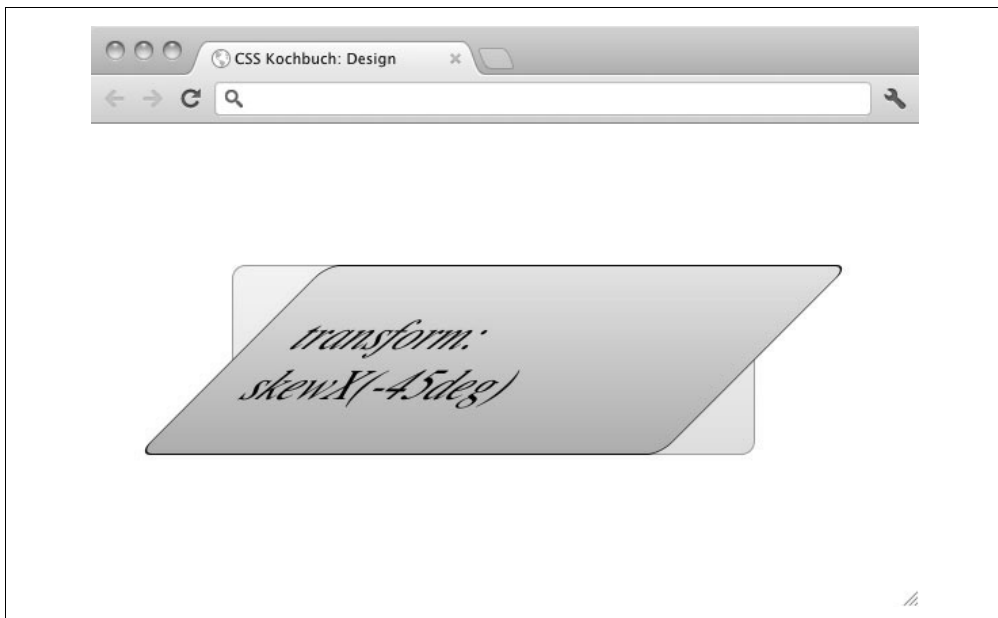


Abbildung 12-13: Ein Element wurde entlang der x-Achse verzerrt

```
#verzerrung
  transform: skewX(-45deg);
}
```

Ähnlich wie bei `rotate()` ist auch bei den `skew`-Funktionen der Parameter für die Funktion ein Winkel, der die Richtung (über das Vorzeichen) und den Grad der Verzerrung angibt. Eine Verzerrung entlang der y-Achse erreichen Sie entsprechend mit der Funktion `skewY()`:

```
#verzerrung {  
  transform: skewY(10deg);  
}
```

Die Funktionen können auch miteinander kombiniert werden:

```
#kombiniert {  
  transform: translate(2em, 3em) scale(0.6) rotate(-30deg) skewX(-45deg);  
}
```

Diese Regel verschiebt das Element um 2em nach rechts und um 3em nach unten. Danach wird es um den Faktor 0.6 verkleinert, um 30 Grad entgegen dem Uhrzeigersinn gedreht und schließlich um 45 Grad entlang der x-Achse nach links verzerrt.

Diskussion

Mit der Eigenschaft `transform` kann die Darstellung eines Elements sehr vielfältig verändert werden. Anhand verschiedener Funktionen werden die Veränderungen im Stylesheet definiert.

Die Veränderungen werden anhand verschiedener Funktionen durchgeführt, denen je nach Zweck unterschiedliche Werte als Parameter (umgeben von einem Paar runder Klammern) übergeben werden. Die folgende Tabelle soll Ihnen einen Überblick geben:

Tabelle 12-1: Mögliche Werte für die Eigenschaft `transform`

Funktion(en)	Parameter
<code>translateX()</code> , <code>translateY()</code>	Längenangabe
<code>translate()</code>	Längenangabe, Längenangabe (optional)
<code>scaleX()</code> , <code>scaleY()</code>	positive Fließkommazahl (1 = Originalgröße)
<code>scale()</code>	positive Fließkommazahl, positive Fließkommazahl (optional)
<code>rotate()</code>	Winkel
<code>sewX()</code> , <code>skewY()</code>	Winkel

Durch die Kombination der verschiedenen Funktionen kann ein Element auf recht komplexe Weise verändert werden. (So richtig spannend wird dies, wenn die statischen `transform`-Funktionen animiert werden. Informationen zu CSS-Animationen finden Sie in Rezept 12.10.)

Zusätzlich zu den hier gezeigten Funktionen gibt es außerdem noch die Möglichkeit, eigene Transformationen zu definieren, indem Sie die Änderungen in Form einer *Matrix* angeben. Hierfür verwenden Sie bei Bedarf die Funktion `matrix()`. Eine ausführliche

Abhandlung würde allerdings den Rahmen dieses Buchs sprengen. Weitere Informationen finden Sie unter anderem in der Spezifikation und im Artikel von André Bräking. (Die Links finden Sie im Abschnitt »Siehe auch«.)

transform-origin

Ergänzend zu `transform` gibt es die Eigenschaft `transform-origin`. Hiermit können Sie den *Ursprung* einer Transformation festlegen. Das ist beispielsweise bei der Drehung eines Elements interessant. Anstatt das Element standardmäßig um seinen Mittelpunkt zu drehen, können Sie mit der folgenden Anweisung die Drehung zum Beispiel um die linke obere Ecke durchführen:

```
#ursprung {  
  transform-origin: top left;  
  transform: rotate(45deg);  
}
```

Ähnlich wie bei der Positionierung von Hintergrundbildern können Sie auch für `transform-origin` Prozentwerte, CSS-Längenangaben oder verschiedene Schlüsselwörter nutzen. Der erste Wert gibt die horizontale Position an, der zweite steht für die vertikale Position.

Offiziell können Sie unterschiedliche Werte miteinander kombinieren (z.B. `top 50%`). Im wahren Leben funktioniert das aber nicht immer wie gewünscht. Daher ist es besser, wenn Sie für beide Werte den gleichen Typ verwenden.

Die folgende Tabelle gibt einen Überblick über die möglichen Werte für `transform-origin`:

Tabelle 12-2: Mögliche Werte für die Eigenschaft `transform-origin`

Erster Wert	Zweiter Wert
Prozentwert	Prozentwert
Längenangabe	Längenangabe
top, center, bottom	left, center, right

Wird nur der erste Wert angegeben, wird für den zweiten Wert automatisch die Angabe `center` bzw. `50%` angenommen.

Internet Explorer 7 und 8

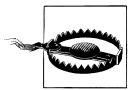
Von Haus aus versteht der Internet Explorer in den Versionen 7 und 8 die Eigenschaften `transform` und `transform-origin` noch nicht. Mithilfe des proprietären *Matrix*⁵-Filters können Sie auch für diese Browser die gewünschten Transformationen erreichen. Hierbei kommen Verschiebungsmatrizen zur Anwendung, die angeben, wie das Element verändert werden soll.

⁵ Hiermit ist nicht der Film gemeint.

Eine Drehung um 20 Grad im Uhrzeigersinn sieht dann beispielsweise so aus:

```
<!--[if lte IE 8]>
<style type='text/css'>
#drehung-ie {
  filter: progid:DXImageTransform.Microsoft.Matrix(M11=0.9396926207859084,
    M12=-0.3420201433256687, M21=0.3420201433256687, M22=0.9396926207859084,
    sizingMethod='auto expand');
  zoom: 1;
}
</style>
<![endif]-->
```

Wie Sie sehen, sind diese Werte alles andere als intuitiv. Daher empfehlen wir, für die Erstellung ein Werkzeug wie beispielsweise *css3please.com* zu verwenden.⁶



Auch wenn die transform-Eigenschaften bereits in allen gängigen Browsern in der Präfixschreibweise implementiert sind, kann es passieren, dass am Verhalten noch einige Veränderungen vorgenommen werden.

Transformationen und feste Positionierung

Normalerweise sorgt die Deklaration `position: fixed` dafür, dass ein Element zuverlässig im Ansichtsbereich verankert wird, auch wenn der Rest der Seite gescrollt wird. Das Verhalten ändert sich jedoch, wenn für das umgebende Element die Eigenschaft `transform` einen anderen Wert hat als `none`. Angenommen, Sie haben folgenden HTML-Code:

```
<div id="aussen">
  <div id="innen">INNEN</div>
</div>
```

Hierfür werden nun die folgenden CSS-Regeln definiert:

```
#aussen {
  -webkit-transform: rotate(10deg);
}

#innen {
  position: fixed;
}
```

Eigentlich sollte man erwarten, dass das innere Element weiterhin im Ansichtsbereich verankert bleibt. Tatsächlich bewirkt `transform` für das äußere Element aber, dass sich die Positionierung nun auf das umgebende Element bezieht. Dadurch geht die feste Positionierung für das innere Element verloren, und es wird nun mit dem äußeren Element zusammen gescrollt.

⁶ Die Anweisung `zoom: 1` sorgt dafür, dass für das Element das Konzept des *hasLayout* aktiviert wird. Mehr dazu finden Sie in Rezept 13.15.

Eric A. Meyer, Autor des Buchs *CSS – Das umfassende Handbuch* (O'Reilly 2007), hat hierzu einen ausführlichen Artikel geschrieben. Den Link finden Sie im Abschnitt »Siehe auch«.



Ende 2011 wurde die bisher eigenständige Spezifikation für Transformationen in CSS mit der entsprechenden SVG-Spezifikation zusammengeführt.

Siehe auch

Die Website *Can I Use* zur Browserunterstützung von CSS-2-D-Transformationen unter <http://caniuse.com/#search=css3%20transforms>; den aktuellen Arbeitsentwurf zur kombinierten Spezifikation von CSS-2-D-Transformationen unter <http://dev.w3.org/csswg/css3-transforms/>; ein Onlinewerkzeug zur Erstellung von CSS-Transformationen unter <http://westciv.com/tools/transforms/index.html>; den Artikel von Eric Meyer zum Problem der festen Positionierung bei gleichzeitiger Verwendung von `transform` für das umgebende Element unter <http://meyerweb.com/eric/thoughts/2011/09/12/un-fixing-fixed-elements-with-css-transforms/>; das Rezept 2.18 zur relativen Positionierung; den deutschsprachigen Artikel von André Bräklings zur Verwendung von CSS-2-D-Transformationen unter <http://www.braekling.de/web-design/3443-css3-transform-lineare-transformationen.html>; ein Werkzeug zur Berechnung von Verschiebungsmatrizen für die `matrix()`-Funktion unter <http://www.braekling.de/testlab/css3tools/css-transform-matrizenrechner.html>; ein Onlinewerkzeug (unter anderem) zum Erzeugen der Verschiebungsmatrizen für IE 7 und 8 unter <http://css3please.com/>.

12.10 Einfache Animationen erstellen

Problem

Sie möchten den Wechsel von Eigenschaftswerten zum Beispiel mit den Pseudoklassen `:focus` oder `:hover` animieren.

Lösung

Verwenden Sie die CSS3-Eigenschaft `transition` bzw. die entsprechenden Einzeleigenschaften in der Präfixschreibweise. Das folgende Beispiel verwendet ein `div`-Element mit der ID `#transit`:

```
#transit {
    color:          black;
    background-color: orange;

    -webkit-transition-property: background-color, color;
    -webkit-transition-duration: 0.5s;
    -webkit-transition-delay:   0.2s;
```

```

-webkit-transition-timing-function: linear;

-moz-transition-property: background-color, color;
-moz-transition-duration: 0.5s;
-moz-transition-delay: 0.2s;
-moz-transition-timing-function: linear;

-ms-transition-property: background-color, color;
-ms-transition-duration: 0.5s;
-ms-transition-delay: 0.2s;
-ms-transition-timing-function: linear;

-o-transition-property: background-color, color;
-o-transition-duration: 0.5s;
-o-transition-delay: 0.2s;
-o-transition-timing-function: linear;

transition-property: background-color, color;
transition-duration: 0.5s;
transition-delay: 0.2s;
transition-timing-function: linear;
}

#transit:hover {
    background-color: yellow;
    color: gray;
}

```

Wie Sie sehen, haben wir für das Pseudoelement `:hover` die Eigenschaften `background-color` und `color` geändert. Befindet sich der Mauszeiger über dem Element, erhalten die Eigenschaften neue Werte. Um den Wechsel der Werte fließender zu gestalten, bedienen wir uns der verschiedenen `transition`-Eigenschaftseigenschaften. Diese werden für das Element definiert, dessen Werte geändert werden (nicht für das Pseudoelement).

Die Eigenschaft `transition-property` übernimmt eine durch Kommata getrennte Liste der Eigenschaften, die animiert werden sollen. Sollen alle geänderten Werte animiert werden, können Sie hier auch das Schlüsselwort `all` angeben.

Die Dauer der Animation wird über die Eigenschaft `transition-duration` bestimmt. Der Wert ist eine Zeitangabe in Sekunden (s) oder Millisekunden (ms). In unserem Fall soll die Animation für Hintergrund- und Textfarbe eine halbe Sekunde dauern.

Mit `transition-delay` können Sie eine mögliche Verzögerung definieren. Hier haben wir uns entschieden, 0,2 Sekunden zu warten, bevor die Animation ausgelöst wird.

Abschließend legen wir per `transition-timing-function: linear` noch fest, dass die Animation gleichmäßig ablaufen soll. Anhand verschiedener Schlüsselwörter können Sie den Ablauf beeinflussen. Verwenden Sie beispielsweise `ease-in` als Wert, beginnt die Animation langsam, um dann gleichmäßig bis zum Ende durchzulaufen. Details zu den Schlüsselwörtern finden Sie in der Diskussion.

Anstelle der Einzeleigenschaften können Sie auch die Kurzschrifteigenschaft `transition` benutzen. Der obige Regelsatz könnte beispielsweise so zusammengefasst werden:

```
#transit {
  color:          black;
  background-color: orange;

  -webkit-transition: background-color 0.5s linear 0.2s,
                      color 0.5s linear 0.2s;
  -moz-transition:  background-color 0.5s linear 0.2s,
                      color 0.5s linear 0.2s;
  -ms-transition:   background-color 0.5s linear 0.2s,
                      color 0.5s linear 0.2s;
  -o-transition:    background-color 0.5s linear 0.2s,
                      color 0.5s linear 0.2s;
  transition:       background-color 0.5s linear 0.2s,
                      color 0.5s linear 0.2s;
}

#transit:hover {
  background-color: yellow;
  color:          gray;
}
```



Der erste Wert (0.5s) wird der Eigenschaft `transition-duration` zugewiesen. Ein möglicher zweiter Wert (0.2s) steht für `transition-delay`.

Die Reihenfolge der einzelnen Werte entspricht hier der Definition der Einzeleigenschaften im ersten Beispiel. Um weitere Eigenschaften zu animieren, trennen Sie die nötigen Werte durch ein Komma von den vorangehenden Definitionen.

Werden mehrere Eigenschaften mit den gleichen Werten für `transition-duration`, `transition-function` usw. animiert, ist es unter Umständen einfacher, die animierten Eigenschaften nicht einzeln zu benennen, sondern das Schlüsselwort `all` einzusetzen:

```
-webkit-transition: all 0.5s linear 0.2s;
-moz-transition:   all 0.5s linear 0.2s;
-ms-transition:    all 0.5s linear 0.2s;
-o-transition:     all 0.5s linear 0.2s;
transition:        all 0.5s linear 0.2s;
```

Beachten Sie, dass sich diese Anweisung auf alle animierbaren Eigenschaften, deren Wert sich ändert, angewandt wird.

Diskussion

Im ersten Rezept dieses Kapitels haben wir erläutert, wie Sie Rollover-Effekte mit dem Pseudoelement `:hover` erzeugen können. Mithilfe der CSS3-Eigenschaft `transition` können Sie diese Übergänge nun auch animieren.

Die folgende Tabelle gibt einen Überblick über die animierbaren CSS 2.1-Eigenschaften und die möglichen Werte:

Tabelle 12-3: Animierbare CSS-Eigenschaften und Werte

Eigenschaft	Animierbare Werte
background-color	Farbwert
background-image	nur Farbverläufe per gradient()
background-position	Prozentwert, Längenangabe
border-color (bzw. die entsprechenden Einzeleigenschaften)	Farbwert
border-width (bzw. die entsprechenden Einzeleigenschaften)	Längenangabe
top, left, bottom, right	Prozentwert, Längenangabe
color	Farbwert
crop	Rechteck
font-size	Prozentwert, Längenangabe
font-weight	Zahlwert
width, height	Prozentwert, Längenangabe
letter-spacing	Längenangabe
line-height	Zahlwert, Prozentwert, Längenangabe
padding, margin (bzw. die entsprechenden Einzeleigenschaften)	Längenangabe
max-height, max-width, min-height, min-width	Prozentwert, Längenangabe
opacity	Zahlwert
outline-color	Farbwert
outline-width	Längenangabe
text-indent	Prozentwert, Längenangabe
text-shadow	Schattendefinition
word-spacing	Prozentwert, Längenangabe
z-index	Ganzzahl

Neben diesen »offiziellen« Eigenschaften gibt es noch eine Reihe weiterer CSS-Eigenschaften, die per transition animiert werden können. Hierzu gehören insbesondere die in Rezept 12.9 behandelten transform-Eigenschaften, aber auch Dinge wie background-size, border-radius usw. Eine Liste mit weiteren proprietären oder Präfixeigenschaften finden Sie im Abschnitt »Siehe auch«.



Die oben stehende Tabelle enthält nur die animierbaren Eigenschaften der CSS 2.1-Spezifikation. Für CSS3-Eigenschaften ist laut der CSS-Arbeitsgruppe des W3C vorgesehen, die Definitionen in der jeweiligen Spezifikation mit Angaben zur Animierbarkeit zu ergänzen.

Bei der Angabe von Kurzschrifteigenschaften werden diese nicht selbst animiert, sondern die entsprechenden Einzeleigenschaften.

Die transition-Eigenschaften müssen für das Element angegeben werden, das animiert werden soll. In unserem Beispiel ist das das div-Element #transit. Die Animation wird ausgelöst, wenn der Mauszeiger über das Element bewegt wird.

Wird der Cursor wieder vom Element entfernt, wird die Animation ohne jede Verzögerung umgekehrt, und die Eigenschaften erhalten ihre ursprünglich definierten Werte zurück. Wenn Sie die Animation beim Zurückkehren in den Ursprungszustand genauer steuern wollen, können Sie das Pseudoelement :hover mit eigenen transition-Anweisungen versehen.

Kombiniert mit den verschiedenen transform-Eigenschaften aus CSS3, lassen sich mit wenigen Zeilen CSS-Code Animationseffekte realisieren, die bisher nur mit JavaScript möglich waren. Für das folgende Beispiel wird ein Bild mithilfe der Eigenschaft transform ein wenig gedreht und verkleinert. Wird der Cursor darüber bewegt, wird die Drehung aufgehoben, und das Bild erhält seine normale Größe wieder. Das Ganze wird per transition animiert, wie Abbildung 12-14 zeigt.



Die transform-Eigenschaften können aktuell nur in der Präfixschreibweise benutzt werden! Werden Präfixeigenschaften animiert, müssen diese auch für transition mit Präfix angegeben werden, z.B.: -moz-transition: -moz-transform 1s; und so weiter.

Den Ablauf der Animation können Sie mit der Eigenschaft transition-timing-function beeinflussen. Hierfür steht eine Reihe von Schlüsselwörtern zur Verfügung (siehe Tabelle 12-4)

Tabelle 12-4: Schlüsselwörter für die Eigenschaft transition-timing-function

Name	Beschreibung
ease	Anfang und Ende der Animation sind verlangsamt. Standardwert
ease-in	Der Beginn der Animation ist verlangsamt.
ease-out	Das Ende der Animation ist verlangsamt.
ease-in-out	Ähnlich wie ease. Allerdings beschleunigt und verlangsamt sich die Animation nicht so schnell.
linear	Die Animation verläuft durchgehend mit der gleichen Geschwindigkeit.
cubic-bezier(x1,x2,y1,y2)	Die ersten beiden Werte definieren die Anfangsgeschwindigkeit, die beiden letzten Werte stehen für die Endgeschwindigkeit. (Details siehe unten.)
steps(Zahl, start bzw. end)	Die Animation wird stufenweise ausgeführt.

In dieser Tabelle finden Sie zwei Werte, die wir bisher nicht besprochen haben. Die Funktion cubic-bezier() ermöglicht es Ihnen, den Ablauf der Animation durch die Definition einer Bézierkurve zu steuern. Eine ausführliche Beschreibung finden Sie beispielsweise im Mozilla Developer Network (vgl. dazu den Link im Abschnitt »Siehe auch«).



Abbildung 12-14: Befindet sich der Mauszeiger über dem Bild, wird es gedreht und erhält seine Originalgröße wieder

Mit der Funktion `steps()` können Sie die Animation stufenweise ablaufen lassen. Obwohl sie nur im *Editor's Draft* des W3C auftaucht, wird sie bereits in Chrome 8+, Safari 5.1+ und Firefox 6+ unterstützt. (Um das Beispiel einfach zu halten, verzichten wir hier auf die entsprechenden Präfixformen.)

```
transition-duration:      1s;  
transition-timing-function: steps(5, start);
```

Statt fließend läuft die Animation in fünf Einzelschritten ab, die gleichmäßig auf die mit `transition-duration` angegebene Zeit verteilt werden. Der erste Wert gibt die Anzahl der Schritte an, der zweite legt fest, ob der erste Animationsschritt gleich ausgeführt werden soll (`start`) oder nach Verstreichen des errechneten Intervalls (`end`). In unserem Beispiel wird der erste Animationsschritt sofort ausgeführt. Danach wird 0,2 Sekunden gewartet, bevor der nächste Schritt folgt, und so weiter bis zum Ende.



Unterstützt ein Browser die Funktion `steps()` noch nicht, wird stattdessen der Standardwert `ease` genommen. Wird `steps()` in der Kurzschrifteigenschaft `transition` definiert, aber nicht unterstützt, wird die Regel möglicherweise als ungültig angesehen und verworfen.

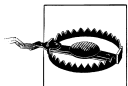
Die Kurzschrifteigenschaft `transition`

Mit der Kurzschrifteigenschaft `transition` können die Werte der Einzeleigenschaften zusammengefasst werden. Damit das funktioniert, müssen zumindest die Werte für `transition-property` und `transition-duration` angegeben werden. Für `transition-timing-function` und `transition-delay` werden in diesem Fall automatisch die Standardwerte `ease` bzw. `0s` verwendet.

Browserunterstützung

Animationen mit der Eigenschaft `transition` werden von fast allen aktuellen Browsern unterstützt. Firefox versteht `transition` ab der Version 4, Opera ist seit der Version 10.5 an Bord. Der Internet Explorer kann mit großer Wahrscheinlichkeit ab Version 10 mit `transition` umgehen.

Da `transition` bzw. die dazugehörigen Einzeleigenschaften noch nicht finalisiert sind, müssen Sie die Eigenschaften momentan in allen Browsern in der jeweiligen Präfixschreibweise angeben. Trotzdem sollten Sie am Ende des Regelsatzes die Standardform ohne Präfix nicht vergessen.



Das vorliegende Rezept basiert auf aktuellen Browserimplementierungen beim Schreiben dieses Buchs. Es ist möglich, dass sich diese vor der Finalisierung der `transition`-Eigenschaften verändern.

Speziell die fehlende Unterstützung in den gegenwärtigen Versionen des Internet Explorer machen die Eigenschaft `transition` zu einem perfekten Kandidaten für das Progressive Enhancement. Browser, die die `transition`-Eigenschaften noch nicht verstehen, können möglicherweise per Polyfill zum Mitmachen überredet werden.

Das *Animations*-Modul der CSS3-Spezifikation definiert die Möglichkeit, noch deutlich komplexere Animationen mithilfe von Keyframes zu erstellen. Allerdings ist die Browserunterstützung hierfür momentan noch relativ gering. Sollte sich die Situation ändern, werden wir in einer Folgeauflage darauf eingehen.

Momentan ist es nicht ganz einfach, herauszufinden, welche Eigenschaften animiert werden können. Für Eigenschaften aus der CSS 2.1-Spezifikation enthält der aktuelle Arbeitsentwurf des Moduls eine Tabelle. Für CSS3-Eigenschaften ist vorgesehen, die »Property Definition«-Tabellen um einen entsprechenden Eintrag zu erweitern.

Siehe auch

Die offizielle Spezifikation für die verschiedenen transition-Eigenschaften unter <http://www.w3.org/TR/css3-transitions/>; den aktuellen Entwurf der Spezifikation unter <http://dev.w3.org/csswg/css3-transitions/#properties-from-css>; eine ausführliche Beschreibung der Funktion cubic-bezier im Mozilla Developer Network unter <https://developer.mozilla.org/en/CSS/timing-function>; eine umfangreiche Sammlung mit Beispielen und Live-Demos unter <http://www.olivergast.de/wp-content/demos/transitions/index.html>; die deutschsprachige Website CSS-Wiki unter <http://www.css-wiki.com/css-eigenschaft/transition-timing-function>; den Webkrauts-Artikel zum Thema unter <http://www.webkrauts.de/2011/12/20/css-3-im-praxistest-transition/>; das Rezept 13.19 zur Präfixschreibweise; ein jQuery-Polyfill, mit dem CSS3-Übergänge auch in älteren Browsern möglich werden, unter <http://louisremi.github.com/jquery.transition.js/test/index.html>.

12.11 Alternative Cursorsymbole definieren

Problem

Wird die Maus über einen Link bewegt, soll statt des üblichen Cursors ein anderes Symbol angezeigt werden, wie in Abbildung 12-15 gezeigt.



Abbildung 12-15: Als Cursorsymbol wird eine Uhr angezeigt

Lösung

Verwenden Sie die Eigenschaft `cursor`, um den Cursor anzupassen:

```
a:link, a:visited {  
    cursor: wait;  
}
```

Diskussion

Der Eigenschaft `cursor` können mehrere verschiedene Werte zugewiesen werden, aufgelistet in Tabelle 12-5, die von allen aktuellen Browsern verstanden werden.

Tabelle 12-5: Werte für die Eigenschaft `cursor`

Wert	Beschreibung	Beispiel
auto	Die Darstellung des Cursors wird vom Browser festgelegt.	
crosshair	Zwei Linien, die sich in der Mitte kreuzen. Ein Fadenkreuz, ähnlich einem großen Pluszeichen.	
default	Plattformabhängiger Cursor, der in den meisten Browsern als Pfeil dargestellt wird; kann je nach Benutzerprogramm und Betriebssystem variieren.	
pointer	Wird verwendet, um zu illustrieren, dass sich der Mauszeiger über einem Link befindet. Wird manchmal als Hand mit ausgestrecktem Zeigefinger dargestellt, kann aber je nach Betriebssystem und Benutzerprogramm etwas anderes sein.	
move	Zeigt an, dass ein Element bewegt werden kann. Gelegentlich dargestellt als Fadenkreuz mit Pfeilspitzen an den Enden oder als gespreizte Hand.	
e-resize, ne-resize, nw-resize, n-resize, se-resize, sw-resize, s-resize, w-resize	Ein Pfeil, der anzeigt, in welche Richtung eine Seite bewegt werden kann. <code>se-resize</code> steht zum Beispiel für eine südöstliche Bewegungsrichtung, d.h. nach rechts unten.	
text	Zeigt an, dass Text ausgewählt werden kann. Entspricht dem Cursor, der oft in Textverarbeitungsprogrammen verwendet wird; ähnelt einem großgeschriebenen I.	
wait	Zeigt an, dass der Computer gerade beschäftigt ist. Gelegentlich als Stundenglas dargestellt.	
progress	Zeigt an, dass der Computer gerade beschäftigt ist, der Benutzer aber weiterhin mit dem Browser interagieren kann.	
help	Zeigt an, dass Informationen oder Hilfe zur Verfügung stehen, oftmals auf der Zielseite. Gelegentlich als Fragezeichen oder Pfeil mit einem Fragezeichen dargestellt.	
<uri>	Der Standardcursor kann durch einen extern definierten Cursor ausgetauscht werden. Das kann zum Beispiel ein Bild, eine Windows-Cursordatei, ein SVG-Cursor oder etwas anderes sein.	N/A

Siehe auch

Die CSS3-Spezifikation zur Eigenschaft `cursor` unter <http://www.w3.org/TR/css3-ui/#cursor>; Beispiele für die verschiedenen Cursor unter <http://webdesign.about.com/od/examples/l/blstylescursorexamples.htm#table>.

12.12 Modulare Skalen verwenden

Problem

Die verwendeten Schrift- und Elementgrößen sollen in einem harmonischen Verhältnis zueinander stehen.

Lösung

Verwenden Sie eine oder zwei modulare Skalen. Eine solche Skala kann auf verschiedenen Dingen basieren, beispielsweise auf der von Ihnen gewählten Schriftgröße des Haupttexts. Um hieraus weitere passende Schrift- und Elementgrößen zu errechnen, wählen Sie ein harmonisches Verhältnis wie zum Beispiel den Goldenen Schnitt (1:1,618). Anhand dieses Verhältnisses lassen sich weitere Werte berechnen, die zu Ihrem Ausgangswert passen.

Folgende Schritte sind auszuführen:

1. Finden Sie eine Schriftgröße, die zu Ihrem Inhalt und der gewählten Schriftart passt.
2. Wählen Sie ein passendes Verhältnis, z.B. den Goldenen Schnitt.
3. Multiplizieren und dividieren Sie die Schriftgröße im gewünschten Verhältnis.
4. Weitere Größen erhalten Sie durch zusätzliche Multiplikationen und Divisionen der resultierenden Schriftgrößen im gewünschten Verhältnis.



Damit Sie die Werte nicht einzeln berechnen müssen, verwenden Sie am besten ein Onlinewerkzeug wie www.modularscale.com.

Als Ausgangswert wählen wir für dieses Beispiel die Standardschriftgröße von 16 Pixeln (1em). Um den nächsthöheren Wert zu erhalten, wird der Ausgangswert mit der Zahl auf der rechten Seite der Verhältnisgleichung (1,618) multipliziert:

$$16 \text{ Pixel} \times 1,618 = 25,888 \text{ Pixel (1.618em)}$$

Den nächsthöheren Wert erhalten Sie wiederum durch Multiplikation:

$$\begin{aligned} 25,888 \text{ Pixel} \times 1,618 &= 41,887 \text{ Pixel (2.618em)} \\ 41,887 \text{ Pixel} \times 1,618 &= 67,773 \text{ Pixel (4.236em)} \\ &(\text{und so weiter...}) \end{aligned}$$

Kleinere Werte erhalten Sie entsprechend per Division:

$$16 \text{ Pixel} : 1,618 = 9,889 \text{ Pixel (0.618em)}$$

Diese Reihen können Sie beliebig in beide Richtungen fortführen. Auf diese Weise erhalten Sie mehrere mögliche Werte, die in einem harmonischen Verhältnis zueinander stehen. Diese können Sie nun in Ihrem Stylesheet verwenden, zum Beispiel:

```

p {
    font-size: 1em;
    margin: 1em;
    padding: 0.618em;
    line-height: 1.618em;
}

h1 { font-size: 2.618em; }
h2 { font-size: 1.618em; }

```

Dieses Konzept lässt sich beliebig erweitern, beispielsweise um passende Werte für die Größe eines Hauptteils, von Seitenleisten und so weiter zu ermitteln.

Hierbei kann es passieren, dass die Werte der errechneten Skala zu weit auseinanderliegen. In diesem Fall können Sie eine zweite Skala verwenden. Der Ausgangswert kann eine weitere wichtige Größe in Ihrem Layout sein, zum Beispiel die Schriftgröße, in der Überschriften am besten aussehen.

Durch die Kombination beider Skalen erhalten Sie ausreichend Werte für den Einsatz in Ihrem Design. Ein Beispiel für die Verwendung modularer Skalen sehen Sie in Abbildung 12-16.



Abbildung 12-16: Das Anwenden modularer Skalen

Diskussion

Was hat Webdesign mit Musik gemeinsam? Mehr als Sie vielleicht denken. In beiden Fällen spielt Harmonie eine Rolle. Harmonie entsteht durch das Verhältnis zweier oder mehrerer Werte zueinander – seien dies nun Tonhöhen oder Schriftgrößen.

Bestimmte Zahlenverhältnisse gelten hierbei als »besonders harmonisch«. Typische Beispiele sind der hier gezeigte Goldene Schnitt, aber auch musikalische Intervalle wie Quinten (4:3) und Quarten (5:4) sind durchaus üblich. Durch die Verwendung dieser Intervalle im Design ist es möglich, die Größe von Schriften und anderen Elementen in ein harmonisches Verhältnis zueinander zu setzen.

Anstatt ein Design auf rein technischen Vorgaben aufzubauen, bieten modulare Skalen eine Alternative, die eher von ästhetischen Gesichtspunkten ausgeht.

Hierbei stellen die errechneten Werte kein Dogma dar. Am Ende entscheidet nicht die Mathematik, sondern das Auge des Betrachters, ob ein Design schlüssig und harmonisch erscheint. Natürlich dürfen die errechneten Werte gerundet, verändert und angepasst werden.

Diese Zahlenverhältnisse finden sich in den verschiedensten Bereichen unserer Welt, vom Wachstum der Pflanzen über Musik bis hin zur Architektur. Warum also nicht auch im Webdesign?

Siehe auch

Ein Onlinewerkzeug zum Berechnen modularer Skalen unter <http://modularscale.com/>; den Artikel *More Meaningful Typography* von Tim Brown unter <http://www.alistapart.com/articles/more-meaningful-typography/>; den dazugehörigen Vortrag von Tim Brown unter <http://vimeo.com/17079380>; einen Vergleich zweier Layouts, einmal ohne und einmal mit Verwendung einer modularen Skala, unter <http://nicewebsite.com/x/38> bzw. <http://nicewebsite.com/x/39>; das Rezept 3.2 zur Angabe eigener Schriftgrößen; das Rezept 12.13 zur Definition eines Grundlinienrasters; das Buch *The Elements of Typographic Style* von Robert Bringhurst (Hartley & Marks Publishers, 1992).

12.13 Ein Grundlinienraster verwenden

Problem

Das Design einer Website soll sich an einem Grundlinienraster (auch als *vertikaler Rhythmus* bezeichnet) orientieren, wie in Abbildung 12-17 gezeigt.

Lösung

Überlegen Sie sich zunächst, auf welcher Zeilenhöhe Ihr Design basieren soll. Dieser Wert dient später als Grundlage für die Berechnung der Größe anderer Seitenelemente.

In unserem Beispiel verwenden wir als Basis eine Zeilenhöhe von 24 Pixeln. (Den Grund für die Pixelangabe erfahren Sie in der Diskussion.) Damit uns die Browser-Stylesheets keinen Strich durch die Rechnung machen, setzen wir die Innen- und Außenabstände für alle Elemente auf null:

```
* {  
  margin: 0;  
  padding: 0;  
}
```

Jetzt können wir die Zeilenhöhe und die Schriftgröße festlegen. Außerdem erhält das body-Element einen passenden Außenabstand:

```
body {  
  line-height: 24px;  
  font-size: 16px;  
  
  margin: 72px;  
}
```

Im nächsten Schritt wird festgelegt, dass Überschriften erster, zweiter und dritter Ordnung mit einer Über- und Unterstreichung versehen werden sollen. Dies realisieren wir mit den Eigenschaften border-top bzw. border-bottom:

```
h1, h2, h3 {  
  border-top: 1px solid green;  
  border-bottom: 1px solid green;  
}
```

Wir führen diese Regel mit auf, weil die Rahmenbreite für die Berechnung der Innenabstände für die Überschriften eine wichtige Rolle spielt:

```
h1 {  
  margin-bottom: 24px;  
  padding: 11px 0 11px 0;  
  
  line-height: 48px;  
  font-size: 36px;  
}
```

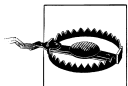
Da wir am Anfang des Stylesheets sämtliche Innen- und Außenabstände auf null gesetzt haben, müssen wir sie bei Bedarf ausdrücklich angeben. Damit das Grundlinienraster gewahrt bleibt, erhält margin-bottom den gleichen Wert wie die eingangs behandelte Eigenschaft line-height für das body-Element. Bei der Berechnung des oberen und unteren Innenabstands muss berücksichtigt werden, dass wir zuvor Rahmenlinien von je 1 Pixel Breite definiert haben, die hier vom oberen und unteren Innenabstand subtrahiert werden müssen. Um genügend Platz für die Überschrift zu haben, verdoppeln wir außerdem die Zeilenhöhe.



Abbildung 12-17: Die Dimensionen der Seitenelemente orientieren sich an einem Grundlinienraster.

Als Wert für `font-size` wählen wir das Anderthalbfache unserer Berechnungsbasis (bzw. $\frac{2}{3}$ der Zeilenhöhe für `h1`-Überschriften). Im Prinzip können Sie den Wert für `font-size` frei wählen. Das Raster bleibt selbst dann noch intakt, wenn Sie eine extrem hohe Schriftgröße wählen. In diesem Fall würden die Buchstaben einfach über die oberen und unteren Rahmenlinien hinausragen. Ob das sinnvoll oder schön ist, müssen Sie selbst entscheiden.

Bei den `h2`-Überschriften gehen wir etwas anders vor. Hier verzichten wir auf den Innenabstand. Daher müssen die Rahmenlinien von der Zeilenhöhe subtrahiert werden.



Diese »Abkürzung« funktioniert nur bei einzeliligen Überschriften. Wird die Überschrift umbrochen, fehlen pro Zeile zwei Pixel.

```
h2 {
  font-size: 24px;
  line-height: 46px;
  margin-bottom: 24px;
}
```

Bei Überschriften dritter Ordnung verwenden wir die Standardzeilenhöhe. Die Schriftgröße wird vom body-Element geerbt. Ohne diese Angabe würde hier der Wert des Browser-Stylesheets verwendet werden. Ähnlich wie bei h1-Überschriften passen wir den Abstand zwischen den beiden Rahmenlinien anhand von padding an:

```
h3 {  
    font-size: inherit;  
    padding: 11px 0 11px 0;  
}
```

Damit einzelne Absätze visuell voneinander getrennt werden, erhalten sie einen unteren Außenabstand, der der Standardzeilenhöhe entspricht:

```
p { margin-bottom: 24px; }
```

Die Darstellung von Listen ist etwas trickreicher. Hier verteilen wir den einer Zeile entsprechenden Platz gleichmäßig auf den oberen und den unteren Außenabstand. Als Folge stehen die einzelnen Listeneinträge jetzt »zwischen« den Zeilen. Trotzdem läuft der Rhythmus nach der Liste gleichmäßig weiter. Die Schrift in den Listeneinträgen soll etwas größer sein als die Standardschriftgröße. Wir haben uns hier für 2/3 des Standardzeilenabstands entschieden:

```
ul {  
    margin-top: 12px;  
    margin-bottom: 12px;  
}  
  
ul li {  
    font-size: 18px;  
}
```

Für das »hängende« V unterhalb der Überschrift zweiter Ordnung haben wir die Schriftgröße absichtlich größer gewählt (96px) als die Zeilenhöhe (72px). Dadurch, dass der Buchstabe über das Raster hinausragt, wird das ansonsten recht starre Design etwas aufgelockert. Für die Platzierung haben wir uns der Technik aus Rezept 3.18 bedient.

Die Abbildung vom CSS-Kochbuch wird zusammen mit seiner Beschriftung mit einem div-Container umgeben. Dieser wird als links angeordneter Float definiert und mit den passenden Außenabständen versehen:

```
.float_links {  
    float: left;  
    margin: 0 24px 24px 0;  
}
```

Damit sich das Bild ebenfalls in den vertikalen Rhythmus einfügt, müssen wir seine Größe anpassen. Der Rahmen soll das Bild in einem Abstand von 3 Pixeln umgeben und 1 Pixel breit sein. Diese Angaben müssen wir von der Höhe des Bilds subtrahieren. Hat das Bild ursprünglich eine Höhe von 240 Pixeln, müssen wir diese auf 232 Pixel reduzieren, damit der Rhythmus wieder stimmt.

Standardmäßig werden Bilder als (ersetzte) Inline-Elemente dargestellt. Das heißt, sie besitzen wie normaler Text eine Grundlinie. Ohne weitere Deklarationen würde hier-

durch die Beschriftung zu weit nach unten rutschen. Wir schaffen das Problem aus der Welt, indem wir das Bild am unteren Rand seines Elternelements (des Container-div) ausrichten:

```
img {  
  padding: 3px;  
  border: 1px solid #ddd;  
  height: 232px;  
  
  vertical-align: bottom;  
}
```

Für die Beschriftung müssen keine Änderungen vorgenommen werden, daher machen wir direkt mit dem Lockzitat rechts unten weiter. Im HTML-Code sieht das so aus:

```
<blockquote><span>I can see, the planets are aligning for me<br /> -  
  but I can not read them...</span>  
<p>Kate Rusby: Planets</p>  
</blockquote>
```

Das Zitat selbst soll auf drei Zeilenhöhen (36 Pixel) verteilt werden. Die Quellenangabe steht auf einer eigenen Zeile. Damit das Zitat auch »lockt«, wählen wir hier eine relativ große Schrift, die 5/4 der Standardzeilenhöhe entspricht, und eine kursive Schriftvariante:

```
blockquote span {  
  font-size: 30px;  
  line-height: 36px;  
  font-style: italic;  
}
```

Die übrigen Anweisungen wiederholen im Prinzip nur das bereits Gesagte. Daher zeigen wir Ihnen zum Abschluss nur noch die Definitionen für den Fußteil. Wie die Überschriften erhält er ober- und unterhalb eine 1 Pixel breite Rahmenlinie. Ähnlich wie bei der Definition für die Listen verteilen wir hier wieder eine Zeilenhöhe gleichmäßig auf den oberen und unteren Innenabstand, um dem Text vertikal etwas Platz zu verschaffen. Auch hier müssen wir die Breite der Rahmenlinien von den Werten des Innenabstands subtrahieren:

```
#footer {  
  border-top: 1px solid green;  
  border-bottom: 1px solid green;  
  
  padding: 11px;  
}
```

Diskussion

Ähnlich wie modulare Skalen (siehe Rezept 12.12) können Grundlinienraster mehr Harmonie in das Design einer Seite bringen. Normalerweise kennt man dieses Verfahren eher aus Desktop-Publishing-Programmen, in denen es eine gängige Methode darstellt, um Inhalte gleichmäßig zu platzieren.

Um den Sachverhalt zu verdeutlichen, haben wir für dieses Beispiel Pixel als Maßeinheit verwendet. Zwar ist es auch möglich, beispielsweise em-Einheiten zu benutzen, dabei kann es jedoch schnell zu Rundungsfehlern kommen, zum Beispiel wenn die Breite eines möglichst dünnen Rahmens von Innenabstand oder Zeilenhöhe subtrahiert werden muss.

Speziell die Zeilenhöhe kann auch ganz ohne Maßeinheit angegeben werden. Der Wert gilt dann als Skalierungsfaktor. Dieser bezieht sich allerdings auf die definierte Schriftgröße des Elements und funktioniert daher nicht wie gewünscht.



Unter Umständen kann es hilfreich sein, das Boxmodell per `box-sizing: content-box`; anzupassen, um die Berechnungen leichter zu machen. Weitere Informationen finden Sie in Rezept 13.8.

Dieses Rezept basiert auf dem Artikel *Setting Type on the Web to a Baseline Grid* von Wilson Miner, veröffentlicht auf der Website *A List Apart* (Link im Abschnitt »Siehe auch«).

Siehe auch

Den Artikel, der als Basis für dieses Rezept dient, unter <http://www.alistapart.com/articles/settingtypeontheweb/>; den Artikel *Compose to a Vertical Rhythm* von Richard Rutter unter <http://24ways.org/2006/compose-to-a-vertical-rhythm>; den Artikel *Unitless Line Heights* von Eric Meyer unter <http://meyerweb.com/eric/thoughts/2006/02/08/unitless-line-heights/>; den Wikipedia-Eintrag zur Definition eines Grundlinienrasters unter <http://de.wikipedia.org/wiki/Grundlinienraster>.

12.14 Quasi zufällige Hintergrundmuster erzeugen

Problem

Sie möchten Hintergrundmuster erzeugen, die sich scheinbar nicht wiederholen.

Lösung

Verwenden Sie das *Zikaden-Prinzip*. Definieren Sie hierfür eine Reihe unterschiedlich breiter Hintergrundbilder, die einen transparenten Bereich enthalten. Die Breite der Bilder muss jeweils einer Primzahl entsprechen. In unserem Beispiel haben wir drei Grafiken mit einer Breite von 13, 23 und 31 Pixeln erstellt (siehe Abbildung 12-18).

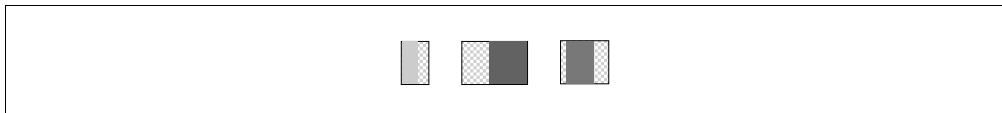


Abbildung 12-18: Die drei Grafiken für das scheinbar zufällige Hintergrundmuster

Diese drei Grafiken werden nun als Hintergrundbilder für das body-Element der Seite definiert. Um den gewünschten Effekt zu erzielen, wird das schmalste Bild zuerst angegeben, das breiteste zuletzt:

```
body {  
    padding: 0;  
    margin: 0;  
    height: 100%;  
  
    background-image: url('bilder/cicada/13x20.png'),url('bilder/cicada/23x20.png'),  
    url('bilder/cicada/31x20.png');  
}
```

Abbildung 12-19 zeigt, wie sich die Überlagerung der verschiedenen breiten Bilder auf den Hintergrund auswirkt. Je mehr Bilder wir hinzufügen, desto seltener wiederholt sich das Hintergrundmuster.



Abbildung 12-19: Die drei Hintergrundbilder werden nacheinander in die Seite eingefügt, und mit jeder Grafik wird das Muster komplexer

Wenn Sie wissen möchten, wann sich das Muster zum ersten Mal wiederholt, brauchen Sie nur die Bildbreiten miteinander zu multiplizieren:

13 Pixel x 23 Pixel x 31 Pixel = 9269 Pixel

Die drei Grafiken sind so klein, dass Sie sie sogar als Daten-URI direkt in das Stylesheet einbetten könnten. Gemeinsam bringen sie nicht einmal ein KByte auf die Waage.

Diskussion

Im April 2011 stellte der Webdesigner und Frontend-Entwickler Alex Walker eine neue Technik vor, mit der man mit äußerst geringen Dateigrößen Hintergrundmuster erzeugen kann, die sich erst nach sehr langer Zeit wiederholen. Seine Methode nennt er das *Zikaden-Prinzip*.

Walker ließ sich von der Natur inspirieren: Wissenschaftler haben herausgefunden, dass bestimmte Zikaden einen Lebenszyklus von 7, 13 oder 17 Jahren haben. Alle drei Werte sind Primzahlen. Dadurch kommt es nur äußerst selten zu Überschneidungen mit den Lebenszyklen der Fressfeinde. Hat ein Feind beispielsweise einen Zyklus von 3 Jahren, würde es nur alle 51 Jahre zu einer Überschneidung kommen.

Dieses Prinzip können wir uns im Webdesign zunutze machen, um Hintergründe zu erzeugen, die sich nur sehr selten wiederholen. Die unterschiedlichen Muster entstehen, wenn wir Hintergrundbilder verschiedener Breite miteinander überlagern. Je breiter die einzelnen Bilder sind, desto später wiederholt sich das Muster. Wichtig ist nur, dass die Breite jedes Bilds einer Primzahl entspricht.

Wie in Rezept 2.12 erklärt wird, werden die Grafiken nicht in der Reihenfolge ihrer Definition überlagert, sondern umgekehrt. Das zuletzt definierte Bild liegt also am weitesten unten. Das heißt, zumindest in den beiden zuerst definierten Grafiken sollten ausreichend transparente Bereiche vorhanden sein, durch die die anderen Bilder sichtbar sind.

Der Webentwickler Randy Merrill erweitert dieses Prinzip. Anstelle von Grafiken verwendet er eine Reihe von Farbverläufen (siehe Rezept 12.4), die als Hintergrundbilder definiert werden. Hier ein Beispiel:

```
body {
  background-color: rgba(189,198,102,.22);
  background-image:
    -webkit-linear-gradient(0, rgba(239,255,210,.13) 50%, transparent 50%),
    -webkit-linear-gradient(0, rgba(189,198,102,.19) 50%, transparent 50%),
    -webkit-linear-gradient(0, transparent 50%, rgba(35,139,124,.29) 50%),
    -webkit-linear-gradient(0, transparent 50%, rgba(66,77,69,.43) 50%);
  background-image:
    -moz-linear-gradient(0, rgba(239,255,210,.13) 50%, transparent 50%),
    -moz-linear-gradient(0, rgba(189,198,102,.19) 50%, transparent 50%),
    -moz-linear-gradient(0, transparent 50%, rgba(35,139,124,.29) 50%),
    -moz-linear-gradient(0, transparent 50%, rgba(66,77,69,.43) 50%);
  background-image:
    -o-linear-gradient(0, rgba(239,255,210,.13) 50%, transparent 50%),
    -o-linear-gradient(0, rgba(189,198,102,.19) 50%, transparent 50%),
    -o-linear-gradient(0, transparent 50%, rgba(35,139,124,.29) 50%),
    -o-linear-gradient(0, transparent 50%, rgba(66,77,69,.43) 50%);
  background-image:
    linear-gradient(0, rgba(239,255,210,.13) 50%, transparent 50%),
    linear-gradient(0, rgba(189,198,102,.19) 50%, transparent 50%),
    linear-gradient(0, transparent 50%, rgba(35,139,124,.29) 50%),
    linear-gradient(0, transparent 50%, rgba(66,77,69,.43) 50%);
}
```

Bei dieser Methode ist es wichtig, die Breite der einzelnen Hintergrundbilder per `background-size` anzugeben, um den gewünschten Effekt zu erzielen. Da die Hintergründe vertikal und horizontal gekachelt werden, reicht es, pro Bild einen Wert anzugeben, der für die Breite und die Höhe gleichermaßen benutzt wird.

In diesem Beispiel wurden sogar vier Hintergründe definiert. Multipliziert man die Breiten, kommt man auf 8.1940.933 Pixel, nach denen sich das Muster wiederholt. Selbst auf einem Retina-Display mit einer Pixeldichte von 326 Pixel/Zoll entspräche dies noch einer Breite von 251.352 Zoll oder etwa sechs Kilometern. Das sollte auch für den größten Bildschirm ausreichen.

Das hier gezeigte Beispiel ist nur der Anfang. Mittlerweile hat sich ein regelrechter Wettbewerb dazu entwickelt, wer die besten »Zikaden-Hintergründe« erstellt (einen Link finden Sie im Abschnitt »Siehe auch«).

Mehrfache Hintergründe werden von allen aktuellen Browsern unterstützt. Für Internet Explorer 7 und 8 können Sie das Polyfill CSS3 *PIE* (siehe Rezept 13.18) einsetzen. Leider enthält es (noch) keine Unterstützung für die Definition von `background-size`. Daher werden Sie hier bis auf Weiteres auf grafische Hintergründe zurückgreifen müssen.

Die Eigenschaft `background-size` wird von den meisten aktuellen Browsern verstanden. Sollen auch Firefox 3.6, Opera 10.0 und 10.01 unterstützt werden, müssen Sie hier die entsprechenden Präfixschreibweisen angeben.

Die Definition von Farbverläufen mit der oben gezeigten `*-linear-gradient()`-Funktion wird noch nicht von allen Browsern unterstützt. Wo es funktioniert, müssen Sie aktuell auf jeden Fall noch die Präfixschreibweise einsetzen.

Siehe auch

Den ursprünglichen Artikel von Alex Walker unter <http://designfestival.com/the-cicada-principle-and-why-it-matters-to-web-designers/>; den Artikel *Cicada Principle with CSS* von Randy Merrill unter <http://forthewebdeveloper.com/2011/cicada-principle-with-css3/>; eine Tabelle mit Primzahlen unter [http://de.wikibooks.org/wiki/Primzahlen:_Tabelle_der_Primzahlen_\(2_-_100.000\)](http://de.wikibooks.org/wiki/Primzahlen:_Tabelle_der_Primzahlen_(2_-_100.000)); den Wikipedia-Eintrag zu Singzikaden unter <http://de.wikipedia.org/wiki/Singzikaden>; das Rezept 2.12 zur Definition mehrfacher Hintergrundbilder; das Rezept 12.4 zur Definition von Farbverläufen mit CSS3; das Rezept 13.20 zur Definition von Daten-URIs; das Rezept 2.13 zur Eigenschaft `background-size`; eine Galerie verschiedener »Zikaden-Hintergründe« unter <http://designfestival.com/cicada/>.

Tipps und Tricks

Einführung

In den letzten zehn Jahren wurde viel Zeit darauf verwendet, nicht standardkonforme Browser dazu zu bringen, ein Design wie gewünscht darzustellen. Glücklicherweise lag der Marktanteil des IE6 beim Schreiben dieses Buchs in Deutschland bereits unter einem Prozent. Das heißt, solange Sie keine Website für den asiatischen Markt entwickeln (siehe <http://www.ie6countdown.com/>), können Sie endlich auf einen Großteil der Hacks und Workarounds verzichten, die Webdesignern lange Zeit das Leben schwer gemacht haben.

Mittlerweile unterstützen alle aktuellen Browser zumindest den CSS 2.1-Standard (mit ein paar kleinen Ausnahmen). Und obwohl die meisten Probleme in den neueren Browserversionen gelöst sind, kommt es vor, dass man als Webdesigner in die Trickkiste greifen muss, damit eine Website möglichst konsistent dargestellt wird.

Die Rezepte in diesem Kapitel sollen Ihnen helfen, robuste und trotzdem flexible Style-sheets zu schreiben. Sie finden hier eine Reihe von Lösungen, die Ihnen die tägliche Arbeit mit CSS erleichtern sollen.

Zu Beginn befassen wir uns mit der Suche nach und Vermeidung von Fehlern sowie mit den verschiedenen Möglichkeiten, Ihr Design mit möglichst vielen Browsern und Rechnerplattformen zu testen.

Der zweite Teil zeigt eine Auswahl an CSS-Tricks, die Ihnen die tägliche Arbeit mit Style-sheets erleichtern sollen. Hier finden Sie Informationen darüber, wie Sie per CSS Inhalte in die Darstellung einfügen können, das Boxmodell anpassen und Float-Container dazu bringen, sich an ihre Höhe zu erinnern.

Wir zeigen Ihnen, wie Sie Regeln nur für bestimmte Versionen des Internet Explorer schreiben können, ohne dafür auf irgendwelche Hacks zurückgreifen zu müssen, wie Sie HTML5-Elemente auch in älteren Browsern nutzen können, und wir gehen auf Probleme ein, die im Internet Explorer im Zusammenhang mit dem Konzept des *hasLayout* auftreten können.

Im letzten Teil dieses Kapitels geht es darum, die Möglichkeiten von CSS mit Präprozessoren und Polyfills zu erweitern, sowie um die korrekte Verwendung der Präfixschreib-

weise für CSS3-Eigenschaften (und -werte). Zum Abschluss zeigen wir Ihnen, wie Sie Bilder und Schriften direkt ins Stylesheet einbetten können.

13.1 Überprüfen, ob ein Stylesheet gültig ist

Problem

Sie möchten wissen, ob Ihre Stylesheet-Regeln der CSS 2.1- bzw. CSS3-Spezifikation entsprechen, um Syntax- und andere Fehler von vornherein ausschließen zu können.

Lösung

Zunächst sollten Sie Ihre CSS-Regeln prinzipiell validieren. Das gilt auch, wenn das Stylesheet scheinbar korrekt funktioniert. Das World Wide Web Consortium stellt extra zu diesem Zweck einen Online-Validator zur Verfügung.

Rufen Sie hierfür die Adresse <http://jigsaw.w3.org/css-validator/> mit einem Browser Ihrer Wahl auf. Nach dem Laden des Validators haben Sie drei Möglichkeiten, Ihr Stylesheet überprüfen zu lassen: per Angabe eines URI, durch Hochladen der CSS-Datei oder per direkter Eingabe in ein entsprechendes Formular (siehe Abbildung 13-1).

Abbildung 13-1: Der URI der zu überprüfenden Seite wird in das Onlineformular eingegeben

Nach einer kurzen Überprüfung erhalten Sie eine detaillierte Rückmeldung über mögliche Probleme mit den CSS-Regeln der angegebenen Seite (siehe Abbildung 13-2).



Abbildung 13-2: Das Ergebnis der Überprüfung mit dem W3C-Validator



Seit einiger Zeit bietet das W3C auch einen einheitlichen Validator namens *Unicorn*, um die Überprüfung von HTML, CSS und andere benutzerdefinierte Dinge auf einmal durchführen lassen.

Diskussion

Schnell kann es passieren, dass man den Namen einer Eigenschaft falsch schreibt, ein Semikolon am Ende einer CSS-Regel vergisst oder eine Maßeinheit nicht angibt. Auch kommt es vor, dass der verwendete Texteditor versehentlich ein »unsichtbares« Leerzeichen in Ihr Stylesheet einfügt, das Browser davon abhält, eine offensichtlich korrekte Regel zu interpretieren.

Für Fälle wie diese gibt es den CSS-Validator, der überprüft, ob Ihr Code den aktuellen Spezifikationen entspricht. Hierbei können Sie selbst wählen, nach welchem Standard die Überprüfung vorgenommen werden soll. Standardmäßig wird davon ausgegangen, dass Ihr Stylesheet nach dem CSS-Standard in der Version 2.1 verfasst wurde.



Natürlich sollten Sie nicht nur Ihre Stylesheets validieren, sondern auch Ihren HTML-Code!

Die Überprüfung von CSS3-Eigenschaften ist ebenfalls möglich. Zwar wird diese Stufe im Moment noch ausdrücklich als experimentell bezeichnet, sie funktioniert aber schon weitgehend fehlerfrei. Die Verwendung von Präfixeigenschaften (siehe Rezept 13.9) wird ebenfalls unterstützt. Allerdings kann es bei bestimmten Präfixwerten (z.B. `-webkit-linear-gradient()` usw.) zu Problemen kommen.

Sollte Ihr Stylesheet dagegen CSS3-Eigenschaften enthalten, die nicht den Spezifikationen entsprechen (wie beispielsweise die proprietären Microsoft-Eigenschaften `filter` und `behaviour`), müssen Sie diese vor der Validierung auskommentieren oder in ein externes Stylesheet auslagern und dessen Einbindung für die Überprüfung abschalten.

Siehe auch

Den CSS-Validator des World Wide Web Consortium unter <http://jigsaw.w3.org/css-validator/>; das Rezept 13.19 zur Verwendung von Präfixeigenschaften; den HTML-Validator des W3C unter <http://validator.w3.org/>; den einheitlichen Validator des W3C zum Überprüfen von HTML, CSS und anderen Dingen in einem Durchgang unter <http://validator.w3.org/unicorn/>.

13.2 Ein Design auf mehreren Plattformen testen

Problem

Sie möchten Ihr Webdesign auf möglichst vielen Browser- und Systemplattformen testen.

Lösung 1

Die aktuellen Browserversionen können Sie sich meist von den Webseiten der Hersteller herunterladen. Ältere Browserversionen finden Sie beispielsweise auf der Website von *Oldapps.com*.

Für den Internet Explorer gibt es außerdem das Werkzeug *IETester*, mit dem Sie aus einem Programm heraus mehrere Versionen des IE testen können. Die Links finden Sie im Abschnitt »Siehe auch«.

Lösung 2

Verwenden Sie eine Screenshot-Website wie beispielsweise *Browsercam* oder *Browsershots*. Hier können Sie Ihre Website mit einer Vielzahl von Kombinationen aus Betriebssystem und Browser aufrufen und automatisch Bildschirmfotos davon anfertigen lassen, wie in Abbildung 13-3 zu sehen.

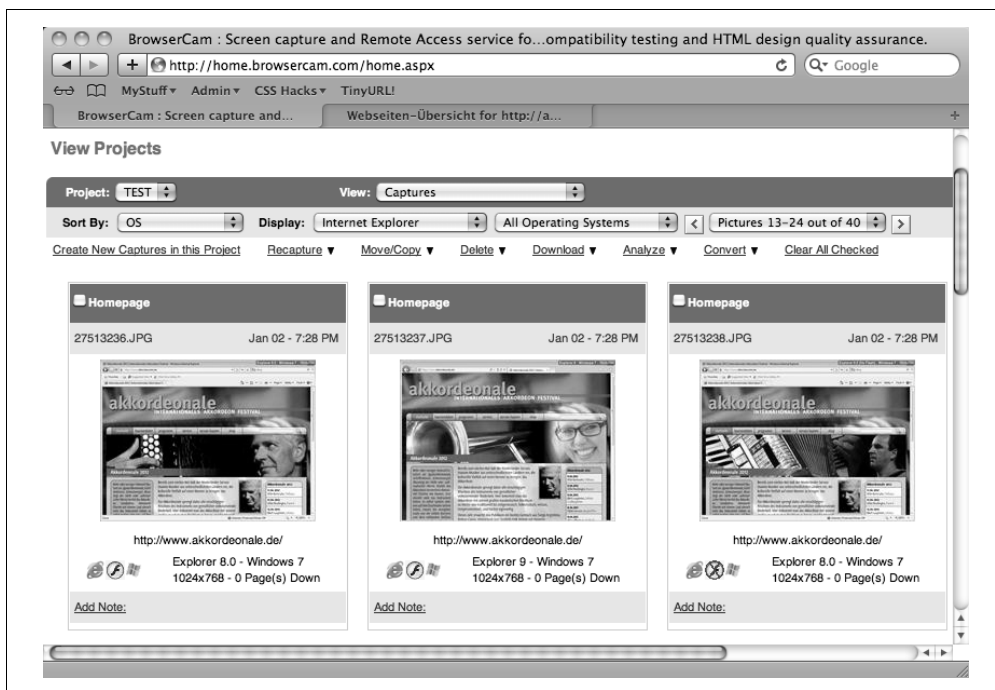


Abbildung 13-3: Eine Reihe von Browserbildschirmfotos, hier von Browsercam.com

Während *Browsercam.com* kommerziell arbeitet, wird *Browsershots.org* von Freiwilligen betrieben und ist nicht profitorientiert. Beide Websites bieten die Möglichkeit, gegen Bezahlung zusätzliche Leistungen und Dienste in Anspruch zu nehmen.

Diskussion

Nur in den wenigsten Fällen wird es möglich sein, Ihr Webdesign tatsächlich mit allen Kombinationen aus Betriebssystem und Browser zu testen. Daher ist es sinnvoll, vor der Entwicklung einer Website genau festzulegen, welche Browser welche Merkmale Ihrer Seiten unterstützen sollen.

Hierbei hat sich das Prinzip des »Graded Browser Support« durchgesetzt. Aktuelle und häufig benutzte Browser werden dabei der höchsten Kategorie zugeordnet. Hier sollten alle Merkmale der Website einwandfrei und konsistent funktionieren.

Es folgen Browser, die entweder schon etwas betagter sind oder weniger Nutzer haben. Auch hier sollten sämtliche Funktionen und die Benutzbarkeit der Website nicht beschränkt sein. Allerdings kann auf einige Elemente des Progressive Enhancement verzichtet werden. Es ist nicht unbedingt nötig, dass eine Website in absolut allen Browsern mit abgerundeten Ecken, Animationen und Schattenwürfen angezeigt wird.

Am Ende der Skala befinden sich die Browser, die zwar noch benutzt werden, aber kaum noch von Bedeutung sind (z.B. IE7). Hier sollte die Website ebenfalls noch brauchbar sein – notfalls auch ohne Stylesheets.



Die tatsächliche Wahl der Browserunterstützung hängt auch davon ab, für welche Gegend der Welt Sie die Website entwickeln. Liegt Ihre primäre Zielgruppe in China oder Südkorea, werden Sie um den Internet Explorer vermutlich nicht herumkommen.

Letztendlich sollte sich die Einteilung danach richten, welche Browser die Besucher Ihrer Website benutzen. Ein Blick in die Log-Dateien Ihres Webserver oder Dienste wie Google Analytics können hierbei helfen.

Selbst wenn Sie nur die gängigsten Browser und Betriebssysteme testen wollen, kommen Sie schnell an die Grenzen des Machbaren. Hier können Screenshot-Websites helfen, die Lücke zu füllen. Zumindest lässt sich so ein Eindruck davon gewinnen, wie die Darstellung unter Umständen aussieht.

Siehe auch

Die Website von *IETester* unter <http://www.my-debugbar.com/wiki/IETester/HomePage>; die Website für verschiedene Versionen älterer Browser unter http://www.oldapps.com/internet_explorer.php (IE), <http://www.oldapps.com/firefox.php> (Firefox) und <http://mac.oldapps.com/safari.php> (Safari/Mac); die Website *Browserscam.com* zum Anfertigen von Screenshots verschiedener Browser; die kostenlose Browsershots-Website unter <http://browsershots.org>; die (recht konservative) YUI-Anleitung zum »Graded Browser Support« unter <http://yuilibrary.com/yui/docs/tutorials/gbs/>; die Einteilung der von jQuery unterstützten Browser unter <http://jquerymobile.com/gbs/>; den englischen Wikipedia-Artikel zum Progressive Enhancement unter http://en.wikipedia.org/wiki/Progressive_enhancement.

13.3 Browsererweiterungen zur Fehlersuche benutzen

Problem

Sie möchten wissen, welche Browsererweiterungen es für die Fehlersuche gibt.

Lösung

Die meisten gängigen Browser bringen inzwischen eigene Entwicklerwerkzeuge mit, die teilweise aber noch aktiviert werden müssen. Mit diesen Programmiererweiterungen können Sie den HTML-Quellcode, Stylesheets, das DOM, JavaScript-Programme, Ladezeiten und andere Dinge inspizieren. Teilweise lassen sich Stildefinitionen deaktivieren oder zeitweilig verändern. (Die Reihenfolge der im Folgenden aufgeführten Browser stellt keine Wertung dar.)

Firefox

Die mit Abstand beliebtesten Firefox-Erweiterungen für die Webentwicklung sind die *Web Developer Toolbar* und das Werkzeug *Firebug*. Beide zusammen ermöglichen umfangreiche Inspektionen und Veränderungen nicht nur am CSS-Code. *Firebug* besitzt außerdem die Besonderheit, dass es selbst erweiterbar ist.

Google Chrome

In Googles Browser finden Sie die Entwicklerwerkzeuge unter dem Menübefehl *Darstellung* → *Entwickler* (Mac), bzw. *Tools* → *Entwicklertools* (Windows).

Internet Explorer

Im Internet Explorer können Sie die integrierten Entwicklerwerkzeuge unter *Extras* → *Entwicklerwerkzeuge* oder durch Drücken der Taste *F12* aktivieren. Zusätzlich gibt es die Browsererweiterung *DebugBar*, die so ähnlich funktioniert wie die bordeigenen Werkzeuge. Sie eignet sich vor allem für die Verwendung mit älteren Versionen des IE oder auch mit dem Programm *IETester* (siehe Rezept 13.2) aus gleichem Hause.

Opera

Auch der norwegische Browserhersteller Opera stattet sein Programm mit Werkzeugen aus, die dem Webautor bei der Erstellung und Optimierung seiner Seiten helfen soll. Hier heißt das Werkzeug *Dragonfly* und kann über den Menübefehl *Extras* → *Weiteres* → *Opera Dragonfly* aufgerufen werden.

Safari

In Safari müssen die Entwicklerwerkzeuge zunächst in den Voreinstellungen aktiviert werden. Nach einem Klick auf den Karteireiter *Erweitert* finden Sie das nötige Ankreuzfeld zum Aktivieren. Danach stehen die Werkzeuge unter dem Menüeintrag *Entwickler* zur Verfügung.

Mobile Browser

Im iPhone gibt es immerhin rudimentäre Entwicklerwerkzeuge. Deutlich besser geeignet sind aber die verschiedenen Entwicklungsumgebungen für Mobilgeräte, die von den jeweiligen Herstellern zur Verfügung gestellt werden. Links zu den SDKs (inklusive Simulatorprogrammen) für iPhone und Android finden Sie im Abschnitt »Siehe auch«.

Diskussion

Oftmals werden die Fehler bei der Erstellung von Stylesheets erst sichtbar, wenn man sich die Darstellung in möglichst vielen Browsern ansieht. Dabei kann es eine große Hilfe sein, wenn man den Code direkt im Browser untersuchen und gegebenenfalls testweise ändern kann.

Die hier vorgestellten browsereigenen Werkzeuge und Erweiterungen stellen eine enorme Hilfe beim Testen und Optimieren von Stylesheets (und zu diversen anderen Aspekten der Webentwicklung) dar. Anstatt durch Versuch und Irrtum mehr oder weniger im Trüben zu fischen, können Sie mit diesen Mitteln gezielt bestimmte Elemente hervorheben, deaktivieren und verändern.

Siehe auch

Die Website für die *DebugBar* für den Internet Explorer unter <http://www.debugbar.com/>; die Firefox-Erweiterung *Web Developer* unter <https://addons.mozilla.org/de/firefox/addon/web-developer/>; die Firefox-Erweiterung *Firebug* unter <https://addons.mozilla.org/de/firefox/addon/firebug/> bzw. <http://getfirebug.com>; einen Überblick über die kommenden Fähigkeiten der Entwicklerwerkzeuge in Firefox 12 unter <http://www.webmonkey.com/2011/12/an-overview-of-firefoxs-coming-developer-tools/>; die iPhone-Entwicklerwerkzeuge unter <https://developer.apple.com/devcenter/ios/index.action>; die Entwicklerwerkzeuge für Android unter <http://developer.android.com/index.html>.

13.4 Bekannte Browser-Bugs und CSS-Probleme erkennen und beheben

Problem

Sie möchten wissen, welcher Browser ab welcher Version eine bestimmte CSS-Eigenschaft unterstützt. Außerdem wollen Sie herausfinden, welche Probleme bestimmte Browser mit der Interpretation von CSS haben und wie diese erkannt und bestenfalls gelöst werden können.

Lösung

Schauen Sie in die Tabellen auf den einschlägigen Websites (die Links finden Sie im Abschnitt »Siehe auch«), um sich die nötigen Informationen zu beschaffen. Wenn Sie beispielsweise wissen möchten, welche Browser die Eigenschaft `border-radius` bereits ohne Präfix unterstützen, können Sie sich auf der Website *Can I Use* informieren, wie in Abbildung 13-4 gezeigt.

Diskussion

Für die Unterstützung von 2.1-Eigenschaften, Selektoren und Werten eignet sich die Website quirksmode.org. Hier finden Sie Tabellen, in denen die Unterstützung für diverse Browser und Browserversionen detailliert aufgeschlüsselt wird.

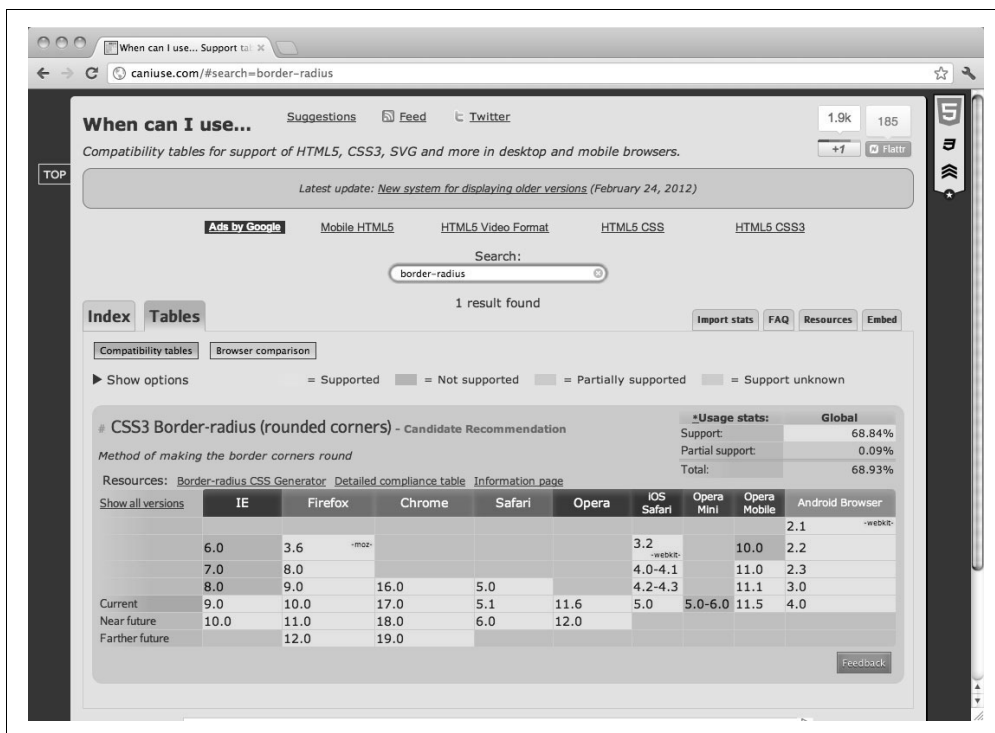


Abbildung 13-4: Ein Blick auf die Website von Can I Use zeigt, dass die Eigenschaft border-radius von vielen Browsern ohne Präfix unterstützt wird

Neben diesen Kompatibilitätstabellen enthält die Quirksmode-Website noch eine Reihe weiterer Ressourcen mit Tests und Dokumentationen zu verschiedenen Browserproblemen, insbesondere in älteren Versionen des Internet Explorer.

Gerade über Unterstützung für CSS3-Eigenschaften und -Werte sind in letzter Zeit einige Websites entstanden. Eine Auswahl finden Sie im folgenden Abschnitt und in Anhang A, Ressourcen.

Siehe auch

Tabellen zur Unterstützung von CSS3 und HTML5 unter caniuse.com; die QuirksMode-Website unter quirksmode.org; die Site Can I Use unter <http://www.caniuse.com>; die Website *Browsersupport* mit einer umfassenden Liste von CSS-Eigenschaften und -Werten sowie Informationen zu deren Unterstützung unter <http://www.browsersupport.net/CSS/>; den englischen Wikipedia-Artikel mit einem Vergleich von Layout-Engines, die in den verschiedenen Browsern für die Darstellung der Seiten sorgen, unter [http://en.wikipedia.org/wiki/Comparison_of_layout_engines_\(Cascading_Style_Sheets\)](http://en.wikipedia.org/wiki/Comparison_of_layout_engines_(Cascading_Style_Sheets)).

13.5 Websites mit einem Textbrowser testen

Problem

Eine Website soll mit einem textbasierten Browser getestet werden.

Lösung

Verwenden Sie ein Onlinewerkzeug wie zum Beispiel *Lynx Viewer*, das einen Textbrowser nachahmt (siehe <http://www.delorie.com/web/lynxview.html>).

Damit das Werkzeug korrekt funktioniert, müssen Sie im Wurzelverzeichnis Ihres Web-servers eine Datei mit dem Namen *delorie.htm* anlegen. Damit teilen Sie der Applikation Ihr Einverständnis zur Abfrage und Darstellung Ihrer Site mit. Danach können Sie die URL Ihrer Site eingeben und die Ergebnisse online betrachten, wie in Abbildung 13-5 gezeigt.

Diskussion

Das Testen mit einem Textbrowser ist in mehrfacher Hinsicht sinnvoll. Sie können damit die Website sehen wie eine Suchmaschine. Die verschiedenen Elemente erscheinen in der Reihenfolge ihrer Definition im HTML-Dokument. Auf diese Weise können Sie feststellen, ob Ihre Inhalte richtig strukturiert sind.

Besonders wenn die Zugänglichkeit Ihrer Seiten wichtig ist, sollte das Testen mit einem Textbrowser zum Standardrepertoire gehören.

Lynx ist ein schon recht betagter Webbrowser. Anstatt ein webbasiertes Emulator-Programm einzusetzen, können Sie Lynx auch auf Ihren PC herunterladen und installieren. Nähere Informationen finden Sie unter <http://www.subir.com/lynx/build.html>. Unter <http://www.osxgnu.org/> gibt es außerdem eine fertig kompilierte Version von Lynx für den Mac.

Siehe auch

Die Website des *Lynx Viewer* unter <http://www.delorie.com/web/lynxview.html>; die Website des Lynx-Browsers unter <http://lynx.isc.org/>.

13.6 Die richtige Zeichenkodierung angeben

Problem

Sie möchten wissen, welche Textkodierung Sie für Ihre Stylesheets (und HTML-Dateien) verwenden sollen.

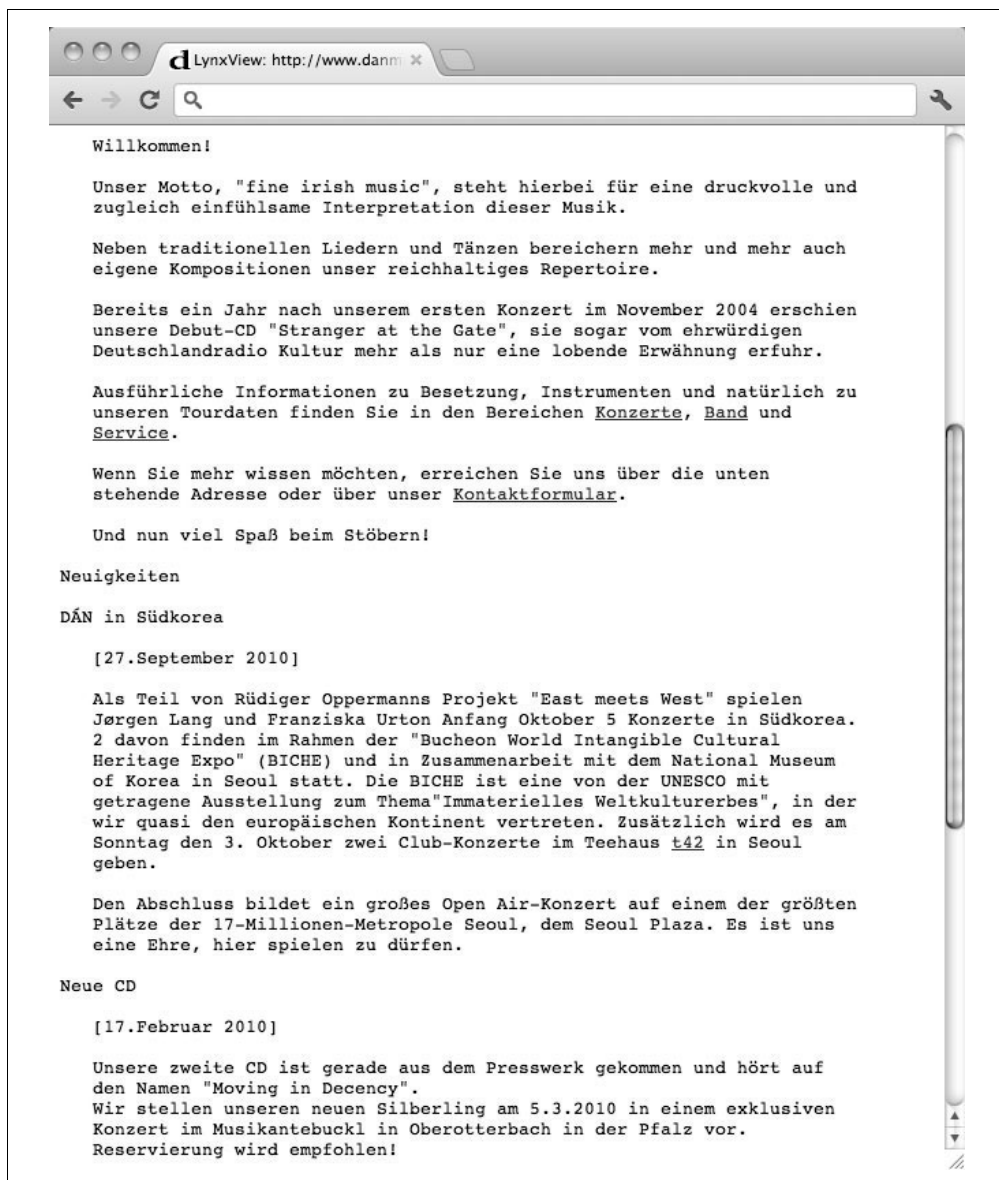


Abbildung 13-5: Eine Website, wie sie von Lynx Viewer dargestellt wird

Lösung

Nehmen Sie nach Möglichkeit die Textkodierung UTF-8. Auf diese Weise haben Sie die größtmögliche Sicherheit, dass die in Ihren Dokumenten verwendeten Zeichen auch tatsächlich korrekt dargestellt werden. Die meisten Text- und Codeeditorprogramme bieten eine Möglichkeit, die Kodierung beim Speichern mit anzugeben.

Damit Browser die Kodierung Ihres Stylesheets richtig erkennen, können Sie am Anfang des CSS-Dokuments schreiben:

```
@charset "UTF-8";

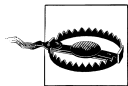
[href^="http://"]:after {
  content: ' ↗';
}
```

Damit geben Sie ausdrücklich an, dass die vorliegende CSS-Datei im Unicode-Format UTF-8 kodiert ist. Auf diese Weise ist es möglich, auch Sonderzeichen ohne Umwege über Escape-Sequenzen direkt im Stylesheet zu verwenden.

Diskussion

Durch den Unicode-Standard haben Sie die Möglichkeit, Dinge wie die Namen von Schriftarten oder mit der Eigenschaft `content` erzeugte Inhalte ohne zusätzliche Kodierung anzugeben.

Ohne die Verwendung der `@charset`-Regel würde der Browser versuchen, den Wert der Eigenschaft `content` nach dem Gebietsschema (Locale) des Computers zu interpretieren. Das Ergebnis wäre eine unverständliche Buchstabensuppe.



Die in dieser Lösung gezeigte `@`-Regel sollte nicht verwechselt verwendet mit dem `charset`-Attribut für `link`-Elemente. In der HTML-4.01-Spezifikation war es noch möglich, die Kodierung des Zieldokuments anzugeben. Aufgrund mangelnder Browserunterstützung wurde dieses Attribut in HTML5 jedoch wieder verworfen.

Um das zu verhindern, können Sie das gewünschte Zeichen auch in der hexadezimalen Unicode-Escape-Sequenz angeben. Hierbei wird anstelle des Zeichens ein bestimmter Zahlen- und Buchstabencode verwendet. Die oben gezeigte Lösung hätte dann zum Beispiel so ausgesehen:

```
[href^="http://"]:after {
  content: ' \21971'; /* hexadezimal */
}
```

Hier haben wir das gleiche Zeichen in hexadezimaler Form angegeben. Damit dies korrekt erkannt wird, wird der Zahlencode mit einem Backslash geschützt.

Das funktioniert ebenfalls zuverlässig. Allerdings ist es meistens einfacher, wenn Sie bereits direkt bei der Erstellung des Stylesheets sehen können, welches Zeichen hier eingefügt werden soll.

Siehe auch

Die Website des World Wide Web Consortium zur Verwendung verschiedener Zeichensätze unter <http://www.w3.org/International/questions/qa-css-charset.de.php>; die CSS 2.1-Spezifikation zur Definition von @-Regeln unter <http://www.w3.org/TR/CSS2/syn-data.html#at-rules>; das Rezept 9.5 zur Verwendung von erzeugten Inhalten in Druck-Stylesheets; einen Artikel auf der *Stackoverflow*-Website zur Verwendung von UTF-8 in Stylesheets unter <http://stackoverflow.com/questions/2526033/why-specify-charset-utf-8-in-your-css-file>; das Archiv der *css-d*-Mailingliste mit einem Artikel zu möglichen Problemen bei der Verwendung von Unicode in Stylesheets unter <http://archivist.incutio.com/viewlist/css-discuss/60075>.

13.7 Inhalte mit CSS erzeugen

Problem

Sie möchten Inhalte in eine Webseite einfügen, ohne hierfür den HTML-Quelltext verändern zu müssen. Das können beispielsweise Icons sein, die den Inhalt einer Seite erläutern, die Ausgabe von URIs in einem Stylesheet für Ausdrücke (siehe Rezept 9.5) oder auch Zähler für nummerierte Listen.

Lösung

Benutzen Sie die Pseudoelemente `:before` und `:after`, um mittels der Eigenschaft `content` verschiedene Inhalte vor oder nach einem Element einzufügen, zum Beispiel:

```
h1:before {  
    content: "Thema: ";  
}
```

Diese einfache Regel sorgt dafür, dass vor jeder Überschrift erster Ordnung die Zeichenkette *Thema* eingefügt wird, wie in Abbildung 13-6 gezeigt.

Eine weitere Möglichkeit besteht darin, die Werte von bestimmten Attributen anzeigen zu lassen. Wenn Sie sich beispielsweise alle Inline-Stile im Browser betrachten möchten, können Sie Folgendes schreiben:

```
[style]:before {  
    content: "CSS: " attr(style) " ";  
    color: green;  
}
```

Hier kombinieren wir einen allgemeinen Attributselektor mit dem Pseudoelement `:before`. Besitzt ein Element ein `style`-Attribut, wird dessen Inhalt direkt im Browserfenster angezeigt.



Im Gegensatz zu vielen anderen Funktionen wird der Name des Attributs innerhalb der `attr()`-Funktion nicht mit Anführungszeichen umgeben.

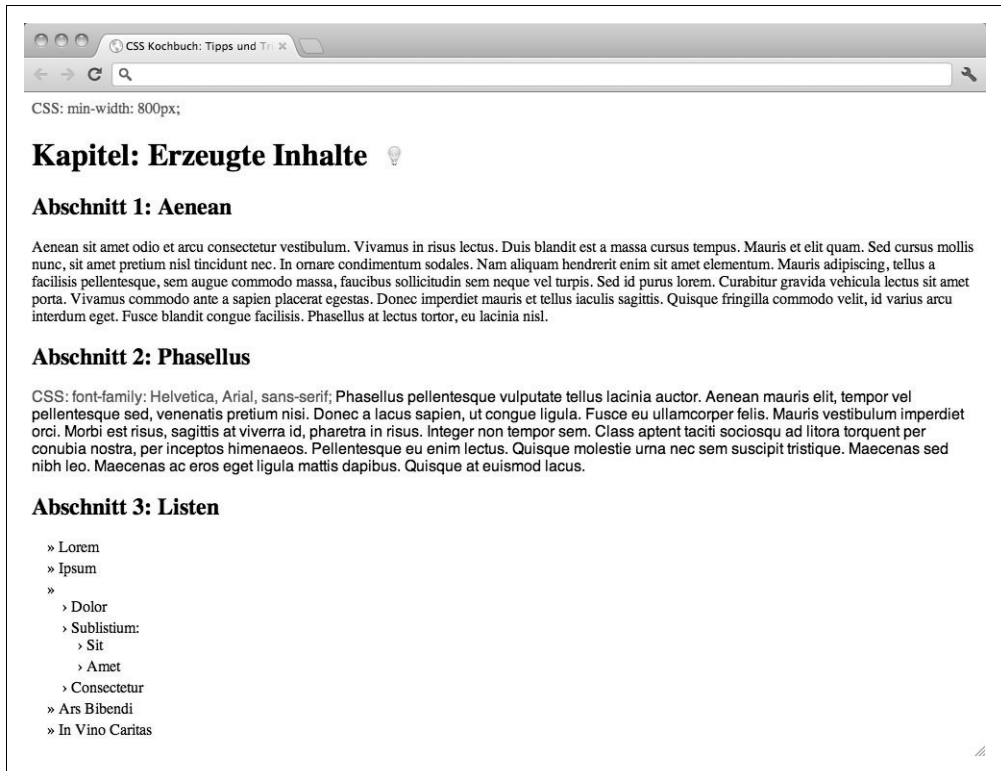


Abbildung 13-6: Verschiedene Möglichkeiten, erzeugte Inhalte zu nutzen

Neben den bereits gezeigten Optionen können Sie für den Inhalt auch eine URL angeben, beispielsweise um ein Bild einzubinden:

```
h1:after {
  content: url('bilder/gluehbirne.png');
  padding-left: 20px;
}
```

Auf diese Weise wird nach jeder Überschrift erster Ordnung eine kleine Grafik eingefügt. Weniger bekannt ist die Möglichkeit, automatische Zähler in ein Dokument einzufügen. Das kann hilfreich sein, wenn Sie bestimmte Abschnitte eines Dokuments automatisch nummerieren möchten:

```
body {
  counter-reset: abschnitt;
}

h2:before {
  counter-increment: abschnitt;
  content: "Abschnitt " counter(abschnitt) ": ";
}
```

Die erste Regel erzeugt anhand der Eigenschaft `counter-reset` einen neuen Zähler mit dem Namen `abschnitt` und setzt ihn auf den Startwert 0.

In der zweiten Regel benutzen wir die Eigenschaft `counter-increment`, um den Wert des Zählers `abschnitt` um den Wert 1 zu erhöhen. Danach binden wir ihn als erzeugten Inhalt vor jeder Überschrift zweiten Grades ein. Übrigens müssen Sie den Zähler nicht unbedingt für das `body`-Element erzeugen. Es ist durchaus möglich, Zähler auch für `article`- und `section`-Elemente zu definieren.

Eine weitere Möglichkeit, erzeugte Inhalte zu nutzen, ist die Definition eigener Aufzählungszeichen für Listen, die Sie vielleicht schon aus Rezept 5.3 kennen. In Abbildung 13-6 sehen Sie eine Übersicht verschiedener Möglichkeiten, erzeugte Inhalte zu nutzen.

Diskussion

Erzeugte Inhalte bieten viele Möglichkeiten, die Darstellung von Webdokumenten zu erweitern. An dieser Stelle ist es wichtig, zwischen den eigentlichen Inhalten der Seite und den mit CSS erzeugten Inhalten zu unterscheiden.

Wir haben hier leider nicht ausreichend Platz, Ihnen alles zu zeigen, was mit erzeugten Inhalten möglich ist. So kann die Definition von Zählern beispielsweise deutlich komplexer ausfallen. Ein paar Dinge sollten Sie dennoch wissen:

- Wenn Sie Umlaute und andere Sonderzeichen als Wert für `content` benutzen, sollten Sie am Anfang auf jeden Fall die Textkodierung per `@charset "UTF-8";` angeben, wie in Rezept 13.6 beschrieben. Ansonsten muss das Sonderzeichen in der hexadezimalen Unicode-Schreibweise angegeben werden.
- Verschiedene CSS-Funktionen können im Wert von `content` nicht ineinander verschachtelt werden. Dinge wie `content: url(attr(href));` sind nicht möglich.
- Per `content` erzeugte Inhalte werden im Internet Explorer ab Version 8 unterstützt. Für frühere Versionen gibt es ein passendes jQuery-Plug-in (siehe unten).
- Laut CSS 2.1-Spezifikation werden die Pseudoelemente `:before` und `:after` mit einem vorangestellten Doppelpunkt versehen. Das mittlerweile gültige *Selectors*-Modul der CSS3-Spezifikation sieht dagegen vor, dass zwecks besserer Unterscheidung von Pseudoklassen zwei Doppelpunkte verwendet werden (`::before`, bzw. `::after`).



Aus Gründen der Rückwärtskompatibilität verwenden wir für CSS 2.1-Pseudoelemente in diesem Buch die weiterhin unterstützte Schreibweise mit einem Doppelpunkt.

Wie am Anfang der Diskussion gesagt, sollten erzeugte Inhalte ausschließlich dafür benutzt werden, die Darstellung von Webdokumenten zu erweitern. Webinhalte sollten auf jeden Fall im HTML-Dokument definiert werden – nicht im Stylesheet!

Siehe auch

Die CSS 2.1-Spezifikation zu erzeugten Inhalten unter <http://www.w3.org/TR/CSS2/generate.html>; das Rezept 3.17 zur Erzeugung »intelligenter« Anführungszeichen mithilfe erzeugter Inhalte; die Rezepte 9.4 und 9.5 zur Verwendung erzeugter Inhalte in Druck-Stylesheets; die Website von Luke Lutman mit einem jQuery-Plug-in, mit dem erzeugte Inhalte auch im IE7 möglich werden unter <http://jquery.lukelutman.com/plugins/pseudo/index.html>; IE7.js, ein Polyfill, mit dem alte Versionen des Internet Explorer standardfreundlicher werden, unter <http://code.google.com/p/ie7-js/>; ein Online-Tutorial zu erzeugten Inhalten unter <http://onwebdev.blogspot.com/2012/02/css-generated-content-tutorial.html>; den Artikel *Styling Ordered List Numbers* unter http://www.456bereastreet.com/archive/201105/styling_ordered_list_numbers/.

13.8 Das Boxmodell anpassen

Problem

Sie möchten selbst festlegen, ob sich Höhe und Breite eines Elements auf dessen Inhaltsbereich beziehen oder auf die *Border Box*.

Lösung

Verwenden Sie die CSS3-Eigenschaft `box-sizing` mit den Werten `content-box` bzw. `border-box`, wie hier gezeigt:

```
div {
    background: orange;
    width: 300px;
    height: 200px;
    padding: 20px;
    margin: 30px;
    border: 10px solid yellow;
}

div#content-box {
    box-sizing: content-box;
}

div#border-box {
    box-sizing: border-box;
}
```

Wie Sie sehen, erhalten zunächst alle `div`-Elemente die gleiche Breite und Höhe. Allerdings wird in den folgenden Regeln ein unterschiedliches Modell zur Berechnung von `width` und `height` verwendet, wie Abbildung 13-7 zeigt.

Obwohl wir für beide `div`-Elemente die gleichen Werte für `width` und `height` verwendet haben, werden sie unterschiedlich groß dargestellt. Beim `div`-Element auf der linken Seite

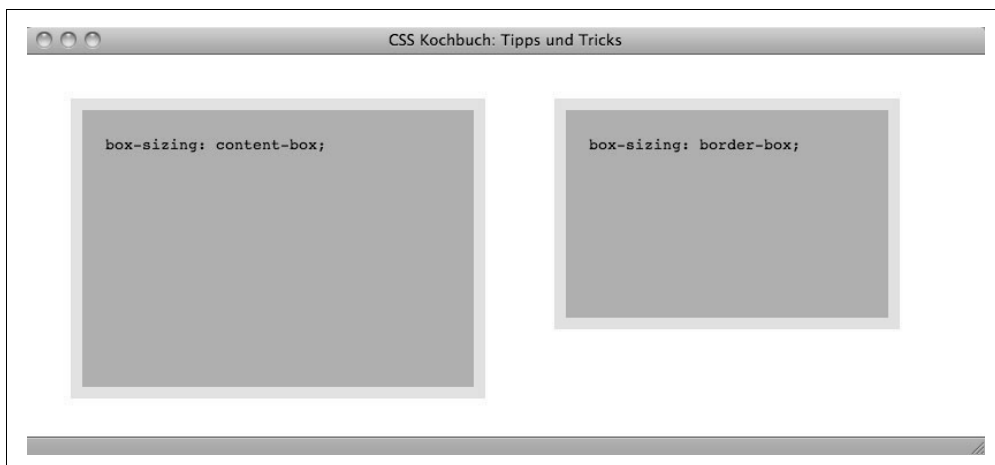


Abbildung 13-7: Verschiedene Werte für box-sizing

werden Breite und Höhe nach dem Standard-Boxmodell berechnet, das in der CSS 2.1-Spezifikation definiert ist.

Das `div`-Element auf der rechten Seite verwendet dagegen den Wert `border-box`. Hierdurch beziehen sich die Werte für `width` und `height` nicht mehr nur auf den Inhaltsbereich. Stattdessen sind die Werte für `padding` und `border-width` darin enthalten.

Diskussion

Das *Basic User Interface*-Modul der CSS3-Spezifikation definiert eine neue Eigenschaft namens `box-sizing`. Obwohl das Modul noch keinen offiziellen Empfehlungsstatus besitzt, wurde diese Eigenschaft bereits von vielen Browserherstellern übernommen.

Sie können, einfach ausgedrückt, mit der Eigenschaft `box-sizing` selbst steuern, wie die Breite und Höhe eines Elements berechnet werden. Hierfür stehen die Werte `content-box` und `border-box` zur Verfügung.

Der Standardwert `content-box` sorgt dafür, dass ein Element gemäß CSS 2.1-Boxmodell beschrieben wird (siehe dazu Rezept 1.16).

Verwenden Sie dagegen die Deklaration `box-sizing: border-box`, beziehen sich die Werte für `width` und `height` nicht mehr auf den Inhaltsbereich, sondern auf die sogenannte *Border Box*, also den Bereich, der von den Außenkanten eines Rahmens begrenzt wird. Wenn Sie wissen möchten, wie breit der Inhaltsbereich tatsächlich ist, müssen Sie die Werte für `padding` und `border` vom Wert für `width` bzw. `height` subtrahieren:

Breite Inhaltsbereich = `width` - (`border-left` + `padding-left` + `padding-right` + `border-right`)

beziehungsweise:

Höhe Inhaltsbereich = `height` - (`border-top` + `padding-top` + `padding-bottom` + `border-bottom`)

Auf diese Weise können Sie einem Element im Layout eine Breite zuweisen, ohne vorher wissen zu müssen, wie breit mögliche Innenabstände und Rahmen sein werden.

Mit der Eigenschaft `box-sizing` haben Sie selbst die Wahl, nach welcher Methode Sie Breite und Höhe angeben wollen. Unter Umständen hilft Ihnen die Eigenschaft sogar dabei, alte Stylesheets, die für den Internet Explorer im Quirks-Modus geschrieben wurden, weiterzuverwenden.

Um möglichst viele Browser zum Genuss dieser Eigenschaft zu verhelfen, sollten Sie sie für Firefox und Safari zusätzlich in der Präfixschreibweise angeben:

```
div.border-box {
    -webkit-box-sizing: border-box;
    -moz-box-sizing: border-box;
    box-sizing: border-box;
}
```

Siehe auch

Das *Basic User Interface*-Modul der CSS3-Spezifikation unter <http://www.w3.org/TR/css3-ui/>; den Artikel *Box-Sizing – CSS3* von Sven Wolfermann unter <http://maddesigns.de/box-sizing-1597.html>; die Website *When Can I Use* zur Unterstützung für die Eigenschaft `box-sizing` unter <http://caniuse.com/#search=box-sizing>.

13.9 Float-Containern per Clearfix ihre Höhe wiedergeben

Problem

Ein Element wird als Float definiert. Dadurch hat das umgebende Element plötzlich keine Höhe mehr. Die Höhe des umgebenden Elements soll an die Dimensionen der enthaltenen Elemente angepasst werden.

Lösung

Verwenden Sie eine *Clearfix*-Technik. Geben Sie hierfür der Eigenschaft `overflow` für das umgebende Element einen anderen Wert als `visible`. Angenommen, Sie haben folgenden HTML-Code:

```
<div class="galerie">
  
  
</div>
```

Die Bildern werden nun als links angeordnete Floats definiert:

```
img {
    float: left;
}
```

Um das Problem sichtbar zu machen, umgeben wir den Float-Container mit einem 3 Pixel breiten Rahmen:

```
div.galerie {  
    border: 3px solid red;  
}
```

Ohne weitere Angaben erscheinen die Bilder nun unter dem Container, wie in Abbildung 13-8 zu sehen ist.

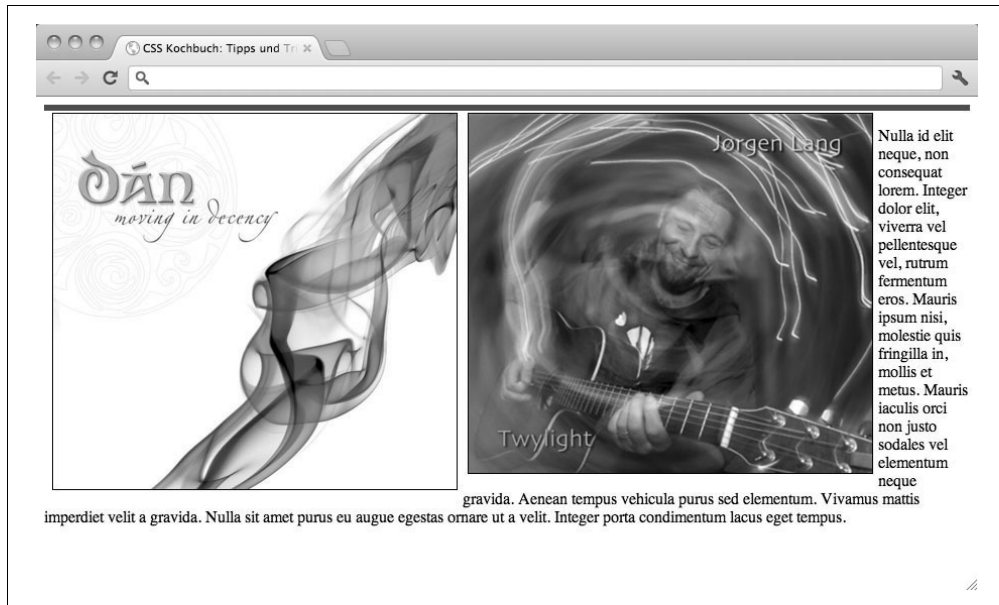


Abbildung 13-8: Die im Containerelement enthaltenen Bilder werden unterhalb des umgebenden Elements dargestellt und vom folgenden Inhalt umflossen

Damit sich der `div`-Container wieder an seine Höhe »erinnert«, reicht es aus, der Eigenschaft `overflow` einen anderen Wert als `visible` (den Standardwert) zu geben:

```
div.galerie {  
    overflow: auto;  
}
```

Sie lesen richtig. Das ist schon alles. Die `overflow`-Methode funktioniert selbst im Internet Explorer 7 ohne Probleme, da `overflow` dafür sorgt, dass für das Element `hasLayout` aktiviert wird. (Mehr zu `hasLayout` finden Sie in Rezept 13.15.) Das Ergebnis sehen Sie in Abbildung 13-9.

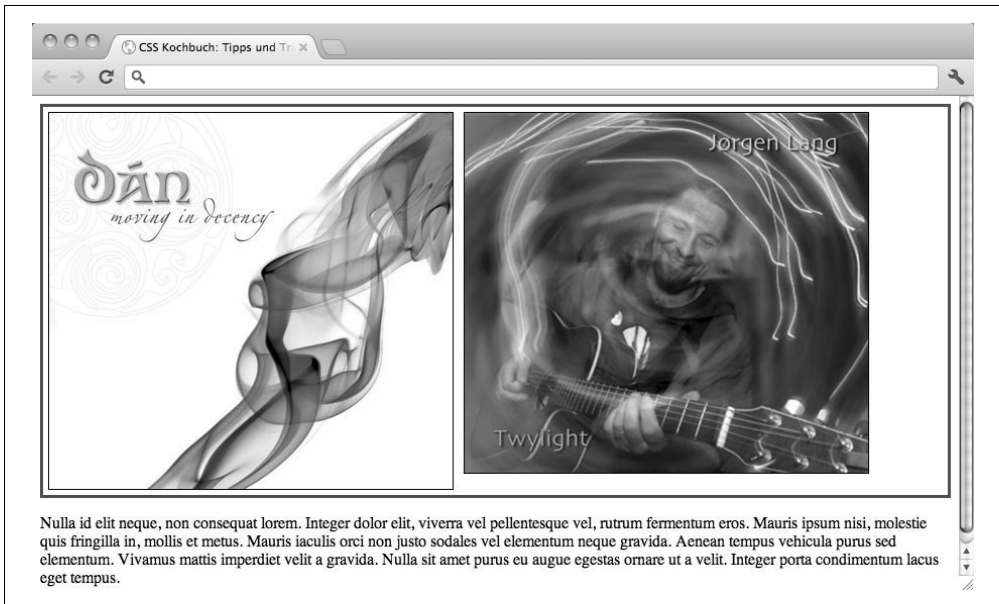


Abbildung 13-9: Durch die Deklaration `overflow:auto` erhält das umgebende `Container`-element seine Höhe wieder zurück

Diskussion

Als Floats definierte Elemente werden vom normalen Dokumentenfluss ausgenommen. Dadurch hat ihre Höhe für das umgebende `div`-Element keine Bedeutung mehr.

Durch die einfache Deklaration `overflow: auto`; können Sie dem `div`-Container seine Höhe wiedergeben. Das Prinzip hinter dieser so einfachen wie eleganten Lösung ist das Konzept des *Block Formatting Context*.

Normalerweise werden Blockelemente gemäß ihrer Reihenfolge im HTML-Quelltext im Ansichtsbereich angeordnet. Der Formatierungskontext ist dann also das `body`-Element der Seite. Hat `overflow` allerdings einen anderen Wert als `visible`, erhält das entsprechende Element seinen eigenen Blockformatierungskontext. Dadurch leben die Floats (in unserer Lösung die beiden Bilder) jetzt quasi in ihrer »eigenen Welt«.

Standardmäßig hat die Eigenschaft `overflow` den Wert `visible`. Dadurch laufen Inhalte, die für das umgebende Element zu groß sind, einfach darüber hinaus. Durch den Wert `auto` werden diese Inhalte verborgen und nur bei Bedarf mithilfe von Scrollbalken zugänglich gemacht. Ist das nicht erwünscht, können Sie anstelle von `auto` auch den Wert `hidden` verwenden, wodurch die Inhalte gegebenenfalls verborgen werden.

Sollte es bei dieser Methode zu Problemen kommen, können Sie immer noch auf die ältere Clearfix-Methode ausweichen, die im Folgenden beschrieben wird.

Clearfix – Old School-Methode

Wenn Sie die Bilder als Floats definieren, werden sie vom Dokumentenfluss ausgenommen. Das führt dazu, dass der `div`-Container seine Höhe »vergisst«.

Beindet sich aber unterhalb der Bilder ein weiteres Element, das kein Float ist, muss der Browser die Höhe der Bilder verwenden, um das zusätzliche Element an der richtigen Stelle zu platzieren. Das hat zur Folge, dass sich der umgebende Container wieder an seine Höhe »erinnert«. Anstatt ein zusätzliches `div`-Element in den HTML-Code einzufügen, können wir für den Container ein Pseudoelement definieren, das die Bilder nicht umfließt:

```
div.clearfix:after {  
  content: " ";  
  clear:   both;  
  display: block;  
  visibility: hidden;  
  height:  0;  
}
```

Hier erzeugen wir durch `:after` ein neues Blockelement, das nach den Bildern in das Container-`div` eingefügt wird. Um es zu »aktivieren«, muss ihm anhand der Eigenschaft `content` etwas Inhalt (hier ein einfaches Leerzeichen) zugewiesen werden. Durch die Eigenschaft `clear` mit dem Wert `both` legen wir ausdrücklich fest, dass dieses (Pseudo-)Element die zuvor definierten Elemente nicht umfließen soll.

Damit allein die Bilder die Höhe des umgebenden `div`-Containers bestimmen, müssen wir das Pseudoelement quasi unsichtbar machen. Hierfür verwenden wir erst einmal die Deklaration `visibility: hidden`. Dadurch wird das Element zwar nicht mehr angezeigt, der von ihm beanspruchte Platz bleibt aber erhalten. Damit das die Höhe des Containers nicht beeinflusst, wird außerdem die Höhe des Pseudoelements auf null gesetzt.

Siehe auch

Den Artikel *Everything you Know about Clearfix is Wrong* von Thierry Koblentz unter <http://thinkvitamin.com/design/everything-you-know-about-clearfix-is-wrong/>; den Artikel *The New Clearfix Method* zur ursprünglich üblichen Methode von Jeff Starr unter <http://perishablepress.com/press/2009/12/06/new-clearfix-hack/>; den Artikel *CSS 101: Block Formatting Contexts* von Thierry Koblentz unter <http://www.yuiblog.com/blog/2010/05/19/css-101-block-formatting-contexts/>; die CSS 2.1-Spezifikation zum *Block-Formatierungskontext* unter <http://www.w3.org/TR/CSS2/visuren.html>; das Rezept 12.10 zur Definition eines »Sticky Footers«; eine Liste verschiedener Methoden, um Float-Containern ihre Höhe wiederzugeben, unter <http://www.visibilityinherit.com/code/contain-floats.php>; das Rezept 13.18 zum Konzept des *hasLayout* im Internet Explorer 7; eine Liste mit weiteren Links zum Thema unter <http://delicious.com/Flocke/clearfix>.

13.10 HTML-Elemente dynamisch hervorheben

Problem

Sie möchten aktive Links, Formularfelder und Linkziele bei Bedarf hervorheben.

Lösung

Verwenden Sie die passenden Pseudoklassen `:active`, `:focus` und `:target`, um den Elementen bei Bedarf spezielle Stile zuzuweisen. Gerade angeklickten Links können Sie mit der Pseudoklasse `:active` eigene Stile zuweisen:

```
a:link {
    background: white;
    color:      blue;
}

a:active {
    background: white;
    color:      red;
}
```

Durch die zweite Regel werden gerade angeklickte Links in roter Schrift dargestellt.

Um das Formularelement hervorzuheben, das gerade Eingaben entgegennehmen kann, verwenden Sie die Pseudoklasse `:focus`:

```
input:focus, textarea:focus {
    background-color: #FFFFE0;
    outline:          2px solid #FACC2E;
}
```

Hier wird die Pseudoklasse `:focus` dazu genutzt, um Formularelemente, die gerade den Fokus haben, hervorzuheben. Als Hintergrundfarbe verwenden wir ein helles Gelb. Gleichzeitig erhält das Feld einen 2 Pixel breiten orangefarbenen Rahmen. Wir benutzen hier nicht die Eigenschaft `border`, sondern `outline`, da `border` bei einigen Browsern innerhalb des Eingabefelds dargestellt wird. Außerdem benutzt z.B. Safari die Eigenschaft `outline` bereits von sich aus, um das Feld von anderen zu unterscheiden. Durch diese Regel können wir den Standardwert des Browsers mit einem eigenen Wert überschreiben. (Diese Methode wird im Internet Explorer erst ab Version 8 unterstützt.)



Übrigens haben Sie in HTML5 die Möglichkeit, anhand des Attributs `autofocus` ein bestimmtes Element vorzuwählen. Ein so markiertes Element erhält den Fokus automatisch beim Laden der Seite und ist sofort zur Eingabe bereit.

Wenn Sie innerhalb eines Dokuments navigieren, können Sie die CSS3-Pseudoklasse `:target` benutzen, um beim Anklicken eines Links das Linkziel hervorzuheben und Benutzern so die Orientierung zu erleichtern:

```
p {
    background-color: #eee;
    border:          1px solid #666;
}

p:target {
    background-color: yellow;
    border-color:     orange;
}
```

Diese Regeln sorgen dafür, dass ein Absatz, auf den ein bestimmter Link verweist, farblich hervorgehoben wird, sobald der entsprechende Link angeklickt wird. Der HTML-Code dazu könnte beispielsweise so aussehen:

```
<a id="top"></a>
<nav>
<ul>
<li><a href="#kap1">Kapitel 1</a></li>
<li><a href="#kap2">Kapitel 2</a></li>
<!-- weitere Links hier -->
</ul>
</nav>

<div>
<p id="kap1">Lorem ipsum dolor sit amet...</p>
<!-- weitere Absätze hier -->
```

Klickt der Benutzer auf einen der Links, wird der entsprechende Absatz durch die zuvor definierten Regeln hervorgehoben, indem sich Hintergrund- und Rahmenfarbe ändern. So ist schnell erkennbar, auf welchen Teil des Dokuments der Link verweist.

Diskussion

Während die Verwendung der Pseudoklasse `:active` zum Standardrepertoire gehört, wird `:focus` gern vernachlässigt. Dabei kann sie die Benutzbarkeit – speziell von Formularen – deutlich erhöhen.

Die Pseudoklasse `:target` ist Teil des *Selectors*-Moduls der CSS3-Spezifikation. Obwohl sie noch nicht den Status einer W3C-Empfehlung hat, wird sie bereits von den meisten aktuellen Browsern unterstützt (IE ab Version 9, bei Bedarf können Sie hier den Polyfill *selectivizr* verwenden). Das funktioniert übrigens nicht nur bei Verweisen innerhalb eines Dokuments, sondern auch, wenn von »außen« auf das Element verwiesen wird. Wichtig ist hierbei nur, dass das Element ein zum Fragment-Identifizierer (z.B. `#kap3`) passendes `id`-Attribut trägt. Hierbei werden die `:target`-Stile im Zieldokument definiert.

Siehe auch

Das Rezept 1.14 zur Verwendung von Pseudoklassen; die CSS 2.1-Spezifikation zu den Pseudoklassen `:active` und `:focus` unter <http://www.w3.org/TR/CSS2/selector.html#dynamic-pseudo-classes>; die CSS3-Spezifikation zur Pseudoklasse `:target` unter <http://www.w3.org/TR/selectors/#target-pseudo>; einen Artikel von Jens O. Meiert zur `:target`-

Pseudoklasse unter <http://meiert.com/de/publications/articles/20060119/>; die (vorläufige) HTML5-Spezifikation zum autofocus-Attribut unter <http://www.w3.org/TR/html5/association-of-controls-and-forms.html#autofocusing-a-form-control>.

13.11 Elemente nur in Hilfsgeräten darstellen

Problem

Bestimmte Elemente, beispielsweise ein Link zum Hauptinhalt oder zusätzliche Hinweise für Benutzer mit visuellen Einschränkungen, sollen in bildschirmbasierten Medien nicht dargestellt werden. Für Screenreader und andere Hilfstechnologien sollen diese Elemente jedoch sichtbar sein.

Lösung 1

Verwenden Sie die in Rezept 4.6 beschriebene »Off-Left«-Technik, um bestimmte Elemente aus dem Ansichtsbereich herauszuschieben. Geben Sie den Elementen hierfür eine absolute Positionierung und verwenden Sie dann einen möglichst hohen negativen Wert für die Eigenschaft `left`:

```
.versteckt {  
  position: absolute;  
  left:    -9999em;  
}
```

Zunächst werden mit der Klasse `versteckt` markierte Elemente durch die absolute Positionierung vom normalen Dokumentenfluss ausgenommen. Anschließend sorgt der negative Wert für die Eigenschaft `left` dafür, dass das Element außerhalb des Ansichtsbereichs »geparkt« wird.

Screenreader und andere Hilfsgeräte ignorieren die Eigenschaften `position: absolute` und `left: -9999em`, wodurch der Link entsprechend markiert und mit vorgelesen bzw. auf der Braillezeile ausgegeben wird.

Lösung 2

Eine weitere, weniger bekannte Methode beschreibt Jonathan Snook auf seiner Website. Hierbei wird das betreffende Element absolut positioniert und mit der Eigenschaft `clip` so »zurechtgestutzt«, dass es effektiv nicht mehr zu sehen ist:

```
.versteckt {  
  position: absolute;  
  height:  1px;  
  width:    1px;  
  overflow: hidden;  
  clip: rect(1px 1px 1px 1px); /* IE7- */  
  clip: rect(1px, 1px, 1px, 1px);  
}
```

Die absolute Positionierung sorgt dafür, dass das Element vom normalen Dokumentenfluss ausgenommen wird und das übrige Layout nicht mehr stört. Durch die Angabe einer Höhe und Breite größer als null wird das versteckte Element auch von Apples VoiceOver ausgegeben. Durch die Anweisung `overflow: hidden` sorgen wir dafür, dass die Darstellung auch in älteren Webkit-Browsern (Safari und Chrome) funktioniert, wenn sich das Element am Rand des Ansichtsbereichs befindet.

Tatsächlich versteckt wird das Element jedoch durch den Einsatz der Eigenschaft `clip`. Sie besagt, dass das Element von einem Quadrat mit einer Kantenlänge von jeweils 1 Pixel umgeben sein soll.

Die Deklaration wird hier in zwei Schreibweisen angegeben, weil Internet Explorer 7 und seine Vorgänger die korrekte Syntax (bei der die einzelnen Werte durch Kommata getrennt werden) nicht verstehen. Das Ergebnis ist ein Element, das effektiv vor visuellen Browsern verborgen wird, ohne dass die Zugänglichkeit darunter leidet.

Diskussion

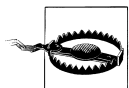
Die in der Lösung beschriebene »Off-Left«-Methode hat sich als am zuverlässigsten erwiesen. Da Screenreader-Programme wie zum Beispiel JAWS die Eigenschaften `position` und `left` nicht interpretieren, werden derart »versteckte« Elemente mit Hilfstechnologien zuverlässig ausgegeben.

Gelegentlich kann es passieren, dass mit der »Off-Left«-Methode verborgene Elemente den Fokus erhalten (z.B. Formularelemente oder Links). Hierbei versehen einige Browser das Element per `outline` mit einem Rahmen, um es hervorzuheben, der möglicherweise auch in der Bildschirmdarstellung zu sehen ist. Um das zu unterbinden, können Sie die in Lösung 2 gezeigte Regel bei Bedarf um ein `overflow: hidden` ergänzen.

Elemente vor allen Browsern verstecken

Gelegentlich kommt es vor, dass Sie bestimmte Elemente nicht nur vor Bildschirmbrowsern, sondern auch vor Leseprogrammen und anderen Hilfstechnologien verstecken möchten. Das kann beispielsweise der Fall sein, wenn Sie keinen Zugriff auf die Templates eines CMS haben, sondern nur die Stylesheets editieren können. Vielleicht wollen Sie aber auch unerwünschte Werbung im Benutzer-Stylesheet Ihres Browsers ausfiltern.

Eigentlich sollte man denken, dass es reicht, das betreffende Element per `display: none` auszublenden.



Mit dieser Methode verbergen Sie Elemente nicht nur vor Bildschirmbrowsern, sondern auch vor den meisten Hilfstechnologien. Das ist das Gegenteil von Zugänglichkeit. Nutzen Sie diese Technik also nur, wenn Sie wissen, was Sie tun.

Allerdings haben ausgerechnet zwei der beliebtesten Screenreader, namentlich JAWS und Window Eyes, ihre eigene Meinung dazu. Befindet sich innerhalb eines per `display: none` ausgeblendeten `span`-Elements ein Link (a), wird dieser trotzdem vorgelesen. Um das zu unterbinden, müssen Sie zusätzlich die Deklaration `visibility: hidden` verwenden. Soll das Element auch mit Apples VoiceOver-Technologie zuverlässig nicht ausgegeben werden, geben Sie ihm außerdem die Höhe 0. Hier die komplette Regel:

```
.unsichtbar-und-nicht-zu-hoeren {  
    display: none;  
    visibility: hidden;  
    height: 0;  
}
```

Überlegen Sie gut, bevor Sie diese Regel in einem Produktions-Stylesheet benutzen. Möglicherweise gibt es bessere Methoden, unerwünschte Inhalte auszublenden.¹

Siehe auch

Das Wiki der Mailingliste *css-discuss* zur Sichtbarkeit von Elementen in Bildschirmlesegeräten unter http://css-discuss.incutio.com/wiki/Screenreader_Visibility; einen Artikel in der Yahoo! Accessibility Library zu weiteren Techniken zum Verstecken von Inhalten unter <http://yaccessibilityblog.com/library/css-clip-hidden-content.html>; den Artikel von Jonathan Snook zur Verwendung der `clip`-Technik unter http://snook.ca/archives/html_and_css/hiding-content-for-accessibility; eine Beschreibung des `focus`-Problems bei der Verwendung der »Off-Left«-Methode unter <http://www.webkrauts.de/2009/12/11/verliere-nicht-den-focus/>; den Artikel von Gez Lemon über die Probleme bei der Verwendung von `display: none` unter <http://juicystudio.com/article/screen-readers-display-none.php>.

13.12 Inline-Stile überschreiben

Problem

Sie möchten als Inline-Stile (im HTML-Quelltext) definierte CSS-Regeln überschreiben, haben aber keinen Zugriff auf die HTML-Dokumente bzw. -Templates, sondern nur auf ein externes Stylesheet.

Lösung

Verwenden Sie Attributselektoren, um die `style`-Attribute im HTML-Quellcode auszuwählen. Angenommen, es wurde folgender Inline-Stil definiert:

```
<p style='font-weight: bold; color: rgb(255,0,0);'>Lorem Ipsum dolor sit amet...</p>
```

¹ Vielleicht reicht es ja schon aus, dem Systemadministrator ein Bier auszugeben, damit er Ihnen die nötigen Zugriffsrechte auf die Template-Datei gewährt.

Diese Definitionen können Sie in einem internen oder externen Stylesheet überschreiben, indem Sie einen passenden Attributselektor verwenden und die Werte mit dem Schlüsselwort `!important` versehen:

```
p[style] {  
    font-weight: normal !important;  
    color:      black !important;  
}
```

Auf diese Weise erhalten die Werte aus Ihrem Stylesheet Vorrang vor den Inline-Stilen. Der Absatz erscheint jetzt nicht mehr in Fettschrift, sondern in normaler Linienstärke und schwarzer Schrift.

Dieses Konzept lässt sich noch erweitern, um beispielsweise nur Absätze zu finden, die ganz bestimmte Inline-Stile besitzen:

```
p[style*='bold'] {  
    font-weight: normal !important;  
    color:      black !important;  
}
```

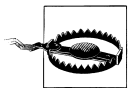
Hier verwenden wir den CSS3-Selektor für Teilstrings im Attributwert, um nur Absätze zu finden, deren `style`-Attribut die Zeichenkette `bold` enthält. Der Vorteil liegt darin, dass Sie die Selektoren für bestimmte Situationen maßschneidern können, etwa wenn ein Zugriff über Klassen- oder ID-Selektoren nicht möglich ist.

Der Nachteil liegt in der noch mangelnden Unterstützung für CSS3-Attributselektoren im Internet Explorer. Diese funktionieren erst ab Version 9 zuverlässig. (In Rezept 13.8 wird beschrieben, wie Sie dieses Problem mit dem JavaScript-Polyfill *selectivizr* beheben können.)

Diskussion

Der Schlüssel zum Verständnis dieser Lösung liegt in der Spezifität von CSS-Regeln. Gemäß der CSS-Spezifikation haben Inline-Stile prinzipiell Vorrang vor Regeln, die in internen oder externen Stylesheets definiert wurden.

Die einzige Möglichkeit, dies zu umgehen, ohne den HTML-Quellcode zu ändern, besteht in der Verwendung des Schlüsselworts `!important`. Ein so markierter Wert erhält, unabhängig von der Spezifität des Selektors, Vorrang vor allen anderen.



Wenn Sie `!important` einsetzen, sollten Sie sehr vorsichtig vorgehen. Es kann leicht passieren, dass Sie sich hiermit Ihr gesamtes Design zerschneiden.

Siehe auch

Das Rezept 1.20 zum Thema Spezifität; das Rezept 1.21 zur Verwendung des Schlüsselworts `!important`; den Artikel *CSS Specificity: Things you should know* von Vitaly Friedman unter <http://coding.smashingmagazine.com/2007/07/27/css-specificity-things-you->

should-know/; die CSS3-Spezifikation zur Verwendung der Spezifität unter <http://www.w3.org/TR/css3-selectors/#specificity>; das Rezept 13.18 zur Verwendung des Polyfill *selectivizr*.

13.13 Stylesheets mit Conditional Comments nur für den Internet Explorer definieren

Problem

Sie wollen bzw. müssen bestimmte CSS-Stile ausschließlich für den Internet Explorer definieren. Hierbei soll es auch die Möglichkeit geben, nur ganz bestimmte Versionen anzusprechen oder auszuschließen.

Lösung

Verwenden Sie Conditional Comments (CC), um den Code vor anderen Browsern bzw. IE-Versionen zu verbergen. Um ein Stylesheet nur für den Internet Explorer zu definieren, können Sie schreiben:

```
<!--[if IE]>
<style type='text/css'>
/* hier stehen die Stile für Internet Explorer */
</style>
<![endif]-->
```

Soll nur eine ganz bestimmte Version des Internet Explorer angesprochen werden, erweitern Sie den CC wie hier gezeigt:

```
<!--[if IE 7]>
<style type='text/css'>
/* Hier stehen die Stile für Internet Explorer 7 */
</style>
<![endif]-->
```

Wollen Sie bestimmte Stile nur für Versionen kleiner als Internet Explorer 8 definieren, erweitern Sie den CC um die Zeichenkette `lt` (less than – kleiner als):

```
<!--[if lt IE 8]>
<style type='text/css'>
/* Stile für Internet Explorer 7 und kleiner */
</style>
<![endif]-->
```

Soll auch der IE8 angesprochen werden, schreiben Sie stattdessen `lte` (engl. less than or equal, kleiner oder gleich):

```
<!--[if lte IE 8]>
<style type='text/css'>
/* Stile für Internet Explorer 8 und kleiner */
</style>
<![endif]-->
```

Das geht auch andersherum. Um ein Stylesheet nur für IE7 und neuer zu definieren, schreiben Sie `gte` (greater than or equal – größer oder gleich) anstelle von `lte`:

```
<!--[if gte IE 8]>
<style type='text/css'>
/* Stile für Internet Explorer 8 und kleiner */
</style>
<![endif]-->
```

Wollen Sie mehrere Versionen ansprechen, können Sie diese mit einer ODER-Verknüpfung gemeinsam ansprechen:

```
<!--[if (IE 6)|(IE 7)|(IE 8)]>
<style type='text/css'>
/* Stile für Internet Explorer 6, 7 und 8 */
</style>
<![endif]-->
```

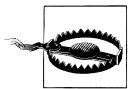
Das Gleiche können Sie auch mit einer UND-Verknüpfung formulieren, indem Sie schreiben:

```
<!--[if (gt IE 5)&(lt IE 8)]>
<style type='text/css'>
/* Stile für Internet Explorer 6, 7 und 8 */
</style>
<![endif]-->
```

Mit Conditional Comments ist es außerdem möglich, bestimmte Teile einer Webseite vor dem Internet Explorer zu verbergen. Die folgende Anweisung definiert ein Stylesheet für alle anderen Browser:

```
<!--[if !IE]> -->
/* Hier steht der Code für alle Browser außer IE */
<!-- <![endif]-->
```

Das Ausrufungszeichen (!) steht hierbei für eine Verneinung. Nun fragen Sie sich vielleicht, wie das funktionieren soll. Der Trick besteht darin, die Bedingung mit einem vollständigen HTML-Kommentar (`<!-- ... -->`) zu umgeben. Der Internet Explorer sieht die Bedingung (`if !IE`) und ignoriert den HTML-Code, bis die Bedingung in einem weiteren HTML-Kommentar beendet wird.



Wenn Sie auf Ihrem Testrechner mehrere eigenständige Versionen des IE installiert haben, werden die Conditional Comments so interpretiert, als sei nur die neueste Version vorhanden. Das lässt sich mit einem Werkzeug wie *IETester* umgehen (siehe Rezept 13.2).

Hierbei sollten Sie allerdings nicht vergessen, dass der Wartungsaufwand mit jedem eingebundenen Stylesheet für den Internet Explorer steigt. Bei Aktualisierungen der Stile müssen Sie möglicherweise an mehreren Dokumenten die gleichen Änderungen vornehmen. Zudem muss für jede eingebundene CSS-Datei ein zusätzlicher HTTP-Request durchgeführt werden, was die Ladezeit und den Bandbreitenverbrauch erhöhen kann.

Außerdem ist es möglich, dass »Fehlerbehebungen« oder Workarounds für Probleme in Folgeversionen nicht mehr richtig funktionieren oder obsolet werden. Daher sollten Sie den CSS-Code, der mithilfe von Conditional Comments geladen wird, auf das Wesentliche beschränken und die gewünschten Browserversionen so präzise wie möglich angeben.



Übrigens plant Microsoft, Conditional Comments ab Internet Explorer 10 nicht mehr zu unterstützen. Die Anweisungen werden dann wieder als das behandelt, was sie eigentlich sind: einfache HTML-Kommentare.

Conditional Classes

Conditional Comments können nicht nur für Stylesheets verwendet werden, sondern für beliebige HTML-Elemente. Diese Möglichkeit macht sich eine Methode zunutze, die erstmals auf der Website *positioniseverything.net* beschrieben wurde. Mithilfe von Conditional Comments wurden bei Bedarf zusätzliche div-Elemente in das HTML-Dokument eingefügt.

Paul Irish, einer der führenden Köpfe in der Frontend-Entwicklung, hat dieses Konzept weiterentwickelt. In seiner Lösung werden Conditional Comments verwendet, um abhängig von der verwendeten IE-Version ein bestimmtes body-Tag in die Seite einzufügen. Es wird jeweils ein bestimmtes class-Attribut mit den passenden Werten verwendet. Hier eine rudimentäre HTML5-Seite, die dieses Konzept umsetzt:

```
<!DOCTYPE html>
<!--[if lt IE 7]><html class="kl-9 kl-8 kl-7"><![endif]-->
<!--[if IE 7]> <html class="kl-9 kl-8"> <![endif]-->
<!--[if IE 8]> <html class="kl-9"> <![endif]-->
<!--[if gt IE 8]><!--><html> <!--<![endif]-->
<head>
  <meta charset="utf-8" />
  <title>Conditional Classes</title>

  <style type="text/css">
    @import url("css/stylesheet.css");
  </style>

</head>
<body>

<!-- Hier steht der Inhalt -->

</body>
</html>
```

Handelt es sich bei dem verwendeten Browser um einen Internet Explorer der Version 6 oder kleiner, greift die erste Bedingung (`lt IE 7`). In diesem Fall wird das `html`-Element mit den Klassen `kl-9` (für »kleiner als IE 9«), `kl-8` und `kl-7` ausgestattet. Wird der IE7 als Browser benutzt, erhält das `html`-Tag die Klassen `kl-9` und `kl-8`, beim IE8 ist es nur noch `kl-9`.

Die letzte Anweisung unterscheidet sich ein wenig von den übrigen. Durch diese Schreibweise wird dafür gesorgt, dass der Internet Explorer 9 und darüber, aber auch alle anderen Browser, das `html`-Tag sehen.

Der Vorteil dieser Methode liegt darin, dass Sie nur noch ein Stylesheet in das HTML-Dokument einzubinden brauchen. Es sind keine weiteren Änderungen am HTML-Code nötig. Im Stylesheet können Sie die Klassen nun benutzen wie folgt:

```
.kl-7 { /* Regeln für sehr alte Versionen von IE */ }
.kl-8 { /* Regeln für IE 7 und älter */ }
.kl-9 { /* Regeln für IE 8 und älter */ }

/* Regeln für alle Browser */
```

Manche Content-Management-Systeme (z.B. WordPress) verwenden das `body`-Tag, um ihm dynamisch Klassen zuzuweisen. Daher werden die »Conditional Classes« hier am `html`-Tag definiert.

Innerhalb der Klassen können Sie jede gültige CSS-Anweisung verwenden. Müssen Sie beispielsweise proprietäre CSS-Erweiterungen verwenden (z.B. die Eigenschaften `filter` oder `behaviour`), können Sie die Regeln auch in eine separate Datei auslagern und hier per `@import` laden.

Siehe auch

Die offizielle Dokumentation zu Conditional Comments unter <http://msdn.microsoft.com/de-de/library/ms537512.aspx>; den Wikipedia-Artikel zu Conditional Comments unter http://de.wikipedia.org/wiki/Conditional_Comments; die Kritik an der Verwendung von Conditional Comments von Jens O. Meiert unter <http://meiert.com/en/blog/20070201/why-conditional-comments-are-bad-repeat-bad/>; weitere Erläuterungen zum vorigen Artikel unter <http://meiert.com/en/blog/20080824/to-be-clear/>; den Artikel *Conditional stylesheets vs CSS hacks? Answer: Neither!* von Paul Irish unter <http://paulirish.com/2008/conditional-stylesheets-vs-css-hacks-answer-neither/>.

13.14 Stile für HTML5-Elemente im IE definieren

Problem

Neue HTML5-Elemente können in Internet Explorer 7 und 8 nicht mit Stilen versehen werden und werden falsch dargestellt.

Lösung

IE7 und 8 kennen diese Elemente nicht. Mit etwas JavaScript-Code können Sie diese Elemente aber beim Browser »anmelden«. So geht's:

```
<script type="text/javascript">
document.createElement("header");
document.createElement("article");
```

```
document.createElement("section");
document.createElement("nav");
document.createElement("footer");
document.createElement("iframe");
document.createElement("video");
document.createElement("audio");
document.createElement("dfn");
</script>
```

Ohne diese Angaben werden die Elemente zwar vom IE erkannt, aber möglicherweise falsch dargestellt. Verschachtelte Elemente werden beispielsweise als Geschwister interpretiert. Nehmen wir einmal den folgenden HTML-Codeschnipsel:

```
<section>
  <p>Etwas Text</p>
</section>
```

Der oben stehende Code wird ohne die »Anmeldung« der Elemente wie folgt umgesetzt:

```
<section></section>
<p>Etwas Text</p>
```

Anstatt die Elemente ineinander zu verschachteln, werden sie nacheinander dargestellt. Das führt schnell zu Inkonsistenzen und kann schlimmstenfalls Ihr gesamtes Layout durcheinanderbringen.

Diskussion

Von Haus aus haben ältere Versionen des Internet Explorer keine Kenntnis von den neuen HTML5-Elementen. Sie werden zwar erkannt, aber, wie oben zu sehen, nicht korrekt einsortiert bzw. dargestellt. Mit dieser einfachen Technik können Sie das ändern.

Falls Ihnen das zu viel Code ist, können Sie das JavaScript auch per »Hotlink« in Ihre Seiten einbinden. Um Bandbreite und HTTP-Requests zu sparen, haben wir die Anweisung hier mit Conditional Comments umgeben. So wird die Datei nur in Internet Explorer vor Version 9 geladen:

```
<!--[if lt IE 9]>
<script src="http://html5shiv.googlecode.com/svn/trunk/html5.js"></script>
<![endif]-->
```

Wie auch die übrigen in diesem Kapitel gezeigten Polyfills dient diese Technik vor allem dem Progressive Enhancement. Das heißt, Sie sollten die Gestaltung Ihrer Seiten nach Möglichkeit nicht von der Unterstützung für diese Methode abhängig machen.

Siehe auch

Den Artikel von Remy Sharp zum *HTML5 enabling Script* unter <http://remysharp.com/2009/01/07/html5-enabling-script/>; den Artikel *HTML5 elements in Internet Explorer without JavaScript* von Elco Klingen unter <http://debeterevormgever.nl/en/articles/html5-elements-ie-without-javascript>.

13.15 hasLayout-Probleme beheben

Problem

Viele Darstellungsprobleme (zum Beispiel mit Floats, Margin, Padding, Listenmarker usw.) in älteren Versionen des Internet Explorer haben mit dem Konzept des *hasLayout* zu tun. Sie möchten wissen, wie Sie die Darstellungsprobleme in den Griff bekommen.

Lösung

In vielen Fällen (speziell bei Problemen mit der Darstellung von Floats oder bei Verwendung der proprietären Eigenschaft *filter*) kann es helfen, für das betreffende Element oder seinen Container, *hasLayout* zu aktivieren. Hierfür reicht es normalerweise aus, zum Beispiel der Eigenschaft *width* einen expliziten Wert zu geben, zum Beispiel:

```
#element {  
    width: 1%;  
}
```

In anderen Situationen, etwa wenn Außenabstände nicht korrekt zusammengefasst werden oder bei Problemen mit Listen, ist es genau umgekehrt.

Um *hasLayout* wieder abzuschalten, müssen Sie diese Eigenschaft (und alle anderen, die möglicherweise *hasLayout* aktivieren) wieder auf ihren Standardwert zurücksetzen:

```
#element {  
    width: auto;  
}
```

Diskussion

In älteren Versionen des Internet Explorer (bis Version 7) verwendet Microsoft das Konzept des *hasLayout*. Dies ist keine CSS-Eigenschaft, sondern eher eine Art »Schalter«, der erheblichen Einfluss darauf haben kann, wie Elemente dargestellt werden. Die Auswirkungen sind sehr vielfältig. Wird ein Element im IE7 oder früher nicht korrekt dargestellt, lässt sich das Problem oftmals lösen, indem Sie den *hasLayout*-Schalter umlegen, wie in der Lösung beschrieben.

Die gesamte Thematik um *hasLayout* ist äußerst komplex. Eine ausführliche Behandlung würde schnell ein eigenes Kapitel in diesem Buch füllen. Daher werden wir uns hier auf das Wesentliche beschränken und gegebenenfalls auf weitere Ressourcen zum Thema verweisen.

Für folgende Elemente ist *hasLayout* offenbar² standardmäßig aktiv:

html, body, table, tr, th, td, img, hr, input, button, select, textarea, fieldset, legend, iframe, embed, object und applet.

² Die Informationen von Microsoft zum Konzept des *hasLayout* sind leider nur sehr spärlich.

Tabelle 13-1 zeigt, welche Eigenschaften und Werte Sie verwenden können, um *hasLayout* für Blockelemente zu aktivieren.

Tabelle 13-1: Eigenschaften/Werte zum Aktivieren von *hasLayout*

Eigenschaft	Wert(e)
display	inline-block
float	left, right
overflow(-x/-y)	hidden, scroll, auto
position	absolute, fixed
max-width/-height	alle Werte außer none
min-width/-height	alle Werte (auch 0)
width/height	alle Werte außer auto
writing-mode	tb-rl ¹
zoom	alle Werte außer normal ²

¹ ursprünglich Microsoft-proprietär, Teil von CSS3

² Microsoft-proprietär



Sofern Sie in Ihrem Stylesheet Deklarationen haben, die eine dieser Kombinationen aus Eigenschaft und Wert verwenden, ist *hasLayout* bereits für das betreffende Element aktiviert.

Bei Inline-Elementen ist die Auswahl schon deutlich kleiner. Hier können Sie den gewünschten Effekt nur anhand der Eigenschaften *width*, *height* oder *zoom* mit den in der Tabelle genannten Werten erreichen. Allerdings gibt es im IE7 eine Reihe von Problemen mit *height*. Es sollte daher zum Aktivieren von *hasLayout* für Inline-Elemente vermieden werden.

Soll *hasLayout* dagegen deaktiviert werden, müssen Sie umgekehrt vorgehen. Haben Sie eine oder mehrere Eigenschaften aus der oben stehenden Tabelle definiert, müssen diese wieder auf ihren Standardwert zurückgesetzt werden. Tabelle 13-2 zeigt die möglichen Kombinationen aus Eigenschaften und Werten.

Tabelle 13-2: Eigenschaften/Werte zum Deaktivieren von *hasLayout*

Eigenschaft	Wert(e)
float	none
overflow(-x/-y)	visible
position	static
max-width/-height	none
min-width/-height	auto ¹
width/height	auto
writing-mode	lr-tb
zoom	normal

¹ Dieser Wert ist nicht standardkonform, wird aber vom IE7 akzeptiert.

Wie Sie in Tabelle 13-2 sehen, fehlt hier die Eigenschaft `display`. Das heißt, Sie können `hasLayout` zwar per `display: inline-block` aktivieren, danach aber nicht mehr zurücksetzen (zum Beispiel per `display: block` oder `display: inline`)!

Ein ähnliches Problem tritt bei Verwendung der Eigenschaften `min-width` und `min-height` auf. Hier führen alle Werte dazu, dass `hasLayout` aktiviert wird. Um den Schalter zurückzusetzen, können Sie im IE7 allerdings den nicht standardkonformen Wert `auto` angeben.

Auf vielen Websites wird empfohlen, zum Aktivieren von `hasLayout` die Deklaration `zoom: 1` zu verwenden. Wir möchten hier davon abraten. Diese Formulierung ist nicht standardkonform und sorgt dafür, dass Ihr Stylesheet nicht validiert. In den meisten Fällen sollte die Angabe von `width: 1%` (oder eines anderen passenden Werts) den gleichen Effekt haben.

Die Darstellungsrollen `inline` und `inline-block` im Internet Explorer 7

Gelegentlich soll einem Element die Darstellungsrolle `Inline-Block` zugewiesen werden. Leider versteht der Internet Explorer die Deklaration `display: inline-block` nicht.

Stattdessen können Sie für das gewünschte Element `hasLayout` aktivieren und es dann explizit als `Inline-Element` definieren, zum Beispiel so:

```
.kl-8 #element {  
    width: 1%; /* hasLayout aktivieren */  
    display: inline; /* als Inline-Element definieren */  
}
```

Diese Regel sorgt dafür, dass sich das betreffende Element verhält, als hätte es die Darstellungsrolle `inline-block`. Im Selektor verwenden wir die Klasse `.kl-8`, wie in Rezept 13.13 gezeigt, um die Stile ausschließlich für den Internet Explorer 7 zu definieren.



Bitte versuchen Sie nicht, `hasLayout` für alle Elemente pauschal zu aktivieren. Das schafft meistens mehr Probleme, als es löst.

Um zu überprüfen, ob für ein Element `hasLayout` aktiviert ist, benutzen Sie am besten ein Werkzeug wie die IE Developer Toolbar.

Siehe auch

Die deutsche Übersetzung des Artikels *On having Layout* von Ingo Chao (Übersetzung von Corina Rudel) unter <http://onhavinglayout.fwpf-webdesign.de/>; die Microsoft-Dokumentation zu `hasLayout` unter <http://msdn.microsoft.com/de-de/library/Bb250481>; das Rezept 13.2 zum Testen von Websites in verschiedenen Browsern; den Sitepoint-Artikel zum Konzept von `hasLayout` unter <http://reference.sitepoint.com/css/haslayout>; den Artikel *IE7-/Win: inline-block and hasLayout* zur Definition von `Inline-Block-Elementen` im IE7 unter <http://www.brunildo.org/test/InlineBlockLayout.html>; eine umfassende Liste mit

Tests zu Problemen im Internet Explorer im *IE7-/Win: inline-block and hasLayout* zur Definition von Inline-Block-Elementen im IE7 unter <http://www.brunildo.org/test/InlineBlockLayout.html>; die CSS-Spezifikation zum Blockformatierungskontext unter <http://www.w3.org/TR/1998/REC-CSS2-19980512/visuren.html#block-formatting>.

13.16 Reset-Stylesheets benutzen

Problem

Alle aktuellen Browser verwenden ein sogenanntes Browser-Stylesheet (siehe Rezept 1.19), um eine Website darzustellen. Für jedes HTML-Element werden auf diese Weise bestimmte Grundeinstellungen definiert. Hierzu gehört die Schriftgröße von Überschriften genau so wie die Einrückungen von Listen (siehe Rezept 5.1) und die Darstellung von Bildern als Inline-Elemente.

Unglücklicherweise gibt es bei diesen Grundeinstellungen mehr oder weniger große Unterschiede zwischen den Browsern. Während einige Browser beispielsweise für die Einrückung von Listen und Listeneinträgen die Eigenschaft `margin` verwenden, benutzen andere Browser hierfür die Eigenschaft `padding`. Um die Darstellung Ihrer Seiten browserübergreifend anzugleichen, sollen die verschiedenen Grundeinstellungen der Browser ganz oder teilweise vereinheitlicht werden.

Lösung 1

Verwenden Sie eine oder mehrere Regeln, um bestimmte Eigenschaften mit selbst definierten Standardwerten zu versehen.

Die Minimalversion setzt Innen- und Außenabstände für alle Elemente auf null:

```
* {  
    margin: 0;  
    padding: 0;  
}
```

Das hat aber auch zur Folge, dass Sie diese Werte für alle verwendeten Elemente nun selbst definieren müssen. Eine genaue Kenntnis der Struktur Ihrer HTML-Seiten bzw. -Schablonen ist hierfür unabdingbar. So müssen Sie die Einrückung von Listen (`ol`, `ul`) und Listeneinträgen (`li`) nun beispielsweise selbst vornehmen.

Lösung 2

Deutlich weiter geht der CSS-Experte Eric Meyer mit seinem Reset-Stylesheet (*meyerweb-reset*). Er versucht, einen möglichst großen Satz an Elementen mit einer neutralen Darstellung zu versehen. Wir ersparen Ihnen hier den Abdruck des Codes. Bei Bedarf können Sie sich das Stylesheet von Eric Meyers Website herunterladen. (Die Adresse finden im Abschnitt »Siehe auch«.)

Eric Meyer betrachtet sein Reset-Stylesheet ausdrücklich nicht als fertige Lösung, die einfach kopiert und unverändert weiterverwendet wird. Stattdessen sollen die Regeln als Ausgangspunkt für Ihr eigenes Design dienen. Änderungen und Anpassungen sind nicht nur erwünscht, sondern teilweise sogar notwendig.

Durch einheitliche Innen- und Außenabstände, Zeilenhöhen und Schriftgrößen sollen die Darstellungsunterschiede zwischen den verschiedenen Browsern nach Möglichkeit ausgeräumt werden, damit Sie mit einem sprichwörtlichen »leeren Blatt« beginnen können.

Lösung 3

Eine weitere recht beliebte Lösung ist das von Yahoo! entwickelte *YUI 2: Reset CSS*. Auch hier wird versucht, durch das Zurücksetzen der Werte für bestimmte Elemente eine möglichst einheitliche Browserdarstellung zu erreichen.

Lösung 4

Einen anderen Weg geht *normalize.css*. Im Gegensatz zu den gerade vorgestellten Reset-Verfahren hat *normalize.css* nicht den Anspruch, alle Elemente und Eigenschaften auf ihre Standardwerte zurückzusetzen. Stattdessen geht es darum, sinnvolle Standardeinstellungen (z.B. Einrückungen von Listen) beizubehalten und zu vereinheitlichen. Gleichzeitig sollen Darstellungsfehler der Browser ausgeglichen und die Benutzbarkeit erhöht werden. Aufgrund der detaillierten Kommentare im Code müssen auch weniger versierte Benutzer verstehen können, was die einzelnen Regeln tun.

Diskussion

Die hier vorgestellten Lösungsansätze haben im Prinzip alle das gleiche Ziel: eine möglichst konsistente Darstellung in allen Browsern zu erreichen. Danach können Sie ohne Umwege Ihr Design umsetzen – zumindest in den meisten Fällen.

Alle diese Reset-Stylesheets haben mehr oder weniger große Überschneidungen, wobei der Schwerpunkt von *normalize.css* etwas anders gelagert ist.

Übrigens ist die ganz am Anfang der Lösung gezeigte Reset-Methode nicht unbedingt die effizienteste. Durch die Verwendung des universellen Selektors (*) versucht der Browser, die beiden Deklarationen auf restlos alle Elemente des Dokuments anzuwenden. Das kann gerade bei langen und komplexen Webseiten eine ziemliche Last bedeuten, die sich unter Umständen auch auf die Ladezeit auswirkt.

Siehe auch

Eine Sammlung verschiedener Reset-Techniken unter <http://perishablepress.com/press/2007/10/23/a-killer-collection-of-global-css-reset-styles/>; das Reset-Stylesheet von Eric Meyer unter <http://meyerweb.com/eric/tools/css/reset/>; Eric Meyers Erläuterungen zu sei-

nem Reset-Stylesheet unter <http://meyerweb.com/eric/thoughts/2011/01/03/reset-revisited/>; das YUI 2: CSS Reset unter <http://developer.yahoo.com/yui/reset/>; *normalise.css* unter <http://necolas.github.com/normalize.css/>; ein Onlinewerkzeug zum automatischen Erzeugen von Reset-Stylesheets unter <http://cssresetr.com/>; einen Vergleich der Browser-Stylesheets in verschiedenen Versionen des Internet Explorer unter <http://www.iecss.com/>; eine Tabelle mit Darstellungsunterschieden zwischen verschiedenen Browsern unter <http://css-class.com/test/css/defaults/UA-style-sheet-defaults.htm>; eine Tabelle mit den Standardeinstellungen verschiedener Browser unter <http://css-class.com/test/css/defaults/UA-style-sheet-defaults.htm>; einen Vergleich der Browser-Stylesheets für die letzten Versionen des Internet Explorer unter <http://www.iecss.com/>.

13.17 Die Möglichkeiten von CSS mit Präprozessoren erweitern

Problem

Sie möchten die Fähigkeiten von CSS erweitern, Variablen benutzen und den Code teilweise automatisch erzeugen lassen.

Lösung

Verwenden Sie einen CSS-Präprozessor wie z.B. *Less CSS* oder *Sass*. Anstatt beispielsweise die gleiche Farbe immer wieder anzugeben, reicht es aus, am Anfang des Stylesheets eine passende Variable zu definieren (hier am Beispiel von *Sass*):

```
$schwarz50 = rgba(255,255,255,0.5);
```

An anderer Stelle können Sie die Variable dann benutzen wie einen normalen Wert:

```
#div {  
  background-color: $schwarz50;  
}
```

Sass macht daraus wieder eine standardkonforme CSS-Regel:

```
#div {  
  background-color: rgba(255,255,255,0.5);  
}
```

Sinnvoll ist das erst, wenn Sie den gleichen Wert auch mehrmals in Ihrem Stylesheet verwenden wollen. In dem Fall kann es allerdings eine sehr große Hilfe sein, weil Sie die Definition nur noch an einer Stelle anzupassen brauchen.



Kurz vor Fertigstellung dieses Kapitels wurde der Entwurf für ein neues CSS-Modul namens *CSS Variables* veröffentlicht. Hiermit soll es möglich sein, mithilfe einer neuen data-Eigenschaft Werte zu definieren und im Stylesheet wiederzuverwenden.

Ein weiterer Vorteil ist die Möglichkeit, Regeln ineinander zu verschachteln. Anstatt zwei separate Regeln für Listen und die enthaltenen Einträge zu formulieren, können Sie schreiben:

```
ul {  
  margin: 2em 0;  
  > li {  
    text-align: right;  
  }  
}
```

Sass macht daraus:

```
ul {  
  margin: 2em 0;  
}  
  
ul > li {  
  text-align: right;  
}
```

Präprozessoren sind normalerweise keine eigenständigen Programme. Damit sie funktionieren, benötigen sie eine Skriptsprachenumgebung wie beispielsweise Ruby oder PHP, die zuvor auf Ihrem Rechner installiert werden muss.

(Auf der Website haben Sie die Möglichkeit, die Sprache ohne größeren Aufwand schon einmal zu testen. Die Adresse finden Sie im Abschnitt »Siehe auch«.)

Diskussion

CSS-Präprozessoren (CPP) können Ihnen eine Menge Arbeit ersparen. Allerdings ist hierfür zusätzliches Fachwissen nötig, das Ihnen jetzt näherzubringen den Rahmen dieses Buchs sprengen würde.

Die in der Lösung aufgeführten Beispiele zeigen nur einen Bruchteil der Dinge, die mit CPP möglich sind. Hierzu gehören die Möglichkeiten, Variablen zu definieren, Regeln zu verschachteln und komplexe Berechnungen durchzuführen. Diese Merkmale (und viele mehr) sollen verhindern, dass Sie die gleichen Anweisungen ständig wiederholen müssen.

Neben dem hier gezeigten Sass gibt es noch eine Reihe weiterer CSS-Präprozessoren, wie beispielsweise *Less CSS*. Diese unterscheiden sich weniger in den Funktionen als in der Syntax und in der Art, wie sie implementiert wurden.

Im Abschnitt »Siehe auch« finden Sie eine Reihe von Links zu den wichtigsten CSS-Präprozessoren.

Siehe auch

Die Website für den CSS-Präprozessor *Sass* unter <http://sass-lang.com>; die Möglichkeit, *Sass* online auszuprobieren, unter <http://sass-lang.com/try.html>; *Less CSS* unter <http://lesscss.org/>; eine Liste mit acht verschiedenen CSS-Präprozessoren unter <http://www>.

catswhocode.com/blog/8-css-preprocessors-to-speed-up-development-time; den Entwurf eines CSS3-Moduls für CSS-Variablen unter <http://dev.w3.org/csswg/css-variables/>.

13.18 Älteren Browsern mit Polyfills auf die Sprünge helfen

Problem

Auch ältere Versionen des Internet Explorer sollen bestimmte Fähigkeiten von CSS3, z.B. erweiterte Selektoren, abgerundete Ecken, Schattenwürfe usw. unterstützen.

Lösung

Verwenden Sie sogenannte *Polyfills* wie *CSS3 PIE* und *selectivizr*, um ältere Versionen des IE (und möglicherweise auch andere Browser) mit CSS3-Fähigkeiten auszustatten. Die Polyfills decken meist nur einen Teilaspekt ab, daher ist es unter Umständen sinnvoll, sie miteinander zu kombinieren. Hier zwei der gängigsten:

CSS3 PIE

CSS3 PIE ermöglicht älteren IE-Versionen die Darstellung und Verarbeitung neuer CSS3-Eigenschaften. Speziell geht es hierbei um Gestaltungsmerkmale wie abgerundete Ecken, Schattenwürfe usw. Die folgenden Dinge werden unterstützt: `border-radius`, `box-shadow`, `border-image`, mehrere Hintergrundbilder für das gleiche Element, Farbverläufe, `rgba()`-Farbwerte und einige PIE-eigene Funktionen.

Um *CSS3 PIE* zu nutzen, müssen Sie es zunächst auf Ihrem Webserver installieren. Laden Sie sich hierfür die Datei *pie.htc* von der PIE-Website herunter und legen Sie sie in ein Verzeichnis Ihrer Wahl. (Wir haben hier das Verzeichnis `/scripts/` im Wurzelverzeichnis des Webserver gewählt.) Danach können Sie es wie folgt in Ihr Stylesheet einbinden:

```
.runde-ecken {  
    -moz-border-radius: 15px;  
    -webkit-border-radius: 15px;  
    border-radius: 15px;  
    behaviour: url("/scripts/pie.htc");  
}
```



Die Eigenschaft `behaviour` ist nicht standardkonform und funktioniert nur im Internet Explorer. Soll Ihr Stylesheet validieren, sollten Sie diesen Teil in ein separates Stylesheet auslagern.

Die Pfadangaben für `behaviour` müssen entweder, wie hier gezeigt, auf das Wurzelverzeichnis bezogen (mit vorangestelltem Schrägstrich) oder relativ zum HTML-Dokument, das die CSS-Dateien einbindet, angegeben werden. Nähere Informationen zur Verwendung von *CSS3 PIE* finden Sie auf der Projekt-Website.

selectivizr

Die JavaScript-Bibliothek *selectivizr* ermöglicht die Benutzung der neuen CSS3-Selektoren in älteren Versionen des Internet Explorer.

Im Gegensatz zu *CSS3 PIE* ist *selectivizr* keine eigenständige Lösung, sondern benötigt ein JavaScript-Framework wie *jQuery*, *Mootools* oder *NWMatcher*. Um *selectivizr* zu benutzen, müssen Sie das Framework Ihrer Wahl und die Bibliothek selbst in Ihre Seiten einbinden. Wie *CSS3 PIE* muss die *selectivizr*-Bibliothek allerdings auf Ihrem Webserver installiert sein. Auch hier verwenden wir wieder das Verzeichnis */scripts/*:

```
<script type="text/javascript" src="http://s3.amazonaws.com/nwapi/nwmatcher/nwmatcher-1.2.4-min.js"></script>
<!--[if (gte IE 6)&(lte IE 8)]>
  <script type="text/javascript" src="/scripts/selectivizr.js"></script>
  <noscript><link rel="stylesheet" href="/css/kein_selectivizr.css" /></noscript>
<![endif]-->

<style type='text/css'>
[title*="SommerMusikFest"] {
    border: 3px solid green;
}
</style>
```

In der ersten Zeile binden wir hier das Framework *NWMatcher* per »Hotlink« ein. Es liegt also nicht auf unserem Server, sondern wird direkt aus der Cloud geladen. Für den nächsten Schritt verwenden wir Conditional Comments (siehe Rezept 13.13), um die Verwendung von *selectivizr* nur auf den IE in den Versionen 6 bis 8 zu beschränken.

Im *noscript*-Tag geben wir außerdem ein Notfall-Stylesheet an, das verwendet wird, falls es Probleme mit der Verarbeitung von JavaScript gibt oder die Bibliothek aus anderen Gründen nicht geladen werden konnte.

Danach können Sie, je nachdem, welches Framework Sie verwenden, einen Großteil der neuen CSS3-Selektoren benutzen, als würde der Internet Explorer sie von sich aus unterstützen. Eine Tabelle aller unterstützten Selektoren und den Einschränkungen der verschiedenen Frameworks finden Sie auf der *selectivizr*-Website.

Diskussion

Der englische Webentwickler Remy Sharp definiert einen Polyfill als »...*ein Stück Code (oder Plug-in), das die Technologie zur Verfügung stellt, die der Browser Ihrer Ansicht (als Entwickler) eigentlich von sich aus bereitstellen sollte.*«

Und genau das sind *CSS3 PIE* und *selectivizr*. Beide stellen fehlende Funktionalität zur Verfügung, die ältere Browser von sich aus nicht beherrschen.

Während sich *CSS3 PIE* hierbei auf die Darstellung der beliebtesten CSS3-Effekte im Internet Explorer konzentriert, ermöglicht *selectivizr* die Benutzung neuer CSS-Selektoren. Damit haben Sie zwei mächtige Werkzeuge an der Hand, mit denen Sie Ihre Websites auch ohne viel zusätzlichen CSS-Code auf den neuesten Stand bringen können.

Damit ist die Liste der Polyfills aber noch längst nicht erschöpft. Mittlerweile gibt es für fast jeden Zweck das passende Werkzeug. Eine Auswahl finden Sie im Abschnitt »Siehe auch«.



Gelegentlich ist es aus technischen, organisatorischen oder anderen Gründen nicht möglich, auf eine neue Version des Internet Explorer zu aktualisieren. Hier kann unter Umständen das Browser-Plug-in *Chrome Frame* von Google helfen. Es erweitert ältere IE-Versionen um Fähigkeiten, die sonst nur in aktuellen Versionen von Google Chrome zur Verfügung stehen.

Damit *Chrome Frame* funktioniert, müssen Website-Betreiber allerdings das folgende meta-Tag in ihre Seiten einfügen: `<meta http-equiv="X-UA-Compatible" content="chrome=1">`

Allerdings sollten Sie Polyfills nicht einfach auf gut Glück laden. Das kann schnell zu Performanceeinbußen und höheren Ladezeiten führen. Am besten verwenden Sie hier ein Werkzeug wie *Modernizr.js* oder *jepnope.js*, um zu ermitteln, welche Fähigkeiten der Browser von sich aus hat. Danach müssen Sie nur die Polyfills laden, die tatsächlich gebraucht werden. Umfassende Informationen zur Verwendung von Modernizr zusammen mit Polyfills finden Sie unter <http://www.modernizr.com/docs/#polyfills>.

Siehe auch

Die Website von *CSS3 PIE* unter <http://csspie.com>; die Website von *selectivizr* unter <http://selectivizr.com>; die Website für *Modernizr* unter <http://www.modernizr.com>; die Website für die *Yepnope*-JavaScript-Bibliothek unter <http://yepnopejs.com/>; das Browser-Plug-in *Chrome Frame* unter <http://code.google.com/intl/de-DE/chrome/chromeframe/>; den Polyfill *css3-mediaqueries-js*, mit dem die Verwendung von Media Queries auch in älteren Browsern möglich wird, unter <http://css3-mediaqueries-js.googlecode.com/>; den Polyfill *Prefixfree* von Lea Verou, der die Verwendung von Präfixeigenschaften erleichtert, unter <http://leaverou.github.com/prefixfree/>; den Polyfill *Flexie*, der Browsern das Flexbox-Layout-Modul (siehe Rezept 11.11) beibringt, unter <http://flexiejs.com/>; eine umfassende Liste mit Polyfills für die verschiedensten Aufgaben unter <https://github.com/Modernizr/Modernizr/wiki/HTML5-Cross-browser-Polyfills>; eine Liste mit CSS3-Features mit Verweisen auf die passenden Polyfills unter <http://css3clickchart.com/>; den Artikel *10 ways to make Internet Explorer act like a modern browser* von Jean-Baptiste Jung unter <http://www.catswhocode.com/blog/10-ways-to-make-internet-explorer-act-like-a-modern-browser>.

13.19 CSS3-Eigenschaften in der Präfixschreibweise angeben

Problem

Sie möchten CSS-Fähigkeiten nutzen, die noch nicht vollständig implementiert sind, die in der CSS3-Spezifikation noch nicht endgültig festgelegt sind oder die nur für bestimmte Browser/Plattformen zur Verfügung stehen.

Lösung

Verwenden Sie die entsprechenden Präfixschreibweise. Hier ein Beispiel für die CSS3-Eigenschaft transform:

```
#element {  
  -webkit-transform: translateX(5em) translateY(4em);  
  -moz-transform: translateX(5em) translateY(4em);  
  -ms-transform: translateX(5em) translateY(4em);  
  -o-transform: translateX(5em) translateY(4em);  
  transform: translateX(5em) translateY(4em);  
}
```

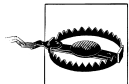
Durch diese Regel wird das #element um 5em nach links und 4em nach unten verschoben.

Diskussion

In den Modulen der CSS3-Spezifikation wird eine Reihe neuer Eigenschaften definiert. Damit Browserhersteller deren Implementierung »in freier Wildbahn« testen können, ohne dass Änderungen Probleme verursachen, werden die Eigenschaften zunächst in der Präfixschreibweise umgesetzt.

Dadurch können Sie einige Merkmale von CSS3 bereits nutzen, obwohl diese noch nicht standardisiert sind. Sobald sich die Spezifikation gefestigt hat, werden die Präfixe in neueren Browserversionen dann entfernt.

Die Präfixschreibweise gibt es für alle gängigen Browser. Für Safari und Chrome sowie den Android-Browser gilt das Präfix -webkit, für Firefox und dessen Verwandte -moz; Opera verwendet -o als Präfix, und für den Internet Explorer gibt es -ms. Selbst für den Linux-Browser Konqueror gibt es das Präfix -khtml.



Während dieses Kapitel entstand, wandte sich Daniel Glazman, einer der Vorsitzenden der CSS-Arbeitsgruppe des W3C, an die Gemeinschaft der Webentwickler. Er warnt davor, nur -webkit- als Präfix zu verwenden. Allzu schnell könnte es passieren, dass eine neue Monokultur, wie vor zehn Jahren mit dem Internet Explorer 6, entsteht. Tun Sie also sich und dem freien Web einen Gefallen und verwenden Sie alle Präfixvarianten, die nötig sind, um zumindest die fünf wichtigsten Browser zu unterstützen.

Je nachdem, um welche Eigenschaft und um welchen Browser es sich handelt, sind unterschiedliche Präfixe nötig, um möglichst viele Browser zu unterstützen. Teilweise kann es sogar sein, dass Sie die Präfixschreibweise in zwei verschiedenen Versionen angeben müssen, um alle Browser zu unterstützen, weil sich die Implementierung im Laufe der Zeit geändert hat. Wo das in diesem Buch der Fall ist, weisen wir darauf hin.

Nach den Eigenschaften in der Präfixschreibweise sollten Sie am Schluss des Regelsatzes immer auch die tatsächliche in der Spezifikation definierte Eigenschaft angeben. So ist sichergestellt, dass Ihre Stylesheet-Regeln auch dann noch funktionieren, wenn die Browserhersteller die Unterstützung für die Präfixvariante entfernen.

Wenn ein Browser eine Zeit lang beide Schreibweisen unterstützt, kann es außerdem passieren, dass eine möglicherweise fehlerhafte Präfixversion bevorzugt wird.

Es ist kaum möglich, sich zu merken, welche Browserversion eine bestimmte Eigenschaft nun unterstützt oder nicht. Zum Glück gibt es die Website *When can I use*. Hier finden Sie zu allen aktuellen Browsern Tabellen, in denen Sie schnell finden, wie es um die Unterstützung bestimmter CSS3-Eigenschaften (und HTML5-Features) bestellt ist. Außerdem wird angezeigt, bis zu welcher Version gegebenenfalls die Präfixschreibweise verwendet werden muss.

Standardmäßig werden nur die letzten drei Versionen eines Browsers angezeigt. Da besonders Chrome und Firefox sehr schnelle Versionswechsel haben, lohnt es sich, auf den Link *Show all versions* zu klicken, um auch ältere Versionen anzuzeigen. Eine Möglichkeit, die Verbreitung oder das Alter eines Browsers herauszufinden fehlen leider.

Siehe auch

Die *When can I use...*-Website zum Überprüfen, welcher Browser in welcher Version bestimmte Eigenschaften mit oder ohne Präfix unterstützt, unter <http://caniuse.com>; das CSS3-Modul *Syntax and basic data types* zur Präfixschreibweise unter <http://www.w3.org/TR/CSS2/syntax.html#vendor-keywords>; eine umfangreiche Liste mit Präfixeigenschaften und -werten unter <http://peter.sh/experiments/vendor-prefixed-css-property-overview/>; ein Polyfill von Lea Verou, das die nötigen Präfixvarianten automatisch einfügt, unter <http://leaverou.github.com/prefixfree/>; ein Onlinewerkzeug zum Erstellen der nötigen Präfixversionen einer Eigenschaft unter <http://prefixr.com/>; den Artikel von Daniel Glazman unter <http://tinyurl.com/glazman-prefix>.

13.20 Daten-URLs verwenden

Problem

Sie möchten die Anzahl der HTTP-Requests verringern, indem das Laden von kleinen Bildern und anderen Dateien vermieden wird.

Lösung

Verwenden Sie Daten-URIs, um beispielsweise Sprites direkt in die CSS-Datei einzubetten:

```
a:link[href^="http"] {  
    padding-right:    20px;  
    background-position: right -20px;  
    background-repeat: no-repeat;  
    background-image: url("data:image/png;base64,  
    iVBORwOKGgoAAAANSUhEugAAABIAAADYCAAAADRY003AAABoVBMVEX%2F% ...");  
}
```

Diese Regel sorgt dafür, dass nach einem externen Link ein Symbol angezeigt wird. Anstatt hierfür eine Grafikdatei zu laden, wurde diese mithilfe eines Daten-URI direkt in die CSS-Datei eingebettet. Das Ergebnis sehen Sie in Abbildung 13-10.

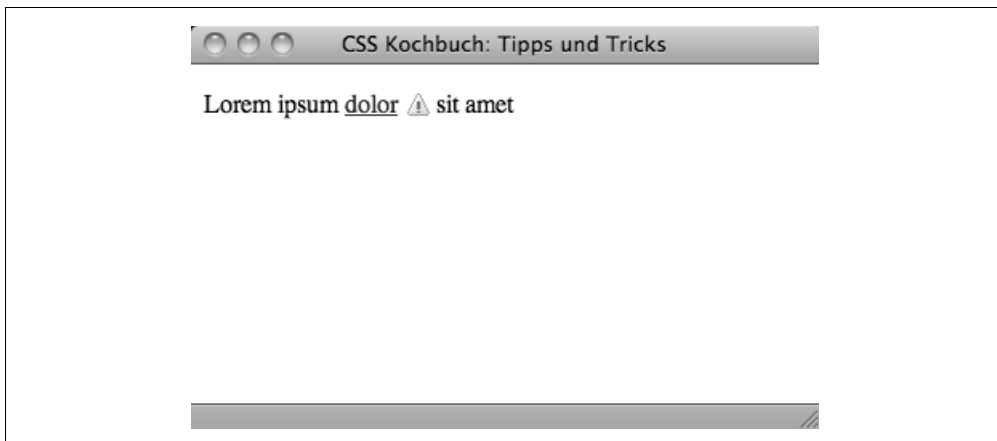


Abbildung 13-10: Nach dem externen Link wurde eine per Daten-URI gespeicherte Grafik eingebunden

Für die Definition von Daten-URIs kommt die CSS-Funktion `url()` zum Einsatz. Nur werden anstelle eines Dateipfads die Daten direkt im Stylesheet gespeichert. Damit der Browser das richtig versteht, beginnt der URI mit dem Schlüsselwort `data`. Nach einem Doppelpunkt folgt der MIME-Typ (`image/png`) und, durch ein Semikolon getrennt, die Kodierung (`base64`). Nach einem Komma folgen die eigentlichen Daten.

Diskussion

Daten-URIs können helfen, die Ladezeit Ihrer Webseiten zu verringern. Anstatt für jede noch so kleine Datei (Icons, Sprites usw.) eine eigene HTTP-Abfrage zu starten, werden die Grafiken direkt in das Stylesheet eingebettet.

Sobald es länger dauert, eine externe Datei anzufordern, als diese im Stylesheet eingebettet zu übertragen, lohnt sich die Verwendung von Daten-URIs. Laut der Google-Entwickler-Website ist dieses Verfahren bis zu einer maximalen Dateigröße von 10 kB sinnvoll.

Daten-URIs können überall dort verwendet werden, wo die `url()`-Funktion benutzt wird. Neben Grafiken können also beispielsweise auch Schriften geladen werden.

Um eine Datei ins Stylesheet einzubetten, muss sie korrekt kodiert werden. In der Regel kommt hierfür entweder die URL-Kodierung oder das Format *base-64* zum Einsatz. Am besten verwenden Sie hierfür eines der unter »Siehe auch« genannten Onlinewerkzeuge.

Bei der Verwendung der URL-Kodierung sollten Sie die Zeichenkette auf jeden Fall mit doppelten Anführungszeichen umgeben, weil in den Daten möglicherweise einfache Anführungszeichen enthalten sein können. Die Verwendung eines Texteditors mit Syntax-Highlighting kann helfen, Fehler mit Anführungszeichen zu finden.

Sicherer ist das base-64-Format, da hier mit Sicherheit keine Anführungszeichen enthalten sind. Dafür sind base-64-kodierte Daten ungefähr ein Drittel größer als URL-kodierte.

Die Browserunterstützung für Daten-URIs ist mittlerweile recht gut. Nur der Internet Explorer 7 spielt nicht mit. Hier ist es möglicherweise sinnvoll, mehrere `url()`-Funktionen zu schichten:

```
a:link[href^="http"] {  
    background-image: url('pfad/zur/datei.png');  
    background-image: url("data:image/png;base64,  
        iVBORw0KGgoAAAANSUhEUgAAABIAAADYCAMAAADRY003AAABoVBMVEX%2F% ...");  
}
```

Kann die zweite Deklaration nicht interpretiert werden, verwendet der Browser (z.B. IE7) automatisch die erste Variante. Der Internet Explorer 8 kann mit Daten-URIs umgehen, allerdings nur bis zu einer Länge von 32 KByte. Ab Version 9 liegt das Limit bei satten 4 GByte. Das sollte reichen!



In Firefox 5 (Juni 2011) dürfen Daten-URIs keine Newline-Zeichen enthalten.

Soll die gleiche Grafik für mehrere Elemente verwendet werden, müssen Sie die Datei nicht mehrmals einbinden. Stattdessen können Sie gruppierte Selektoren (siehe Rezept 1.4) einsetzen.

Siehe auch

Ein Onlinewerkzeug zum Konvertieren von Bildern in Daten-URIs unter <http://yellow-green.de/image-to-base64-string-encoder>; einen Artikel von *Fontspring.com*, der sich (unter anderem) mit dem Einbetten von Schriften als Daten-URIs befasst, unter <http://www.fontspring.com/blog/the-new-bulletproof-font-face-syntax>; eine alternative Möglichkeit für ältere Versionen des Internet Explorer (7 und darunter), Daten einzubetten, unter <http://www.phpied.com/mhtml-when-you-need-data-uris-in-ie7-and-under/>; den Artikel *Data URIs* von Chris Coyier unter <http://css-tricks.com/data-uris/>; den Artikel

Minimize Request Overhead unter <http://code.google.com/intl/de-DE/speed/page-speed/docs/request.html>; eine Anleitung zur Erstellung und Benutzung von Daten-URIs inklusive der passenden Konfiguration des Apache-Webservers unter <http://www.7cynics.com/webdesign/css/css-inline-images-data-uri.html>; das Rezept 2.4 zur Verwendung der `url()`-Funktion.

Ressourcen

Bei der Arbeit mit Cascading Style Sheets sollten Sie zwei Dinge beachten: Halten Sie Ihren Code einfach, indem Sie nur die Selektoren und Eigenschaften benutzen, die Sie wirklich brauchen – zu viele überflüssige Erweiterungen könnten später für Verwirrung sorgen – und verifizieren Sie Ihren HTML- und CSS-Code mithilfe der Validator-Programme des W3C unter <http://jigsaw.w3.org/css-validator/> (CSS) bzw. <http://validator.w3.org/> (HTML). Weitere nützliche Tipps finden Sie außerdem in Kapitel 13, *Tipps und Tricks*, dieses Buchs.

Wenn Sie beides berücksichtigen, können Sie die meisten Probleme, auf die Sie als Entwickler stoßen werden, lösen. Sollten Sie trotzdem einmal Schwierigkeiten haben, finden Sie in diesem Anhang die wichtigsten Referenzen, Diskussionsgruppen und Werkzeuge im Internet, die Ihnen bei Ihrer Arbeit mit CSS helfen können.

Eigenschaften und Selektoren

Liste aller CSS-Eigenschaften

<http://www.css-wiki.com/>

Diese von Stephan Heller geschaffene Website enthält eine umfassende, alphabetisch geordnete Liste der aktuellen CSS-Eigenschaften. Auf Wunsch können nur CSS3- oder CSS 2.1-Eigenschaften angezeigt werden.

CSS-Selektoren

<http://www.w3.org/TR/selectors/#selectors>

Eine Liste aller Selektoren in CSS 2.1 und CSS3 mit einer kurzen Erklärung ihrer Funktion.

<http://tinyurl.com/nettuts-selectors>

Eine Aufstellung von 30 wichtigen CSS-Selektoren (2.1 und 3), inklusive Kurzanleitung und Live-Beispielen.

CSS-Anleitungen

CSS-Anleitung von »Web Page Design for Designers«

<http://www.wpdfd.com/editorial/basics/index.html>

Diese Anleitung hilft Webentwicklern beim Einstieg in die Arbeit mit CSS.

Grundlagen der Positionierung von Elementen mit CSS

<http://www.communitymx.com/content/article.cfm?cid=3B56F&print=true>

Weitere Informationen zur Positionierung von Elementen mit CSS finden Sie in dieser Anleitung von Community MX (einer Website für Benutzer von Werkzeugen wie Dreamweaver, Flash usw.)

Anleitung für die Verwendung von CSS-Floats

<http://css.maxdesign.com.au/floatutorial/index.htm>

Hier können Sie lernen, wie Sie mit CSS als Float definierte Elemente verwenden können. Mit dabei ist eine Reihe von Übungen und Codebeispielen.

Anleitung für CSS-Selektoren

<http://css.maxdesign.com.au/selectutorial/index.htm>

Mit dieser Anleitung können Sie Ihr Verständnis von CSS-Selektoren verbessern. Teil der Anleitung ist ein Beispiel für die Verwendung von Selektoren zum Erstellen eines dreispaltigen Layouts.

Designressourcen

A List Apart

<http://www.alistapart.com/topics/code/css/>

Von dieser sehr empfehlenswerten Website von A List Apart stammen die meisten Artikel von Webdesignern, die hier ihre Gedanken und Neuentwicklungen der Öffentlichkeit mitteilen.

CSS Zen Garden

<http://www.csszengarden.com/tr/deutsch/>

Im CSS Zen Garden präsentieren sich Webdesigner aus aller Welt. Anhand einer vorgegebenen Seitenstruktur entwickeln sie nur mit CSS immer neue Layouts. Mittlerweile etwas betagt, aber immer noch eine gute Inspirationsquelle.

CSS-Layouttechniken von Glish.com

<http://www.glish.com/css/>

Eine umfangreiche Sammlung mehrspaltiger Layouts mit CSS.

Real World Style

<http://www.realworldstyle.com/>

Eine Designressource von Mark Newhouse. Das Ziel dieser Site ist die Förderung von CSS-basierten Designs nicht nur für populäre Browser, die auf Macintosh- und Windows-Rechnern laufen, sondern auch für Browser auf Unix-Rechnern.

Typetester

<http://www.typetester.org>

Ein vielseitiges Werkzeug, mit dem Webentwickler drei Schriftarten auswählen und ihre Darstellung auf einer Webseite vergleichen können. Das Werkzeug erzeugt einfache CSS-Regeln für die gewählten Schriften, die per Kopieren und Einfügen direkt weiterverwendet werden können. Neben der Schriftart selbst können auch Werte für Schriftgröße, Zeilenhöhe, Wortabstände, Textausrichtung und vieles mehr eingestellt werden.

Diskussionsgruppen, Mailinglisten und Foren

css-discuss

<http://www.css-discuss.org/>

Diese von Eric A. Meyer geleitete Mailingliste behandelt die praktische Anwendung von CSS. Eric A. Meyer ist Autor des bei O'Reilly erschienenen Buchs *CSS – Das umfassende Handbuch*.

Forum für XHTML und CSS

<http://xhtmlforum.de/css/>

Ein deutschsprachiges Webforum zum Thema CSS.

www-style (die Stylesheets-Mailingliste des W3C)

<http://lists.w3.org/Archives/Public/www-style/>

In dieser vom World Wide Web Consortium (W3C) betreuten Mailingliste geht es um die Theorie und die Zukunft von CSS. Fragen zur Spezifikation und Vorschläge zu CSS sind willkommen. Diskussionen zur praktischen Anwendung sind hier jedoch nicht gern gesehen.

Referenzen zum Nachschlagen

browsersupport.net

<http://www.browsersupport.net/CSS/>

Eine visuelle Darstellung zur Browserunterstützung für die verschiedenen CSS 2.1- und CSS3-Eigenschaften.

Quirksmode

<http://www.quirksmode.org/compatibility.htm>

Immer noch aktuell: Die Tabellen zur Browserunterstützung von *Quirksmode.org* geben einen umfassenden Überblick über die Unterstützung, speziell von CSS 2.1. Zu vielen Eigenschaften und Selektoren gibt es Testszenarien. Außerdem eine der wichtigsten Quellen zu den Eigenheiten verschiedener Browser, speziell älterer Versionen des IE.

When can I use?

<http://www.caniuse.com>

Eine der wichtigsten Ressourcen zur Browserunterstützung (mit und ohne Präfix) für CSS3, HTML5, SVG und andere Dinge. Teilweise mit weiterführenden Links.

HTML5 und CSS3 Readiness

<http://html5readiness.com/>

Eine Infografik, die verdeutlicht, welche Merkmale von HTML5 und CSS3 von welchen Browserversionen unterstützt werden.

CSS3 Click Chart

<http://css3clickchart.com/>

Eine Website zu CSS3. Mit Beispielcode, Live-Demos, Angaben zur Browserunterstützung, möglichen Polyfills und Links zu passenden Anleitungen.

HTML5 Please

<http://html5please.com/>

Angaben zur verantwortungsvollen Verwendung der neuen Fähigkeiten von CSS3 und HTML5. Mit Informationen zu Alternativen, Polyfills, nötigen Präfixschreibweisen usw.

Die CSS-Seiten des W3C

<http://www.w3.org/Style/CSS/>

Dies ist die offizielle CSS-Website. Hier finden Sie Informationen zur Geschichte von CSS, Ressourcen zum Lernen von CSS, eine Liste mit Werkzeugen zum Schreiben von CSS-Code und Neuigkeiten zum Thema.

Die CSS3-Spezifikation

<http://www.css3.info/modules/>

Eine Übersicht über die Module der CSS3-Spezifikation und ihren jeweiligen Entwicklungsstand. Außerdem eine gute allgemeine Ressource zu CSS3.

Die CSS 2.1-Spezifikation

<http://www.w3.org/TR/CSS21/>

Manchmal sind die Implementierungen der CSS-Spezifikation durch Benutzerprogramme ziemlich schwer zu durchschauen. Wenn Sie wissen wollen, wie Sie ein bestimmtes Erscheinungsbild für Ihre Seiten erstellen können, oder versuchen, einen Fehler in der Implementierung zu finden, sollten Sie auf jeden Fall auch einen Blick in die Spezifikation (und die Tabellen zur CSS-Unterstützung) werfen. (Eine »offizielle« deutsche Übersetzung der CSS2-Spezifikation finden Sie unter <http://www.edition-w3c.de/TR/REC-CSS2.1/>.)

Die HTML5-Spezifikation

<http://www.w3.org/TR/html5/>

Die aktuelle Version der HTML5-Spezifikation.

<http://www.w3.org/TR/html5-author/>

Die HTML5-Spezifikation in einer speziellen Version für Webautoren (Webworker, -designer usw.).

Accessible Rich Internet Applications

<http://www.w3.org/TR/wai-aria/>

Richtlinien des W3C zur Barrierefreiheit von Websites.

Barrierefreie Informationstechnik Verordnung (BITV)

<http://www.einfach-fuer-alle.de/artikel/bitv/>

Richtlinien zur Barrierefreiheit von Websites in Deutschland des deutschen Innenministeriums.

Mikroformate

<http://www.microformats.org/>

Definiert und fördert Standards für die Erstellung bestimmter Inhalte. Auf der Website finden Sie beispielsweise eine Liste mit Methoden für die Programmierung und Gestaltung häufig vorkommender Seiteninhalte wie Veranstaltungskalender, Kontaktinformationen und sogar des Elements abbr.

Werkzeuge

Der CSS-Validator des W3C

<http://jigsaw.w3.org/css-validator/>

Dieser kostenlose Dienst, der vom W3C betrieben wird, überprüft CSS-Code auf seine korrekte Struktur. Sie können Ihr Stylesheet als Datei hochladen, eine Webadresse in das Formular schreiben oder den CSS-Code direkt in ein Formularfeld eingeben. Es werden verschiedene Versionen der CSS-Spezifikation, inklusive CSS3 und sogar Präfixschreibweisen, unterstützt.

Der HTML-Validator des W3C

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

Der HTML-Validator des W3C ist ein weiterer kostenloser Dienst des W3C. Auf ähnliche Weise wie der CSS-Validator überprüft der HTML-Validator, ob Ihr Markup den Webstandards entspricht.

Webpagetest

<http://www.webpagetest.org/>

Ein Werkzeug zum Überprüfen der Darstellung von Websites mit verschiedenen Browsern, speziell älteren Versionen des Internet Explorer.

CSS3 Generator

<http://css3generator.com/>

Ein Werkzeug zur Erstellung von CSS3-Code, inklusive der nötigen Präfixschreibweisen.

Byte Size Matters

<http://bytesizematters.com/>

Ein Werkzeug zum Messen der Größe von Stylesheets (und anderen Textdateien).

Statcounter Global Stats

<http://gs.statcounter.com/#browser-DE-monthly-201104-201204>

Umfangreiche Statistiken zur Verwendung verschiedener Browser. Möglichkeit zur Begrenzung der Auswahl auf verschiedene Regionen (z.B. Deutschland, Europa), Zeiträume, Darstellung usw.

SelectORacle

<http://gallery.theopalgroup.com/selectoracle/>

Dieser kostenlose Dienst soll dabei helfen, komplexe CSS-Selektoren zu verstehen, indem ihre Bedeutung in verständliches Englisch übersetzt wird. Die Selektoren können auf verschiedene Arten eingegeben werden: Entweder Sie geben den fraglichen Selektor durch Kopieren und Einfügen in das Formular ein, oder Sie geben die URL einer Webseite an, die ein eingebettetes Stylesheet enthält, oder Sie geben direkt die URL eines Stylesheets ein. Durch ein Auswahlmenü können auch andere Sprachen – unter anderem Deutsch – ausgewählt werden, die aber teilweise noch nicht vollständig unterstützt werden.

Weiterführende Ressourcen

Förderung von Webstandards im deutschsprachigen Raum

<http://www.webkrauts.de>

Eine Website, die sich zum Ziel gesetzt hat, »Webstandards auch im deutschsprachigen Raum stärker zur Geltung zu bringen«. Eine Reihe lesenswerter Artikel zu verschiedenen Aspekten von CSS, aber auch zur Verwendung von HTML5.

Jens O. Meiert

<http://meiert.com/>

Eine Reihe nützlicher und hilfreicher Artikel von Jens O. Meiert zur richtigen Verwendung von CSS und HTML.

456 Berea Street

<http://www.456bereastreet.com/>

Die Website von Roger Johansson. Artikel und Kommentare zu Webstandards, speziell zum Thema Zugänglichkeit.

CSS Tricks

<http://css-tricks.com/>

Eine umfassende Sammlung von Anleitungen zur Arbeit mit CSS aus den verschiedensten Bereichen.

Standardista

<http://www.standardista.com/>

Die Website von Estelle Weyl zu den Themen Webstandards, CSS3, HTML5 und JavaScript.

Lea Verou

<http://lea.verou.me/>

Lesenswerte Artikel und Werkzeuge für die Arbeit mit CSS3 von Lea Verou.

Smashing Magazine

<http://www.smashingmagazine.com/>

Hervorragend recherchierte Artikel rund um Webprogrammierung, -grafiken und -design.

Weiterführende Bücher

CSS – Das umfassende Handbuch, 2. Auflage (deutsche Ausgabe der 3. engl. Auflage)

von Eric A. Meyer, O'Reilly 2007, ISBN 978-3-89721-493-4

CSS – kurz & gut, 4. Auflage

von Eric A. Meyer, O'Reilly 2011, ISBN 978-3-86899-144-4

HTML & XHTML – kurz & gut, 4. Auflage

von Jennifer Niederst Robbins, O'Reilly 2010, ISBN 978-3-89721-558-0

CSS3 for Web Designers

von Dan Cederholm, A Book Apart 2010, ISBN 978-0-9844425-2-2

HTML5 for Web Designers

von Jeremy Keith, A Book Apart 2010, ISBN 978-0-9844425-0-8

Responsive Web Design

von Ethan Marcotte, A Book Apart 2010, ISBN 978-0-9844425-7-7

Mobile First

von Luke Wroblewski, A Book Apart 2011, ISBN 978-1-937557-02-7

Barrierefreiheit verstehen und umsetzen – Webstandards für ein zugängliches und nutzbares Internet

von Jan Eric Hellbusch und Kerstin Probiesch, dpunkt Verlag 2011,
ISBN 978-3-89864-520-1

Symbole

(Doppelkreuz), für id-Selektoren 29
& (Ampersand), aus anderem Zeichensatz
einfügen 142
* (Asterisk), universeller Selektor 13–14, 337,
345, 510
+ (Pluszeichen), Selektoren
benachbarte Geschwisterelemente 20, 318,
321, 324
. (Punkt), zum Kennzeichnen von Klassen-
Selektoren 27
: (Doppelpunkt)
für Pseudoelemente 488
zum Markieren von Pseudoklassen und
-elementen 31
zur Trennung von Eigenschaft und Wert 6
:: (doppelter Doppelpunkt)
für Pseudoelemente in CSS3 34, 488
; (Semikolon), zur Trennung von Deklarationen 6
> (schließende spitze Klammer), Selektor für
Kindelemente 18, 375
@import-Regel
für Mobilgeräte 356
\ (Backslash), für hexadezimale Werte 340
\ (Backslash), hexadezimale Unicode-Werte
schützen 196
^ (Caret-Zeichen), in Attribut-Selektor 339
{ } (geschweifte Klammern) 6
| (senkrechter Strich), horizontale Menüs 207
/ (Schrägstrich), zum Trennen von font-family
und line-height in font-Kurzschrift-
eigenschaft 126
, (Komma), Selektoren trennen mit 12, 13
~ (Tilde), Selektor für allgemeine Geschwister-
elemente 22
1140 CSS Grid (Layout-Framework) 414
1kb CSS Grid (Layout-Framework) 411, 413
960.gs (Layout-Framework) 414

A

A List Apart 522
abgerundete Ecken 298, 438–442
Browserunterstützung 439
für Bilder 441
grafikbasierte Lösung 439
spezielle Präfix-Schreibweise für Firefox 440
absolute Längeneinheiten 58
absolute Maßeinheiten, für Druckversion 345
absolute Positionierung 90–92, 176, 262, 497
absolute Positionierung, für Listeneinträge in
Dropdown-Navigationsmenüs 245
abwechselnde Hintergründe für Tabellen-
zeilen 324–326
:active-Pseudoklasse 32, 212–214, 280,
495–497
a-Element 211–250
:after-Pseudoelement 32, 144, 164, 181, 338,
353, 486, 494
allgemeine Schriftfamilien 112
all-Medientyp 55
Alphakanal 290, 435
alt-Attribut 178
alternative Cursor-Symbole definieren 461–462
analoges Farbschema 425
Anführungszeichen
bei Schriftnamen 112
doppelte, für Daten-URIs 519
in content-Eigenschaft 339
in url()-Funktion 65
typografisch korrekt 143–146, 162
Animationen 454–461
animierbare Eigenschaften (CSS3) 457
Polyfill (jQuery) 461
schrittweise 460
Ansichtsbereich 94, 360
als Positionierungskontext 90
Bilder skalieren für 170–172

- Elemente verankern in 95
- ganze Seite bei Druckversion 336
- in Media Queries 357
- Asterisk (*), universeller Selektor 14, 337, 510
- Ästhetik
 - als Grundlage für Design, anstelle von technischen Aspekten 465
- attr()-Funktion 339
- Attribute
 - alt 178
 - autofocus 282
 - border, für Tabellen 331
 - border-color, für Tabellen 331
 - charset, Verwechslungsgefahr mit @charset-Regel 485
 - checked, für ausgewählte Checkboxes 284
 - class 26, 173, 188, 218, 264, 273, 309, 319, 321, 325, 370
 - HTML-Spezifikation 30
 - Markierung im HTML-Code 27
 - mehrere Klassen angeben 30
 - Unterschied zu id-Attributen 29
 - verständliche Benennung 29
 - colspan, für Tabellen 323
 - darstellen mit attr()-Funktion in content-Eigenschaft 220
 - disabled, für nicht benutzbare Formular-elemente 284
 - enabled, für benutzbare Formular-elemente 284
 - href, als Wert für content-Eigenschaft 338
 - id 26, 188, 215, 223, 227, 303, 309, 370
 - Benennung, verständlich 29
 - für body-Element 230–232
 - Selektor 28
 - Unterschied zu class-Attributen 29
 - verwenden, um Elemente in Druckversion zu verbergen 350
 - Verwendung 27
 - Klassen 26
 - lang, Pseudoklasse 32
 - lang, Selektor 24
 - Namen, Selektor 24
 - optional, für optionale Formularelemente 284
 - placeholder
 - gestalten in Gecko-Browsern 269
 - gestalten in Webkit-Browsern 269
 - in Druckversion ausblenden 347
 - required 283–284
 - required, für erforderliche Formular-elemente 284
 - Selektoren gruppieren 25
 - style 5, 499
 - Teilwert, Selektor 24
 - title 220
 - title, als Wert für content-Eigenschaft 339
 - type
 - checkbox 270
 - radio 270
 - Werte, mit content-Eigenschaft in Webseite einfügen 486
 - Zeichenkette in, Selektoren 25
- Attributselektor 23–26, 146, 161, 217, 276, 278, 499–501
 - für benötigte Formularfelder 283
 - für input-Elemente 265
 - für Substrings 339
 - für verschiedene label-Elemente 264
 - gruppieren 25
 - in CSS3 25
 - reduziert 286
- Aufbau einer CSS-Regel 6–8
- Aufklappenmenüs, select-Element 273
- Aufzählungszeichen
 - andersfarbige 197–198
 - eigene 488
 - für Listen 192–193
 - grafisch 199–201
 - textbasiert 194–197
 - entfernen 206
 - manuell einfügen 196
 - Stildefinitionen für 192
- Ausdrucken verhindern 351
- Außenabstände 66–69
 - anpassen für Formulare 289
 - automatisch 67
 - automatisch, für horizontales Zentrieren 103
 - explizit definieren für fieldset-Elemente 291
 - negativ 67, 163, 403, 409
 - negativ, für horizontale Zentrierung 104
 - vertikal zusammengefasst 67
 - zum beliebigen Anordnen von Layout-Spalten 394
 - zurücksetzen 392, 509
 - zurücksetzen für Mobilgeräte 371
 - zurücksetzen, für zweispaltige Layouts 386

Außenrahmen 213
Auswahllisten (option) 272–274
Auswahllisten, select-Element 273
auto (Wert der Eigenschaft cursor) 462
autofocus-Attribut 282
Autoren-Stylesheets 47

B

background-attachment-Eigenschaft 78–79
 Spezifikation 79
background-color-Eigenschaft 73, 289
 transparent-Schlüsselwort 436
 Werte 61
background-Eigenschaft 86–87, 177, 216–218, 329
 Deklaration wiederholen 254
 doppelt 233
 dreifache Schreibweise 290
 redundante Schreibweise 295
background-image-Eigenschaft 73–74, 183, 289, 400
 Farbverläufe definieren 429
 für Aufzählungszeichen in Listen 200
 grafische Unterstreichungen und 140–142
 mehrere Bilder angeben für 79–82
 Werte 65
background-position-Eigenschaft 76–78, 187, 400, 402
 Besonderheiten für body-Element 78
 Spezifikation 78
 Werte 77
background-repeat-Eigenschaft 75–76, 149, 200, 289, 400
 Werte 75
background-size-Eigenschaft 82–85, 400, 402, 473
Backslash (\), für hexadezimale Werte 340
Backslash (\), hexadezimale Unicode-Werte schützen 196
base-64-Format, für Daten-URIs 519
Baumstruktur von HTML 10
:before-Pseudoelement 32, 144, 164, 181, 194, 195, 197, 205, 207, 220, 338, 340, 377, 486
behaviour (proprietäre Eigenschaft) 440
beliebige Kindelemente, Pseudoklasse
 :nth-child() 33

Beliebigkeitsregel (font-Eigenschaft) 125
benachbarte Geschwisterelemente
 Selektor 20, 291, 295, 314, 318, 349
benannte Farben 62
Benutzbarkeit
 erhöhen durch Rollover-Effekte 423
Benutzerfreundlichkeit 211
Benutzer-Stylesheet 47
Beschriftung von Icons, Schrift des Betriebssystems für 127
Bestandteile einer CSS-Regel 6
besuchte Links
 :visited-Pseudoklasse 32
Beziehungen zwischen Elementen 11
Bildbearbeitungsprogramm, zur Erstellung von Hintergrundbildern 87
Bilder 165–189
 als Aufzählungszeichen 199–201
 als Block-Element definieren 176
 als Hintergrund für Layoutspalten 402
 als Links 168
 als Panorama präsentieren 182–183
 alt-Attribut 178
 an Grundlinienraster anpassen 468
 anstelle von Text anzeigen 178–181
 Bildunterschrift 172–177
 mit HTML5-Elementen 174
 vor dem Bild anzeigen 175
 Darstellungsrolle ändern 308
 doppeln, für Schattenwürfe in IE 7 und 8 445
 Floats und 100
 Formate
 kombinieren 184–187
 PNG 177
 Grundlinie und 176
 halb transparent 436
 Hintergrund 73–74, 183
 als Ersatz für Text 178
 auf Webseiten positionieren 76
 in Druckversion mit darstellen 352–354
 mehrere 79–82
 positionieren 76–78
 skalieren 82–85
 transparent 87
 verankern 78–79
 wiederholen 73, 75–76
 im Stylesheet einbetten, per Daten-URI 517–520

- in Tabellen, Leerraum am unteren Zellenrand entfernen 306–308
- Inline-Element 176
- JPEG, Artefakte bei Skalierung 172
- mit abgerundete Ecken versehen 441
- nicht mit ausdrucken 350
- PNG, für Hintergründe 87
- Rahmen für 165–166
 - entfernen 167–168
- Seitenverhältnis, Skalierung 171
- skalieren 170–172
 - relativ zur Schriftgröße 171
- Skalierung, beschränken 172
- Sprites 187–189
- transparent 434–438
- umfließen lassen 149
- zentrieren 105
- Bildersetzung *siehe* Textersetzung
- Bildformate kombinieren 184–187
- Bildschirmdarstellung, relative Schriftgrößen 114
- Bildschirmfotos
 - von verschiedenen Browsern 477
- Bildunterschriften 172–177
- Block Formatting Context, Floats und 493
- Blockelemente 224–225, 228, 285
 - Bilder als 176
 - Boxmodell und 38
 - Definition von Höhe und Breite 60
 - Dokumentenfluss und 493
 - in Inline-Elementen (nicht erlaubt) 263
 - variable Breite 391
 - zentrieren 103–104
- blockquote-Element 162
- Blueprint CSS (Layout-Framework) 414
- body-Element
 - Besonderheiten für background-position 78
 - Höhe definieren 409
- border-Attribut, für Tabellen 331
- border-bottom-color-Eigenschaft 71
- border-bottom-Eigenschaft 203, 212, 346, 348
 - für Unterstreichungen 140
- border-bottom-left-radius-Eigenschaft 439
- border-bottom-right-radius-Eigenschaft 439
- border-bottom-style-Eigenschaft 71
- border-bottom-width-Eigenschaft 71
- border-collapse-Eigenschaft 303–305, 312, 329
- bordercolor-Attribut, für Tabellen 331
- border-color-Eigenschaft 71, 303
 - Werte 72
- border-Eigenschaft 39, 70–73, 329
 - an Grundlinienraster anpassen 466
 - Auswirkungen von box-sizing 490
 - Druckversion 348
 - für Tabellenzellen 301–305
 - in Formularen durch outline-Eigenschaft ersetzen 495
 - Schlüsselwörter als Werte 72
 - Texteingabefelder und 266, 293
- border-Kurzschrifteigenschaft 165–166, 167–168
- border-left-color-Eigenschaft 71
- border-left-style-Eigenschaft 71
- border-left-width-Eigenschaft 71
- border-radius-Eigenschaft 247, 298, 370, 438–442
 - box-shadow und 443
- Einzeleigenschaften 249
- mit inherit-Wert, für Bilder mit abgerundete Ecken 441
- Vier-Werte-Schreibweise 234
- Zwei-Werte-Schreibweise 440
- border-right-color-Eigenschaft 71
- border-right-style-Eigenschaft 71
- border-right-width-Eigenschaft 71
- border-spacing-Eigenschaft 304, 305–306
- border-style-Eigenschaft 71, 168, 304
 - Werte 72
- border-top-color-Eigenschaft 71
- border-top-Eigenschaft 173, 212
 - für Überstreichungen 140
- border-top-left-radius-Eigenschaft 439
- border-top-right-radius-Eigenschaft 439
- border-top-style-Eigenschaft 71
- border-top-width-Eigenschaft 71
- border-width-Eigenschaft 71, 168
 - Werte 72
- bottom-Eigenschaft 89, 90, 92, 95, 177, 262, 407
- Bowman, Douglas 56
- Boxmodell 38–40, 398, 405
 - anpassen 489–491
 - anpassen für Grundlinienraster 470
 - Außenabstand (margin) 39
 - Border-Box 489
 - Brain Jar 40
 - Content-Box 489
 - Inhaltsbereich 39, 405
 - Innenabstand (padding) 39

- interaktive CSS-Boxmodell-Website 40
- Rahmen (border) 39
- box-shadow-Eigenschaft 229, 247, 298, 330, 370, 442–445
 - Ersatz in IE und 8 445–448
 - für Texteingabefelder 266
 - inset 248
 - inset-Schlüsselwort 443
 - Werte 442
- box-sizing-Eigenschaft 489–491
- Braille-Zeile 181, 279
- Brain Jar-Boxmodell 40
- Breadcrumb-Navigation 232–235
- Breite, minimal, angeben mit min-width-Eigenschaft 228
- Brian Fait 411
- Brotkrümel-Navigation *siehe* Breadcrumb-Navigation
- Browser 323
 - ältere Versionen finden 477
 - Android 381
 - Entwicklerwerkzeuge 481
 - Bugs erkennen und beheben 481–482
 - Caching 183, 189
 - Chrome
 - Erweiterungen für CSS-Entwicklung 480
 - Hintergrundbilder für Tabellenspalten 323
 - Probleme mit Floats in mehrspaltigem Text 417
 - proprietäre Selektoren 347
 - Darstellung angleichen 510
 - Elemente verstecken vor 498–499
 - Entwicklerwerkzeuge, eigene 479
 - Erweiterungen zum Testen 479–481
 - Firefox
 - abgerundete Ecken 438
 - border-radius, eigene Prefix-Schreibweise 249
 - Darstellungsmodell für Tabellenrahmen (separat) 303
 - Darstellungsprobleme bei legend-Elementen (bis Version 5) 292
 - Erweiterungen für CSS-Entwicklung 480
 - Firebug 481
 - Hintergrundbilder für Tabellenspalten 323
 - legend-Element (white-space) 256, 258
 - Web Developer 481
 - Gecko (Mobilgeräte) 358
 - Gecko, placeholder -Attribut gestalten 269
 - Grading 478
 - IEMobile 372
 - Internet Explorer
 - Buttons und font-size-Eigenschaft 276
 - Chrome Frame-Plugin 515
 - Conditional Comments 501–504
 - DebugBar 481
 - Erweiterungen für CSS-Entwicklung 480
 - IETester 477
 - Media Queries 359
 - proprietäre Eigenschaften
 - behaviour 440
 - filter 437, 445
 - Quirks-Modus, box-sizing-Eigenschaft und 491
 - Scrollbalken in textarea-Elementen 269
 - Stile ausschließlich definieren für 501–504
 - Unterstützung für Selektoren 26
 - Vererbung 263
 - Internet Explorer (bis 9), fehlende Unterstützung für CSS3-Farbverläufe 430
 - Internet Explorer 10, Unterstützung für mehrspaltigen Text mit CSS3 417
 - Internet Explorer 7
 - border-spacing, Workaround 305
 - Checkboxes 287
 - Darstellungsprobleme bei Positionierung in Navigationsmenüs 247
 - Darstellungsrollen und 508
 - fehlende Unterstützung für
 - caption-side 315
 - Daten-URLs 519
 - erzeugte Inhalte 488
 - hasLayout 506–509
 - legend-Element 260
 - Standardsmodus
 - Unterstützung für :hover-Pseudo-klasse 423
 - Internet Explorer 7 und 8
 - box-shadow
 - Ersatz für 445–448
 - fehlende Unterstützung 443
 - Fieldset Background Color Bleed Bug 258
 - filter
 - Drop Shadow (unzureichend) 446
 - Shadow (unzureichend) 446
 - legend-Element und 258, 263
 - transform-Eigenschaft simulieren 452
 - Transparenz 437

- Unterstützung für :focus-Pseudoklasse 283
- Unterstützung für CSS3-Selektoren mit selectivizr (Polyfill) 331
- Unterstützung für HTML5-Elemente herstellen 504–505
- Internet Explorer 8, Probleme mit Sticky Footer 411
- iPhone-Entwicklerwerkzeuge 481
- Lynx (textbasiert) 483
- mobile, Erweiterungen und SDKs für CSS-Entwicklung 480
- Opera
 - Darstellungsprobleme mit Positionierung in Navigationsmenüs 247
 - Erweiterungen für CSS-Entwicklung 480
 - Mobile 358
- Opera 11, Unterstützung für HTML5-input-Elemente 267
- Präfix-Schreibweise für CSS3-Eigenschaften und -werte 516–517
- Safari 381
 - abgerundete Ecken 438
 - Erweiterungen für CSS-Entwicklung 480
 - Probleme mit Floats in mehrspaltigem Text 417
 - proprietäre Selektoren 347
- Safari 5+, automatisch ausgefüllte Texteingabefelder, Hintergrundfarbe 267
- Safari 5.1, Zählung von Elementen für :nth-child() 311
- Standardstile zurücksetzen 509–511
- testen 477–479
- Text, zum Testen von Websites 483
- Unterstützung
 - für CSS3-Eigenschaften 417
 - für Farbverläufe 433
 - für page-break-before und page-break-after 341
 - für transition-Eigenschaft 460
 - Zielgruppe definieren 479
- verschiedene Versionen testen 477
- visuelle
 - Elemente verstecken vor 497–499
- Webkit
 - iPhone 358
 - placeholder-Attribut gestalten 269
 - Präfix-Schreibweise, doppelt für Farbverläufe 430

- Probleme mit Floats in mehrspaltigem Text 417
- proprietäre Selektoren 347
- Webkit-basiert, alt-Text und 178
- Browsercam 477
- Browsershots 477
- Browser-Stylesheet 46
- browserübergreifende Einrückungen für Listen 191
- Buchstaben, Stile für 30
- Buchstabenabstand 123–124
- Bugs in Internet Explorer
 - hover-ghost 240
- button-Element 274–277
 - grafisch 277–279
 - Text 279–281
- Buttons
 - für Formulare 274–275
 - grafische 279
 - Links darstellen als, in Mobilgeräten 379–382
 - Systemzeichensätze 126

C

- caption-Element 375
 - Stile für 314–315
- caption-Schlüsselwort 126
- caption-side-Eigenschaft 315, 329
- Caret-Zeichen (^), in Attribut-Selektor 339
- Cascading Cyle Sheets *siehe* CSS
- Cederholm, Dan 143
- Chao, Ingo 508
- @charset-Regel (Zeichenkodierung) 483–486
- Checkboxen 270–272
- checked-Attribut für ausgewählte
 - Checkboxen 284
- :checked-Pseudoklasse 34
- cicada principle *siehe* Zikaden-Prinzip
- cite-Element (HTML 5) 163
- class- und id-Attribute, CSS3-Pseudoklassen und -elemente als Alternative 32
- class-Attribut 173, 188, 216, 218, 264, 273, 309, 319, 321, 325, 370
 - HTML-Spezifikation 30
 - Markierung im HTML-Code 27
 - mehrere Klassen angeben 30
 - Selektor 9, 26–29, 264, 273
 - Unterschied zu id-Attribut 29
 - verständliche Benennung 29

- clear-Eigenschaft 101, 285, 291, 346, 371, 392, 409, 494
 - für zweispaltige Layouts 386
 - Inhalt von Fußzeilen unterhalb der Spaltenunterseite platzieren 387
- Clearfix-Methode 491–494
- ClearType 157
- clip-Eigenschaft
 - Elemente verstecken mit 497
- col-Element
 - eingeschränkte Gestaltung mit CSS 323
- colgroup-Element
 - eingeschränkte Gestaltung mit CSS 323
- Collapsing Margins *siehe* Außenabstände
- Color Picker 424–426
- color-Eigenschaft 116–117, 197, 294, 347
 - Auswirkungen auf Tabellenrahmen 303
 - Kontrast 427
 - nicht vererbt in IE7 und 8 297
 - Risiken und Nebenwirkungen 117
 - Vererbung 228
 - Werte 61
- ColorScheme Designer 424
- column-count-Eigenschaft 415
- column-gap-Eigenschaft 415
- column-rule-Eigenschaft 415
- column-width, fehlende Unterstützung 417
- Conditional Classes, Erweiterung für
 - Conditional Comments 503
- Conditional Comments 137, 158, 248, 296, 349, 411, 437, 445, 501–504
 - Missbrauch 448
- Content Management System (CMS) 415, 504
- content-Eigenschaft 32, 152, 181, 194, 195, 207, 338, 353, 377, 486–489, 494
 - Anführungszeichen 144
 - für Tooltips 220
 - Inhalte einfügen 32
 - Zähler einfügen 487
- counter-increment-Eigenschaft 488
- counter-reset-Eigenschaft 488
- crosshair (Wert der Eigenschaft cursor) 462
- CSS
 - @charset-Direktive 196
 - Anleitung
 - für Designer (Website) 522
 - für Selektoren 522
 - Code direkt im Browser untersuchen 480
 - Code testweise ändern 480
 - Dateien
 - einzel vom Webserver anfordern 5
 - extern, einbinden 4
 - parallel laden 5
 - Pfadangaben in 5
 - Deklarationen 7
 - Eigenschaften 6, 7
 - erweitern, mit Präprozessoren 511–513
 - Fehler vermeiden 476
 - in head-Teil einbetten 5
 - Klassen, dynamisch, in Content Management Systemen 504
 - Präprozessoren 511–513
 - Farbschema und 426
 - keine eigenständigen Programme 512
 - Less CSS 511–513
 - Sass 511–513
 - Probleme erkennen und beheben 481–482
 - Regel
 - allgemeine Form 6
 - Aufbau 6–8
 - Bestandteile 6
 - Deklarationsblock 6
 - für mehrerer Elemente definieren 13
 - für Nachfahren-Elemente 16
 - Selektor 6
 - wiederverwenden 5
 - Regeln verschachteln mit CSS-Präprozessoren 512
 - richtig strukturieren 55
 - Selektoren 6
 - testen mit Mobilgeräten 383
 - Website zur Browserunterstützung 481
 - Werte 8
 - zur Darstellung mathematischer Formeln 160
- CSS 2.1
 - animierbare Eigenschaften 457
 - Pseudoelemente 32
 - Pseudoklassen 31
 - Spezifikation 525
- CSS Template Layout, CSS3-Modul 418
- CSS Zen Garden 522
- .css, Dateierweiterung 2
- CSS3
 - abgerundete Ecken 438–442
 - Animationen 454–461
 - Animations-Modul 461

- animierbare Eigenschaften 457
- Attributselektor 217
 - für Substrings 339
- background-Eigenschaft und Präfix-Schreibweise 86
- background-size-Eigenschaft 82–85, 400, 402, 473
 - Fallback-Lösung 84
 - Werte 85
- benannte Farben 62
- border-radius-Eigenschaft 234, 247, 298, 370, 438–442
 - Einzeleigenschaften 249
- box-shadow-Eigenschaft 229, 247, 298, 330, 370, 442–445
 - für Texteingabefelder 266
 - inset 248
- column-count-Eigenschaft 415
- column-gap-Eigenschaft 415
- column-rule-Eigenschaft 415
- column-width-Eigenschaft
 - fehlende Unterstützung 417
- content-Eigenschaft 180, 181
- Eigenschaften für mehrspaltigen Text 417
- Eigenschaften, als Alternative zu eingebundenen Grafiken 369
- Fähigkeiten mit Polyfills nachrüsten 513–515
- Farbverläufe
 - anstelle von Bildern verwenden 430
 - definieren 429–434
 - Funktionen für 431
- Farbwerte 62
- Flexbox-Modul 418
- Grid Layout-Modul 419
- Grid Positioning-Modul 419
- lineare Farbverläufe 431–432
- linear-gradient()-Funktion 378, 400, 431–432
- Media Queries 356
- mehrere Hintergrundbilder
 - Browserunterstützung 81
 - definieren 79, 400
 - Fallback-Lösung 81
 - Reihenfolge 402
- mehrspaltige Layouts 415–418
- Modul für CSS-Variablen 513
- nachrüsten mit CSS3 PIE (Polyfill) 513
- Navigationslisten aufwerten mit 229
- neue Attribut-Selektoren 25
- neue Module für Seitenlayout 418–420
- Präfix-Schreibweise 82, 155, 298, 382, 415, 430, 516–517
- Pseudoelemente, Kennzeichnung 34
- Pseudoklassen 32
 - für Formulare 284
- Pseudoklassen und -elemente, Unterstützung ab IE9 34
- radiale Farbverläufe 432–433
- radial-gradient()-Funktion 432–433
- rgba()-Farben 370
- rgba()-Funktion 87
- Schattenwürfe 442–445
- Selectors-Modul 23
- Selektoren 22
 - für teilweise Attributwerte 500
 - nachrüsten mit selectivizr 514
- Template-Modul 418
- text-shadow-Eigenschaft 155–158, 248, 330, 370
- transform-Eigenschaft 370, 448–454
- transform-origin-Eigenschaft 452
- transition-delay-Eigenschaft 455
- transition-duration-Eigenschaft 455
- transition-Eigenschaft 230, 247, 361, 454–461
- transition-property-Eigenschaft 455
- transition-timing-function-Eigenschaft 455
- Transparenz 437
- Unterstützung in Mobilgeräten 369
- Werte
 - Präfix-Schreibweise 516–517
 - zum Beschleunigen von mobilen Websites 368–370
- CSS3 PIE (Polyfill) 513
- CSS3, Pseudoelemente 32
- CSS3-Eigenschaften validieren 477
- css-discuss, Mailingliste 523
- CSS-Ressourcen 521
 - Design 522
- cursive (Wert der Eigenschaft font-family) 110, 112
- Cursor
 - alternative Symbole definieren 461–462
- cursor-Eigenschaft 280, 461–462

D

Darstellung

- Anführungszeichen, browserübergreifend 145
- linear, Zurücksetzen von Eigenschaften für 372

- Trennung von Inhalt 5
- von HTML-Elementen 6

- Darstellungsmerkmal, Definition durch Eigenschaft 6

- Darstellungsprobleme
- im Internet Explorer
- hasLayout und 506–509

- Darstellungsrolle 40–44, 106, 224, 228
- ändern 220

- für Links 213
- für Listeneinträge 244
- für Mobilgeräte 377, 379
- für span-Elemente 297

- Block 41, 149, 285, 308, 376, 377
- Formulare 253
- für Inline-Elemente 263
- für span-Elemente 256, 263
- für Tabellenzellen 376
- für textarea-Elemente 269

- display-Eigenschaft 43
- ersetzte Elemente 42
- für label-Elemente 265

- Inline 41
- anonym 42
- für Links 235
- in Internet Explorer 7 508

- Inline-Block 42, 176, 264

- inline-block
- für Links 233
- Internet Explorer 7 und 177, 508

- table-cell 377
- table-header-group 343
- table-row 377
- table-row-group 343

Daten

- im Stylesheet speichern 518

- Datenmenge reduzieren 368

- Datentabellen, in Mobilgeräten darstellen 377

- Daten-URIs 517–520

- fehlende Unterstützung im Internet Explorer 7 519

- korrekte Kodierung 519
- sinnvolle Dateigröße 518

- dd-Element 207–209

- DebugBar 481

- default (Wert der Eigenschaft cursor) 462

- Definitionslisten, browserübergreifend darstellen 209

- Definitionslisten, Stile für 207–209

- Deklarationen, CSS 6, 7

- Deklarationsblock 6

- Deklarationsblöcke, beliebig viele 7

- Design 421–473

- auf mehreren Plattformen testen 477–479
- harmonische Skalen kombinieren 464
- modulare Skalen 463–465
- Ressourcen 522

- Dienst 527

- direction-Eigenschaft 160–161

- direkte Nachbarn *siehe* Selektoren, benachbarte Geschwisterelemente

- direkte Nachfahren *siehe* Kindelemente

- Direktiven

- für meta-viewport-Element 361
- viewport-meta-Element kombinieren 362

- disabled-Attribut für nicht benutzbare Formular-elemente 284

- :disabled-Pseudoklasse 34

- display-Eigenschaft 149, 176, 177, 183, 223, 263, 285, 308, 343, 345, 379, 381
- Elemente in Mobilgeräten ausblenden 370

- Flyout-Menüs und 239

- für vertikale Zentrierung 106

- Spezifikation 308

- Textersetzung und 180

- div-Elemente 215, 384

- als Container für Floats, ohne Höhe 492

- als Spalten darstellen 392

- per Conditional Comments einfügen 446

- dl-Element 207–209

- DOCTYPE-Deklaration 52–53

- korrekt 40

- Validatorprogramme und 53

- Document Object Model (DOM) 10, 326

- Dokumentenbaum, Bedeutung für Selektoren 12

- Dokumentenfluss 89, 91, 93, 351

- Elemente vor Browsern verstecken 497

- Floats 493

- Positionierung und 241

- Dokumententeile

- unbenannt, Selektor 30–34

- verbergen 222–224

- DOM *siehe* Document Object Model (DOM)

- Doppelkreuz (#), für id-Selektoren 29
- Doppelpunkt (:)
 - doppelt, zur Kennzeichnung von Pseudo-
elementen in CSS3 34, 488
 - für Pseudoelemente 488
 - zum Markieren von Pseudoklassen und
-elementen 31
 - zur Trennung von Eigenschaft und Wert 6
- dreispaltiges Layout 403–406
- Dropdown-Navigation 243–250
- Druck 333–354
 - Längeneinheiten 336
 - Stylesheet einbinden, per @import-
Regel 334, 335
 - Stylesheet unterteilen, für Druck-
ausgabe 334, 335
- Druckausgabe von URIs 338
- Druckversion
 - DIN A4 347
 - Elemente verbergen 350–351
 - Empfängeradresse in Formulare
einbinden 348
 - Farbschema für Schwarz-Weiß-Ausdruck
anpassen 337–338
 - für Formulare 344–349
 - Hintergrundbilder mitdrucken 352
 - Internet Explorer
 - Darstellung anpassen 349
 - Schriftgrößen
 - Standardwerte 336
 - Seitenumbrüche anlegen 341–342
 - Tabellenspalten auf jeder Seite
anzeigen 343–344
 - Textzeilen am Seitenanfang 342
 - Textzeilen am Seitenende 342
 - Verteilung auf mehrere Seiten 336
 - Waisen 342
 - Witwen 342
- dt-Element 207–209
- durchscheinende Hintergründe 87–89
- Durchschuss (Typografie) 123
- dynamische Pseudoklassen 280
 - Spezifikation 214

E

- Ecken, abgerundet 298
- eigene Stile für Tabellenüberschriften
definieren 313–314

- Eigenschaften 6
 - background 86–87, 177, 216–218, 329
 - background, Deklaration wiederholen 254
 - background, doppelt 233
 - background, dreifache Schreibweise 290
 - background, Internet Explorer 7 und 221
 - background, mehrfache Verwendung 87
 - background, redundante Schreibweise 295
 - background-attachment 78–79
 - background-color 73, 289
 - transparent-Schlüsselwort 436
 - Werte 61
 - background-image 73–74, 183, 289, 400
 - Farbverläufe definieren 429
 - für Aufzählungszeichen in Listen 200
 - grafische Unterstreichungen und 140–142
 - mehrere Bilder angeben 79–82
 - Werte 65, 73
 - background-position 76–78, 400, 402
 - Sprites 187
 - Werte 77
 - background-repeat 75–76, 149, 200, 289,
400
 - Werte 75
 - background-size 82–85, 400, 402, 473
 - Wechselwirkung mit background-
repeat 85
 - Werte 85
 - border 39, 70–73, 165–166, 167–168, 329
 - an Grundlinienraster anpassen 466
 - Auswirkungen von box-sizing 490
 - Druckversion 348
 - für Tabellenzellen 301–305
 - für Texteingabefelder 266, 293
 - in Formularen durch outline-Eigenschaft
ersetzen 495
 - Schlüsselwörter 72
 - zur Gestaltung von Karteireiter-
Navigationen 235
 - border-bottom 203, 346, 348
 - für Unterstreichungen 140, 212
 - border-bottom-color 71
 - border-bottom-left-radius 439
 - border-bottom-right-radius 439
 - border-bottom-style 71
 - border-bottom-width 71
 - border-collapse 303, 312, 329
 - border-color 71, 303
 - Werte 72

- border-left-color 71
- border-left-style 71
- border-left-width 71
- border-radius 247, 298, 370, 438–442
 - box-shadow und 443
 - Einzeleigenschaften 249
 - mit inherit-Wert 441
 - Vier-Werte-Schreibweise 234
 - Zwei-Werte-Schreibweise 440
- border-right-color 71
- border-right-style 71
- border-right-width 71
- border-spacing 304, 305–306
- border-style 71, 168, 304
 - Werte 72
- border-top 173
 - für Überstreichungen 140, 212
 - weglassen, in Listen 242
- border-top-color 71
- border-top-left-radius 439
- border-top-right-radius 439
- border-top-style 71
- border-top-width 71
- border-width 71, 168
 - Werte 72
- bottom 89, 90, 92, 95, 177, 262, 407
- box-shadow 229, 247, 298, 330, 370, 442–445
 - Ersatz in IE 7 und 8 445–448
 - für Texteingabefelder 266
 - inset-Schlüsselwort 248, 443
 - Werte 442
- box-sizing 40, 489–491
- Browserunterstützung ermitteln 481
- caption-side 315, 329
- clear 101, 285, 291, 346, 371, 392, 409, 494
- clear, für zweispaltige Layouts 386
- clip, Elemente verstecken mit 497
- color 116–117, 197, 294, 347
 - Auswirkungen auf Tabellenrahmen 303
 - Kontrast 427
 - nicht vererbt in IE7 und 8 297
 - Risiken und Nebenwirkungen 117
 - Vererbung 228
 - Werte 61
- column-count 415
- column-gap 415
- column-rule 415
- column-width, fehlende Unterstützung 417
- content 32, 152, 181, 194, 195, 207, 338, 353, 377, 486–489, 494
 - Anführungszeichen 144
 - für Tooltips 220
 - Zähler einfügen 487
- counter-increment 488
- counter-reset 488
- CSS 2.1
 - animierbar 457
 - Browserunterstützung ermitteln mit Quirksmode-Website 481
- CSS3
 - animierbar 457
 - content 180, 181
 - nachrüsten mit CSS3 PIE (Polyfill) 513
 - Präfix-Schreibweise 369, 401
- cursor 280, 461–462
- direction 160–161
- display 43, 149, 176, 177, 183, 223, 263, 285, 308, 343, 345, 379, 381
 - Elemente in Mobilgeräten ausblenden 370
 - für vertikale Zentrierung 106
 - Textersetzung und 180
 - Werte 43
 - zum Verstecken von Untermenüs in Flyout-Navigation 239
- display: inherit 45
- display: none 43
- empty-cells 304, 312, 376
- filter (proprietary) 437
- first-line 150–151
- float 100–102, 173, 206, 220, 285, 346, 371, 389, 392, 403, 409
 - Clearfix 491–494
 - für Grid-Systeme 413
 - für zweispaltige Layouts 385
 - Listeneinträge 205, 245
 - Unterstützung in Mobilgeräten 381
 - zurücksetzen für Mobilgeräte 379
- font 125–127
 - Beliebigkeitsregel 125
 - Mindestanforderungen 126
 - Spezifikation 127
 - zum Angeben von Systemschriften 126
- font-family 110, 142, 280, 329
- Webfonts und 128
- Werte 110

- font-size 113–116, 163, 280, 294, 394
 - erlaubte Maßeinheiten 114
 - für input- und label-Elemente
 - anpassen 266
 - Grundlinienraster und 467
 - in button-Element 276
 - Schlüsselwörter 292
 - und line-height 123
- font-style 118–120, 142
- font-variant 120–121
- font-weight 117, 118, 142, 330
 - für einzelne Rahmenseiten 72
 - für Tabellen 331
- height 60, 178, 353, 407, 410
 - Auswirkungen von box-sizing 490
- Kurzschrift 7
- left 89, 90, 92, 95, 240, 346, 497
- letter-spacing 123–124
 - Spezifikation 124
- line-height 122–123, 152, 159, 160, 294, 465–470
 - als Skalierungsfaktor 123
 - in font-Kurzschrifteigenschaft 125
- list-style 195, 197, 200, 202–203
- list-style-image 199, 200
- list-style-position 202
 - Spezifikation 202
- list-style-type
 - Darstellung von Aufzählungszeichen verändern 192
 - Definition für gesamte Liste 193
 - Werte 192
- margin 39, 66–69, 147, 149, 173, 329, 371, 390, 410
 - Formulare und 289
 - für horizontales Zentrieren 103
 - Probleme mit legend-Element 262
 - Texteingabefelder 266
 - zum beliebigen Anordnen von Layout-Spalten 394
 - zurücksetzen 509
 - zurücksetzen für zweispaltige Layouts 386
- margin-bottom 67, 345, 407
- margin-left 67, 200, 403
 - bei zweispaltigen Layouts 388
 - für Layout-Spalten 397
 - für Listen 191
- margin-right 67, 403
- margin-top 67, 409
- max-width 172, 183
- min-height 410
- min-height, hasLayout und 508
- min-width 228
- min-width, hasLayout und 508
- mit Einzeleigenschaft überschreiben 72
- nur bestimmte Werte möglich 8
- opacity 423, 434–438
- orphans 342
- outline 293
 - als Ersatz für border-Eigenschaft in Formularen 495
 - bei Auswahl von Links über Tastatur 213
 - statt border verwenden 266
 - Texteingabefelder und 266
- outline-color, zum Hervorheben von Elementen mit Fokus 214
- overflow 98–99, 327, 498
 - Clearfix 491–494
 - Darstellungsprobleme von legend-Element in Internet Explorer 7 und 260
 - für Bilder mit abgerundeten Ecken 441
- overflow-x 98–99
- overflow-x, Zeilenumbrüche und 135, 136
- overflow-y 98–99
 - für textarea-Elemente in Internet Explorer 269
 - vertikale Zentrierung und 108
- padding 39, 69–70, 147, 166, 173, 203, 329, 371
 - Auswirkungen von box-sizing 490
 - Formulare und 289
 - für Tabellen 301
 - für Tabellenzellen 301–305
 - für Texteingabefelder 266
 - zurücksetzen 509
 - zurücksetzen für zweispaltige Layouts 386
- padding-bottom 70, 349, 409
- padding-left 70, 200, 403
- padding-left, für Listen 191
- padding-right 70, 403
- padding-top 70
- padding-top, grafische Unterstreichungen und 141
- page-break-after 341–342
- page-break-before 341–342
- page-break-inside 341–342
- position 84, 89–90, 152, 158, 178, 262, 346, 406–408, 497

- absolute 90–92
- fixed 95–96
- horizontale Zentrierung und 103
- relative 92–93
- Werte 90
- proprietär
 - behaviour 440, 504
 - expression (Internet Explorer) 305
 - filter 156, 445, 504
 - Matrix 452
 - PixelRadius 446
 - MakeShadow 446
 - ms-text-size-adjust 372
 - webkit-text-size-adjust 372
- proprietäre, vor Validierung auskommen-
tieren oder auslagern 477
- quotes 146
- right 89, 90, 92, 95
- Standardwert 8
- Tabelle des W3C 8
- table-layout 326–327
- text-align 104, 133–134, 290
 - für Formularelemente 293
 - Spezifikation 134
- text-decoration 138–140, 346
 - mehrere Werte 138
 - Nachfahren-Elemente und 139
- text-indent 150, 277
- text-indent, Textersetzung 178
- text-shadow 155–158, 248, 330, 370
- text-shadow, geprägte Schrift 156
- text-transform 121–122
- top 89, 90, 92, 95, 160, 240, 262
- transform 370, 448–454
 - animieren 458
 - Funktionen 451
 - rotate-Funktion 450
 - scale-Funktion 449
 - scaleX-Funktion 449
 - scaleY-Funktion 449
 - simulieren in Internet Explorer 7 und 8 452
 - skewX-Funktion 450
 - skewY-Funktion 450
 - translate-Funktion 449
 - und position: fixed 453
 - Ursprung angeben mit transform-
origin 452
- transform-origin 452
- transition 230, 247, 361, 370, 454–461
 - Browserunterstützung 460
- transition-delay 455
- transition-duration 455
- transition-property 455
- transition-timing-function 455
- vertical-align 106, 158, 160, 308
- Verwendungszweck und Werte 8
- visibility 494
- vollständige Liste 521
- Werte 6
- white-space 292
- white-space, für legend-Elemente in
Firefox 256, 258
- widows 342
- width 60, 178, 353, 371, 379, 389
 - Auswirkungen von box-sizing 490
 - dreispaltiges Layout, bei Spalten mit
flexibler Breite 391–394
 - horizontale Zentrierung und 103
 - zurücksetzen in Mobilgeräten 372
- word-spacing 123–124
 - Spezifikation 124
- word-wrap 134, 135
- word-wrap, fehlende Trennstriche 135
- z-index 84, 96–98, 153, 220, 353, 446
- zum Aktivieren von hasLayout in IE7 507
- zum Deaktivieren von hasLayout in IE7 507
- Zurücksetzen von Einzeleigenschaften auf
Standardwerte 88
- Einheiten
 - em 266
- Einzeleigenschaften, zusammen mit Kurz-
schrifteigenschaft 7
- einziges Element bestimmten Typs, Pseudoklasse
:only-of-type() 33
- einziges Kindelement, Selektor 33
- Einzug, browserübergreifend in Listen 191
- Elementboxen 40
- Elemente
 - a 211–250
 - abwechselnd gestalten 36
 - alle auswählen mit universellem Selektor 13
 - Auswahl einschränken 14
 - Auswahl nach Zustand 31
 - auswählen 15–16
 - auf einmal 13–14
 - mit Selektoren 6

- nach Attribut 23–26
- unabhängig von Elementtyp 26
- b 118
- benachbarte Geschwister 20–22
 - Selektor 309, 314, 349
- Beziehungen zueinander 11
- Block 40, 285, 308, 377
 - Bilder als 176
 - Breite 103
 - Definition von Höhe und Breite 60
 - in Inline-Elementen (nicht erlaubt) 263
 - Links und 224–226, 228
 - variable Breite 391
 - zentrieren 103–104, 133
- blockquote 162
- body
 - Besonderheiten für background-position 78
 - Höhe definieren 409
 - mit id-Attribut versehen 230–232
 - per Conditional Comments erzeugen, für Internet Explorer 503
- Boxen 40
- Breite automatisch anpassen, in Mobilgeräten 362
- button 274–277, 279
 - anstelle von input verwenden 274
 - Reset, eigene Stile 276
 - Submit, eigene Stile 276
 - Text 279–281
- caption 375
- caption, Stile für 314–315
- col 323
- colgroup 323
- CSS-Regeln definieren für 13
- dd 207–209
- div 26, 384
 - als Container für Floats, ohne Höhe 492
 - als Spalten darstellen 392
 - Einfügen per Conditional Comments 446
- dl 207–209
- drehen 448–454
- dt 207–209
- dynamisch hervorheben 495–497
- Eltern, Beschreibung 12
- ersetzt 42, 275
- ersetzt, Checkboxes 271
- ersetzt, Radiobuttons 271
- erste Zeile, Stile für 30
- fieldset 254–256, 290, 345
 - als Positionierungskontext für legend 262
 - als umgebendes Element für input- und label-Elemente 287
 - Außenabstand explizit definieren 291
 - Darstellung vereinheitlichen 254
 - nebeneinander anordnen 288–300
 - Spezifikation 256
- font (veraltet) 118
- form, als umgebendes Element für input- und label-Elemente 287
- form, Stile für 252–254
- Formularfelder, gruppieren mit fieldset-Element 290
- Geschwister 20
 - allgemein 22–23
 - Beschreibung 12
- Größe, in harmonisches Verhältnis zu Schriftgröße setzen 463
- HTML 6, 11
- html, Höhe definieren 409
- HTML5 372
 - cite 162, 163
- i 118
- img 178
- in Druckversion verbergen 350–351
- ineinander positionieren 93–95
- Inhalte am Anfang einfügen 32
- Inhalte am Ende einfügen 32
- Inhaltsbereich 405
- Inline 40, 235
 - Bilder 176
 - Eigenschaften für Rollover-Effekte 422
 - ersetzt 166
 - ersetzt, Definition von Höhe und Breite 60
 - Links 213
 - Listeneinträge darstellen als 244
 - Text zentrieren in 133
 - zentrieren 104
- Inline-Block 176
- input 285, 292
 - checkbox 270–272
 - Druckversion 345
 - image, Probleme mit Zugänglichkeit 279
 - radio 270–272
 - Spezifikation 267
 - text 265–267
 - Text, Schattenwürfe 266
 - Typen, neu in HTML5, Polyfills 267

- Kind 18, 94
 - Bedeutung 11
 - beliebiges, Selektor 33
 - Beschreibung 12
 - einziges, Selektor 33
 - letztes, Selektor 33
 - Pseudoklasse 32
 - Selektor 375
- label 263–265, 285
 - Darstellungsrolle ändern 265
 - Druckversion 345
- Layout
 - Beeinflussung durch Schattenwürfe 444
- legend 256–260, 291
 - Darstellungsprobleme in Internet Explorer 7 und 8 258
 - Platzierung 261–263
 - Spezifikation 260
- li 206
 - als Inline-Element darstellen 244
 - Außen- und Innenabstände anpassen 227
- link 334, 335, 349
- link, zum Einbinden von Webfonts 128
- meta
 - viewport, Direktiven 361
 - viewport, zur Steuerung von Größe und Skalierung des Ansichtsbereichs 361
- mit Fokus 32
- mit Rahmen versehen 70–73
- mit Schattenwürfen versehen 330
- Nachfahren 16–18
 - Beschreibung 12
 - CSS-Regeln für 16
 - Selektor 309
 - und Vorfahren 16
- nicht benötigt, in Mobilgeräten ausblenden 370–371
- nicht scrollen 408
- nicht-semantisch, Wartungsaufwand 198
- option 272–274
- option, abwechselnd gestalten 273
- positionieren 89–90, 92, 176
- q 145
- Schattenwurf erzeugen in 443
- select 272–274
- Selektoren für 9
- skalieren
 - mit transform-Eigenschaft 449
- span 26, 142, 198, 218
 - als Block-Element definieren 256
 - anstelle von Radiobuttons und Checkboxes gestalten 271
 - Darstellungsrolle ändern 297
 - für legend-Element 256
 - für Textersetzung 178
- strecken 448–454
- sub 158
- super 158
- table 374
- tbody 311, 330, 331
- td 311, 329
- textarea 268–270
- textarea, Darstellungsrolle ändern 269
- tfoot 331
- tfoot, Platzierung vor tbody in HTML-Code 332
- th 311, 313, 329
- th, eigene Stile für 313–314
- thead 311, 329, 331, 343
- thead, tfoot, tbody, Reihenfolge vor HTML5 311
- tr 317–320
 - für Mobilgeräte als Blockelemente darstellen 376
 - in Darstellung für Mobilgeräte einbeziehen 375
- ul 195
 - Außen- und Innenabstände anpassen 227
 - bei Bedarf einblenden 241
 - für Navigation 226–230
- umfließen 100–102
- umfließen verhindern 101
- umformen 448–454
- unterscheiden (Attribut-Selektor) 265
- verschachtelt, Multiplikation relativer Maßeinheiten 59
- verschieben 448–454
- verzerrern 448–454
- vor allen Browsern verstecken 498–499
- Vorfahren 16
 - Bedeutung 11
 - Beschreibung 12
 - und Nachfahren 16
- zentrieren, horizontal 103–105
- zentrieren, vertikal 106–108

- Element-Selektoren 9, 15–16, 207, 254, 313, 329
 - für Auswahllisten 272
 - für Formulare 252–254
 - kombinieren für Kindselektor 18
- em-Einheiten 58, 266
 - Schriftgröße festlegen in 58
- empty-cells-Eigenschaft 304, 312, 376
- :empty-Pseudoklasse 232, 376
- enabled-Attribut für benutzbare Formular-
elemente 284
- :enabled-Pseudoklasse 34
- Entwicklerwerkzeuge, für CSS 479
- e-resize (Wert der Eigenschaft cursor) 462
- Ersatzschrift 130
- ersetzte Elemente 275
- ersetzte Inline-Elemente 60, 166
- erste Zeile, Pseudoelement
 - :first-line-Pseudoelement 32
- erster Buchstabe, Pseudoelement 32
- erstes Kindelement
 - :first-child-Pseudoklasse 32
- erstes Kindelement bestimmten Typs, Pseudo-
klasse :first-of-type() 33
- erzeugte Inhalte 205, 338, 377, 486–489
 - fehlende Unterstützung in IE7 488
 - Funktionen
 - attr() 488
 - url() 488
 - nur für Darstellung verwenden 488
 - Sonderzeichen als UTF kodieren 488
- externes Stylesheet 2, 4

F

- Fahrner-Image-Replacement (FIR) 179, 181
- fantasy (Wert der Eigenschaft font-family) 112
- Farbe
 - Alphakanal
 - rgba() 370
 - Änderungen durch Browser in Druck-
version 337
 - auswählen 424–426
 - definieren für lineare Verläufe 431, 432
 - Definition 61–64
 - Hintergrund
 - als Fallback für Hintergrundbild 73
 - hsla() 434–438
 - hsla(), für Schattenwürfe 442

- Kontrast, Zugänglichkeit 427–428
- Palette 426
- rgb() 220
- rgb()-Schreibweise, für Farben 318
- rgba() 87, 220, 299, 318, 378, 434–438
 - für Schattenwürfe 442
 - Schreibweise 330
- Schema
 - analog 425
 - Dreiklang 425
 - komplementär 425
 - monochrom 425
 - Vierklang 425
- Schrift 116–117
- Schwarz-Weiß für Ausdrucke 337–338
- transparent 434–438
- Vordergrund 116–117, 197, 303
- Werte
 - benannt, Beispiel 61
 - benannt, CSS3 62
 - hexadezimal, Beispiel 61
 - hsla() oder rgba() 64
 - hsla(), Beispiel 61
 - rgb() und rgba() 62
 - rgb(), Beispiel 61
 - rgb(), Prozentwerte
 - Beispiel 61
 - rgba() 177
 - rgba(), Beispiel 61
 - Transparenz 62
 - zurücksetzen für Druckversion 337
- Farbnamen *siehe* Farbe, Werte, benannt
- Farbschema
 - analog 425
 - Dreiklang 425
 - erstellen 424–426
 - komplementär 425
 - monochrom 425
 - passend zum Thema der Website wählen 426
 - Vierklang 425
 - Zugänglichkeit 426
- Farbverläufe 378
 - als Ersatz für Hintergrundbilder von Layout-
Spalten 402
 - als Ersatz für Hintergrundfarben 400
 - anstelle von Bildern verwenden 430
 - Browserunterstützung 433
 - definieren mit CSS3 429–434
 - Farbstopps 432

- Funktionen, Liste 431
- gleiche Farbe für Start- und Endwert verwenden 400
- linear 431–432
 - Schlüsselwörter 431
 - Winkel angeben 431
- linear, Richtung angeben 431
- mit CSS3 erzeugen 370
- radial 432–433
 - elliptische Form 433
 - Kreisform 433
 - Mittelpunkt angeben 432
 - Schlüsselwörter 433
- sich wiederholend 433
- Spezifikation 430
- W3C-Schreibweise, Änderungen 430
- Farbwähler 424–426
- Farbwerte angeben 8
- Faux Columns 399–402
- Festbreitenschriften 110
- feste (oder absolute) Schriftgrößen 116
- feste Breite für Tabellen definieren 326–327
- feste Positionierung 95–96, 406–408
- feste Positionierung, und transform-Eigenschaft 453
- Fieldset Background Color Bleed Bug (Internet Explorer 7 und 8) 258
- fieldset-Elemente 254–256, 290, 345
 - als Positionierungskontext für legend 262
 - als umgebendes Element für input- und label-Elemente 287
 - Darstellung vereinheitlichen 254
 - nebeneinander anordnen 288–300
- figcaption-HTML5-Element 174
- figure-HTML5-Element 174
- filter-Eigenschaft (proprietär) 156, 437, 445
 - Matrix-Filter (Internet Explorer) 452
- FIR (Fahrer Image Replacement) 179, 181
- :first-child-Pseudoklasse 20, 203, 207, 209, 291, 309, 314, 318, 321, 330, 349, 376
 - für erstes Element in Navigationslisten 246
 - in Navigationsmenüs 242
- :first-letter-Pseudoelement 32, 146, 147, 148
 - Einschränkungen 147
- :first-line-Pseudoelement 31, 150–151
- :first-of-type()-Pseudoklasse 33
- first-of-type-Pseudoklasse 121
- Flags zum Finden von Regeln in Stylesheets 56
- Flexible Box Layout, CSS3-Modul 418
- Float-basierte Layouts 394–398
- float-Eigenschaft 100–102, 173, 206, 220, 285, 346, 371, 389, 392, 403, 409
 - Clearfix 491–494
 - für Grid-Systeme 413
 - für zweispaltige Layouts 385–387
 - Listeneinträge 205, 245
 - Unterstützung in Mobilgeräten 381
 - zurücksetzen für Mobilgeräte 379
- Floats 100–102, 149, 173, 371, 403, 468
 - für Layouts 409
 - Höhe wiedergeben 491–494
 - in mehrspaltigem Text, mit CSS3-Eigenschaften 417
 - input- und label-Elemente definieren als 287
 - Listeneinträge 205
 - mehrspaltige Layouts 392
 - mit ausdrücklicher Breite für Layout-Spalten 397
 - mit Bildern 100
 - rückgängig machen 291
 - Spaltenreihenfolge in Layouts 387
 - Spezifikation 102
 - und Listeneinträge 245
- Flyout-Navigation 239–243
- :focus-Pseudoklasse 212–214, 281–283, 293, 495–497
- :focus-Pseudoklasse, animieren 454
- Font Squirrel (Webfonts) 130
 - Alternativen 132
- Fontdeck (Webfonts) 132
- @font-face-Generator (Font Squirrel) 130
- @font-face-Regel 128–132
- @font-face-Regeln, Werte 65
- font-family-Eigenschaft 110, 142, 280, 329
 - Webfonts und 128
- font-Kurzschrifteigenschaft 125–127
 - Beliebigkeitsregel 125
 - Mindestanforderungen 126
 - zum Angeben von Systemschriften 126
- font-size-Eigenschaft 113–116, 163, 280, 294, 394
 - für input- und label-Elemente anpassen 266
 - Grundlinienraster und 467
 - Maßeinheiten 114
 - Schlüsselwörter 292
 - Spezifikation 116
 - und line-height 123
- Fontspring (Webfonts) 132
- font-style-Eigenschaft 118–120, 142
- font-Tag (veraltet) 118

- font-variant-Eigenschaft 120–121
- font-weight-Eigenschaft 117–118, 142, 330
- Formate
 - DIN A4 336, 347
 - Längs in Mobilgeräten 360
 - Quer in Mobilgeräten 360
- form-Elemente
 - als umgebendes Element für input- und label-Elemente 287
 - Stile für 252–254
- Formulare 251–300
 - benötigte Felder hervorheben 283–284
 - Buttons 274–277
 - ausblenden in Druckversion 345
 - grafische 277–279
 - Text 279–281
 - Checkboxes, Darstellungsprobleme im Internet Explorer 7 287
 - Druckversion erstellen 344–349
 - Druckversion, Empfängeradresse einblenden 348
 - Eingabefelder hervorheben 281–283
 - Eingabefelder, Druckversion 348
 - Felder untereinander darstellen in Druckversion 345
 - Felder, erforderlich 283
 - input-Element
 - Druckversion 345
 - label-Element
 - Druckversion 345
 - mehrspaltige 288–300
 - mehrzeilige 285–287
 - Standarddarstellung 251
- Formularelemente
 - Benutzung möglich, Selektor 34
 - Benutzung nicht möglich, Selektor 34
 - Beschriftungen (label) 263–265
 - gruppiert (fieldset) 254–256
 - gruppiert, Beschriftungen (legend) 256–260
 - Platzierung 261–263
 - markiert 34
 - Texteingabe (textarea) 265–267, 268–270
 - zentrieren 293
- FOOUT (flash of unstyled text) 132
- Fragment-Identifizierer 223, 496
- Funktionen
 - attr() 220, 339
 - für transform-Eigenschaft 451
 - linear-gradient() 370, 400, 431–432

- radial-gradient() 432–433
- repeating-linear-gradient() 433
- repeating-radial-gradient() 433
- rotate (transform) 450
- scale (transform) 449
- scaleX (transform) 449
- scaleY (transform) 449
- skewX (transform) 450
- skewY (transform) 450
- translate (transform) 449
- translateX 449
- translateY (transform) 449
- url(), für Daten-URIs 518

G

- Gebietsschema (Locale) 485
- geschweifte Klammern für Deklarationsblock 6
- Geschwisterelemente 20
 - allgemein, Selektor 22–23
 - benachbart, Selektor 255
 - Beschreibung 12
- Gilder/Levin-Textersetzung 178, 181
- Glazman, Daniel 516
- Glish.com 523
- Goldener Schnitt 463, 465
- Google Fonts API 128
 - Alternativen 132
 - Anleitung 131
- Graceful Degradation 198, 249
 - für ältere Browser 196
- Graded Browser Support 478
- Grafiken
 - in Mobilgeräten vermeiden 368–370
- grafische Initialen 148–149
- grafische Unterstreichungen 140–142
- Gral, heilig 403
- Grid Layout, CSS3-Modul 419
- Grid Positioning, CSS3-Modul 419
- Grid-Layouts 411–415
- Gridsysteme 414
 - 1140 CSS Grid 414
 - 960.gs 414
 - Blueprint CSS (Google) 414
 - Nachteile 415
- Grundfarben 426
- Grundlagen zur Positionierung von Elementen mit CSS 522
- Grundlinie 176, 346
- Grundlinie, Bilder in Tabellenzellen und 306

- Grundlinienraster 465–470
 - Bilder anpassen 468
- Gruppieren
 - von Attributselektoren 25
 - von mehreren Klassen in class-Attribut 30
 - von Selektoren 12–13, 232, 299, 318
- gruppierte Formularelemente 254–256
 - Beschriftungen (legend) 256–260
 - Platzierung 261–263
- gruppierte Selektoren, auf mehrere Zeilen verteilen 13
- Gyroskop, in Mobilgeräten 360

H

- Hänsel und Gretel 234
- harmonische Skalen
 - kombinieren 464
 - Online-Werkzeug 463
- hasLayout 448, 506–509
 - Konzept 177
- head, Einbetten von CSS-Code in 5
- height, Direktive, für viewport-meta-Element 365
- height-Eigenschaft 60, 178, 353, 407, 410
 - Auswirkungen von box-sizing 490
- heiliger Gral 403
- help (Wert der Eigenschaft cursor) 462
- hexadezimale Farbwerte 62
- hexadezimale Unicode-Werte, Backslash 340
- hexadezimale Werte 8
- hexadezimale Werte schützen in CSS 196
- Hierarchie der HTML-Elemente 10
- Hintergrund
 - Bilder, Farbe als Fallback 73
 - Definition im HTML-Code, Probleme 85
 - durchscheinend 87–89
 - durchsichtig 436
 - Farben, simulieren mit Farbverläufen 400
 - für Layout-Spalten 399–402
 - Muster
 - quasi zufällig 470–473
 - transparent 177
- Hintergrundbilder 73–74, 183
 - als erzeugte Inhalte 353
 - als Vordergrundbild umdeuten 153
 - für Layout-Spalten, Breite definieren 400
 - für Textersetzung 178
 - kacheln 75

- mehrere 79–82
 - Darstellungsreihenfolge 295
 - definieren für Layout-Spalten 400
 - mit ausdrucken 352–354
 - positionieren 76–78
- Primzahlen als Breite 470
- skalieren 82–85
- verankern 78–79
- wiederholen 75–76
 - horizontal 292
 - Standardverhalten 73
- zum Einfügen von Symbolen nach Links 216–218
- Hintergrundfarbe 61
 - als Fallback für Hintergrundbilder 73
 - Kontrast 427
- Hintergrundmuster
 - quasi zufällig
 - erste Wiederholung berechnen 471
- History Sniffing
 - Gegenmaßnahmen 214
 - Konsequenz für :visited-Pseudoklasse 214
- Höhe und Breite
 - Definition für Block-Elemente 60
 - Definition für ersetzte Inline-Elemente 60
- Höhenangaben, Probleme bei vertikaler Zentrierung 108
- horizontale Navigation 235, 238
- horizontales Zentrieren 103–105
 - mit absoluten Werten 104
- hover-ghost-Bug in Internet Explorer 7 243
- :hover-Pseudoklasse 31, 32, 168–170, 212–214, 220, 280, 315–317, 323, 330, 421–423
- :hover-Pseudoklasse in Navigationsmenüs verdoppeln 242
- :hover-Pseudoklasse, animieren 454
- href-Attribut, als Wert für content-Eigenschaft 338
- hsla()-Farbwerte 63
- HTML
 - Attribute
 - media 334
 - Baumstruktur 10
 - Code erweitern für Gestaltung von Tabellenspalten 322
 - Code strukturieren für Grid-Systeme 413
 - Code validieren 477
 - CSS verwenden mit 1–5
 - Fragment-Identifizier 223

- Listen definieren 206
- Quellcode untersuchen mit Browser-
Erweiterungen 479
- Wurzelelement 11
- Zeichen-Entities 145, 340
- HTML 4.01
 - Spezifikation für DTD 53
- HTML5 53
 - Alternative zu Adobe Flash 371
 - cite-Element 162
 - Dokumenttyp 52–53
 - Elemente für Seitenlayouts 391
 - Formularattribute 284
 - placeholder-Attribut, in Druckversion
ausblenden 347
 - Reihenfolge von Tabellenelementen 311
 - Stile definieren für 504–505
- HTML-Code
 - nicht in style-Tags 5
- HTML-Dokument, Stylesheets definieren in 5
- HTML-Dokumente, CSS-Code direkt
einbetten 5
- html-Element 11
 - Höhe definieren 409
- HTML-Elemente
 - auswählen 9
 - benachbarte 20
 - Beziehung zueinander 12
 - Darstellung 6
 - Eigenarten, Selektoren für 12
 - Hierarchie 10
 - nach ihrer Abstammung auswählen 16
 - Selektoren für 15
- HTTP
 - Anfragen 183
 - gering halten durch Sprites 189
 - vermeiden durch CSS3 430
 - verringern durch Daten-URIs 517
 - Header
 - Content-type, charset-Attribut 196
 - jigsaw.w3.org, Online-Valdiator für
Stylesheets 475
 - Requests
 - beim Einbinden zusätzlicher CSS-
Dateien 502
 - für jede CSS-Datei 5, 367
 - überflüssig bei Stylesheets für Mobil-
geräte 366

I

- Icons
 - für Links 216–218
 - in gemeinsamer Datei speichern 187
 - Schrift des Betriebssystems für 127
 - speichern in Sprites 187–189
- id-Attribut 188, 215, 223, 227, 303, 309, 370
 - Benennung, verständlich 29
 - für body-Element 230–232
 - im HTML-Code verwenden 28
 - Selektor 26–29
 - verwenden, um Elemente in Druckversion zu
verbergen 350
 - Verwendung 5, 27
- id-Selektoren 9, 26–29, 188, 215, 227, 303,
309, 350, 370
 - Alternative 23
 - Benennung 27
 - kombinieren 231
 - Spezifikation 29
 - Verwendung 28
 - zur Auswahl der aktuellen Seite 231
 - zur Gestaltung verschiedener Formulare 252
- IE 7
 - background-Eigenschaft 221
 - Bugs
 - hover-ghost 243
 - erzeugte Inhalte und 196
 - Probleme mit :first-child-Pseudoklasse 204
 - rgb()-Schreibweise, nur mit background-
Kurzschrifteigenschaft 221
- IE Developer Toolbar 508
- ie6countdown.com 474
- IETester, Website 477, 479
- Image Replacement *siehe* Textersetzung
- img-Element 178
- Import von Stylesheets 4
- !important-Regel 51, 316, 500
- !important-Regel, für Druckversion 337, 338
- @import-Regel 4, 334, 335, 367
 - Werte 65
- @import-Regel, für Druck-Stylesheet 345
- individuelle Gestaltung
 - von Tabellenspalten 320–324
 - von Tabellenzeilen 317–320
 - von Tabellenzellen 308–311
- Inhalt
 - automatisch erzeugt 338
 - einfügen mit content-Eigenschaft 32

- erzeugen mit CSS 486–489
- strukturieren mit div-Elementen 384
- und Darstellung trennen 279
- zu groß für Element 98
- Inhaltsbereich 405
- inherit 280–281
- Initialen, hängend 146–148
- initial-scale, Direktive, für viewport-meta-Element 365
- Inline-Block-Elemente 176, 264
- Inline-Elemente 235, 244
 - Eigenschaften für Rollover-Effekte 422
 - ersetzte 166
 - Links 213
- Inline-Stile überschreiben 499–501
- Inline-Stile, im Browser anzeigen 486
- Inline-Stylesheets 4, 5
 - Nachteile 5
- Innenabstände 69–70
 - anpassen für Formulare 289
 - Boxmodell und 38
 - zurücksetzen 392, 509
 - zurücksetzen für Mobilgeräte 371
 - zurücksetzen, für zweispaltige Layouts 386
- input-Element 285, 292
 - checkbox 270–272
 - Druckversion 345
 - image, Probleme mit Zugänglichkeit 279
 - radio 270–272
 - Spezifikation 267
 - Text 265–267
- input-Elemente, neu in HTML5 267
- input-Elemente, unterscheiden 265
- Interaktivität, durch Hyperlinks 213
- interne Stylesheets 3–4, 5
 - Nachteile 5
- Internet Explorer (*siehe* auch Browser)
 - 7 bis 9, Textschatten 156
 - filter-Eigenschaft 156
 - hasLayout 177
 - hover-ghost-Bug 240
 - IE und 8
 - Selektoren 34
 - Unterstützung für CSS3 84
 - Unterstützung für mehrere Hintergrundbilder 81
 - Unterstützung für rgba() 63
- Intervalle
 - für :nth-child() und Verwandte 38

- Quarten, harmonische Skalen 465
- Quinten, harmonische Skalen 465
- iPad
 - Media Queries für 357
 - Stildefinitionen für 357
- iPhone
 - fehlende Unterstützung für Adobe Flash 371
- Irish, Paul 132, 503
- ISO 10646 *siehe* Unicode

J

- JavaScript 326
 - getComputedStyle, History Sniffing 214
 - HTML5-Elemente erzeugen mit, für ältere Versionen von IE 504–505
 - onmouseover-Event, für Rollover-Effekte 423
- Johansson, Roger 372
- Joomla (CMS) 415
- jQuery
 - Unterstützung für mehrspaltigen Text 417

K

- Karteireiter-Navigation 235, 238
- Kaskade 46–48
- Kaufmanns-Und (&), aus anderem Zeichensatz
 - einfügen 142
- Kindelemente 10, 18
 - Bedeutung 11
 - beliebige, Selektor 33
 - Beschreibung 12
 - letztes, Selektor 33
 - Positionierung und 94
 - Selektoren 18–20
- Kindselektor 375
 - in CSS 2.1-Spezifikation 20
 - Nachfahrenselektor, Unterschied 19
 - Spezifität 195
- Klammern, geschweifte für Deklarationsblock 6
- Klassen
 - für Grid-Systeme 413
 - Pseudo 30–34
 - Selektoren, Parallele zu Attributselektor 24
- Klassenselektoren 9, 26–29, 188, 309, 319, 321, 325, 350, 370
 - Alternative 23
 - auf bestimmte Elemente beschränken 27
 - mit Punkt kennzeichnen in CSS 27

- Spezifikation 29
- Verwendung 27
- Kodierung, UTF-8 483–486
- Komma
 - ungültig nach gruppierten Selektoren 13
 - zum Trennen von Selektoren 12, 13
- Kommentare
 - Spezifikation 56
- Komplémentärfarben 425
- Kontrastverhältnis
 - ausreichend für großen Text 427
 - ausreichend für normalen Text 427
 - berechnen, Werkzeuge 427
- kursive Schriften 210
- Kurzschrifteigenschaften 7
 - automatisches Zurücksetzen von nicht definierten Werten 8
 - background 86–87
 - Einzeleigenschaften auf Standardwerte zurücksetzen 88
 - für einzelne Rahmenseiten 72
 - mit Einzeleigenschaft überschreiben 72
 - padding 70

L

- label-Element 263–265, 285
 - Darstellungsrolle ändern 265
 - Druckversion 345
 - Spezifikation 287
- Laden von Stylesheets beschleunigen 367–368
- Laden von Stylesheets per @import 367
- Landschaftsmodus, in Mobilgeräten 360
- lang-Attribut, Pseudoklasse
 - :lang-Pseudoklasse 32
- Längeneinheiten 57–60
 - absolut 58
 - Millimeter (mm) 58
 - Pica (pc) 58
 - Punkt (pt) 58
 - Zentimeter (cm) 58
 - Zoll (in) 58
 - absolute und relative 58
 - für Druckversion 336
 - Pixel, Verwendung in Druckversion (Ausnahme) 345
 - relativ 58
 - em 58
 - Pixel (px) 58
 - x-Höhe (ex) 58

- rem 394
- larger-Schlüsselwort 115
- large-Schlüsselwort 115
- :last-child-Pseudoklasse 32, 33, 209, 249, 330, 331, 345
- :last-of-type()-Pseudoklasse 33
- Laufweite 124
- Layout
 - dreispaltig
 - Seitenleisten mit fester Breite, Hauptspalte flexibel 403–406
 - Faux Columns 399–402
 - feste Breite definieren 389
 - feste Breite, Hintergründe definieren 402
 - flexibel, Hintergründe definieren für 402
 - flexibel, Prozentwerte 170, 392
 - Float-basiert 409
 - Floats
 - mit beliebiger Spaltenreihenfolge 394–398
 - flüssig, Prozentwerte und 170
 - Footer an Unterkante des Ansichtsbereichs verankern 406
 - Footer, feste Höhe für »Sticky Footer« 410
 - Frameworks 414
 - Grid-basiert 411–415
 - Gridsysteme
 - 1140 CSS Grid 414
 - 960.gs 414
 - Blueprint (Google) 414
 - HTML strukturieren 413
 - Nachteile 415
 - Innen- und Außenabstände definieren 394
 - mehrspaltig
 - flexibel, mit Hilfe von Floats 391–394
 - Inhalte in Mobilgeräten untereinander darstellen 371
 - mit CSS3 415–418
 - neue CSS3-Module 418–420
 - Raster 414
 - Spalten
 - in beliebiger Reihenfolge 397
 - mit beliebiger Höhe 403
 - mit unterschiedlicher Höhe, Hintergründe 401
 - Verschiebung bei Floats verhindern 405
 - Sticky Footer 408–411
 - zweispaltig
 - flexibel 384–388
 - Spalten mit fester Breite 388–391

- leere Tabellenzellen verstecken 312
- Leerraum, Bilder in Tabellenzellen und 306
- Leerraum, unterhalb von Bildern entfernen 308
- Leerzeichen, nicht umbrechend 137
- Leerzeichen, zum Trennen von Vorfahren- und Nachfahren-Element 16
- left-Eigenschaft 89, 90, 92, 95, 240, 346, 497
 - horizontale Zentrierung und 103
- legend-Element 256–260, 291
 - Darstellungsprobleme in Internet Explorer 7 und 8 258
 - Platzierung 261–263
- letter-spacing-Eigenschaft 123–124
 - Spezifikation 124
- letztes Element bestimmten Typs, Pseudoklasse :first-of-type() 33
- letztes Kindelement
 - Pseudoklasse 33
- li-Element 206
 - als Inline-Element darstellen 244
 - Außen- und Innenabstände anpassen 227
- lineare Farbverläufe 431–432
 - Winkel angeben 431
- linear-gradient(), CSS3-Funktion 370
- linear-gradient()-Funktion 378, 400, 431–432
- Linearisierung
 - Anpassung für Mobilgeräte 372
 - Prozentwerte 373
 - relative Maßeinheiten 373
 - Tabellen 373
 - von Seitenlayouts für Mobilgeräte 371–373
 - von Tabellen für Mobilgeräte 373–378
- line-height-Eigenschaft 122–123, 152, 159, 160, 294, 465–470
 - als Skalierungsfaktor 123
- link-Element 349
- :link-Pseudoklasse 212–214
- Links 211–250
 - aktiv 32
 - als Buttons darstellen, in Mobilgeräten 379–382
 - anklickbaren Bereich vergrößern 224, 228
 - besuchte 32
 - Cursor, verändern 461
 - Farben ändern 215–216
 - grafische 168
 - Inline-Elemente 213
 - mit Icons versehen 216–218
 - nicht besuchte 32
 - Reihenfolge von dynamischen Pseudoklassen, LoVe/HaTe-Prinzip 215
 - relativ zum Wurzelverzeichnis 339
 - Rollover-Effekte 422
 - Sonderzeichen in Druckversion einfügen vor 340–341
 - Tooltips 218–222
 - Unterstreichung 212
 - URI in Druckversion mit anzeigen 338–339
 - verschachteln, Verbot 221
 - Ziel
 - hervorheben mit :target-Pseudoklasse 495
 - Selektor 33
 - Stile für 222–224
 - Zustände, Darstellung anpassen 212–214
- link-Tag, Vorteile 4
- Listen 190–210
 - Aufzählungszeichen
 - andersfarbig 197–198
 - anzeigen innerhalb von 201–202
 - definieren 488
 - eigene (grafisch) 199–201
 - eigene (textbasiert) 194–197
 - entfernen 206
 - entfernen, für Navigationsmenüs 226
 - verbergen 193
 - browserübergreifende Einrückung in 191
 - Definition im HTML-Code 206
 - Definition, browserübergreifend einheitlich darstellen 209
 - Definition, Stile 207–209
 - Einträge nebeneinander darstellen 205–207, 236
 - Einträge, Darstellungsprobleme 245
 - erzeugte Inhalte 205
 - horizontale Menüs 206
 - Navigation 226–230
 - in Mobilgeräten als Buttons darstellen 379
 - mit CSS3 aufwerten 229
 - Nummerierung 196
 - obere Rahmenseite nicht verdoppeln 242
 - Standardstile verändern 192–193
 - Stildefinitionen für alle Elemente 193
 - Stile für ersten und letzten Eintrag 209–210
 - Stile, per list-style definieren 202–203
 - Trennlinien zwischen Einträgen 203–205
 - ungeordnete, für Navigationsmenüs 228
 - Unterlisten verstecken 239
 - verschachtelte, für Navigationsmenüs 246
- Listeneinträge darstellen als 244

- Listeneinträge, Stile für ersten und letzten 209–210
- Listenmarker, verbergen 193
- list-style-image-Eigenschaft 199, 200
- list-style-Kurzschrifteigenschaft 195, 197, 200, 202–203
- list-style-position-Eigenschaft 202
 - Spezifikation 202
- list-style-type-Eigenschaft
 - Bilder als Aufzählungszeichen und 200
 - Darstellung von Aufzählungszeichen verändern 192
 - Definition für gesamte Liste 193
 - Werte 192
- LoVe/HaTe-Prinzip, für Reihenfolge von dynamischen Pseudoklassen für Links 213, 215
- Lynx Viewer 483

M

- Mailingliste zu CSS 523
- Málek, Vilém 243
- margin-bottom-Eigenschaft 67, 345, 407
- margin-Eigenschaft 39, 66–69, 149, 173, 329, 371, 390, 410
 - bei zweispaltigen Layouts 388
 - Formulare und 289
 - Probleme mit legend-Element 262
 - Texteingabefelder 266
 - zum beliebigen Anordnen von Layout-Spalten 394
 - zurücksetzen 509
 - zurücksetzen für zweispaltige Layouts 386
- margin-Kurzschrifteigenschaft 67, 103, 147
- margin-left-Eigenschaft 67, 200, 403
 - für Layout-Spalten 397
 - für Listen 191
- margin-right-Eigenschaft 67, 403
- margins
 - Boxmodell und 38
- margin-top-Eigenschaft 67, 409
- ::marker-Pseudoelement (CSS3) 198
- Markerstile für Listeneinträge 192, 193
- Markup
 - nicht-semantisches 263
- Maßeinheiten
 - absolut 116
 - absolut, für Druckversion 116, 345

- für Schriften 113–116
- für Schriftgröße 114
- Länge 57–60
- Prozent 59
- Prozentwerte
 - für Linearisierung 373
- relativ
 - für Linearisierung 373
 - für Mobilgeräte 377
 - Multiplikation 59
 - rem 59, 116
- mathematische Formeln, Darstellung mit CSS 160
- Mauszeiger, über Element
 - Pseudoklasse 32
- max-device-width (Media Queries) 357
- maximum-scale-Direktive, für viewport-meta-Element 365
- max-width (Media Queries) 357
- max-width-Eigenschaft 172, 183
 - Spezifikation 183
- Media Queries 355–359, 381
 - device-width 360
 - in älteren Versionen von IE mit Polyfill nachrüsten 359
 - kombinieren 358
 - max-width und max-device-width im Vergleich 357
 - only-Schlüsselwort 356
 - orientation 360
 - Polyfill, laden vor Verwendung der Medienabfragen 359
 - Überschreiben von allgemein Stildefinitionen 374
 - und Responsive Webdesign 359
 - webkit-min-device-pixel-ratio 358
 - width 360
 - zum Unterscheiden von Merkmalen 356
- @media-Regel 334, 335, 372
 - für Druckversion 353
- Medienabfragen *siehe* Media Queries
- Medientyp 53–55
 - all 55
 - ausdrücklich angeben 335
 - für Druckausgabe 334, 335
 - handheld, veraltet 356
 - print 55, 333, 334, 335
 - print, im Vergleich mit screen 335
 - projection 55

- screen 55, 349
- screen, in Media Queries 356
- Stildefinitionen für verschiedene in gemeinsamem Stylesheet 54
- medium-Schlüsselwort 115
- mehrspaltige Formulare 288–300
- mehrspaltige Layouts
 - flexibel mit Hilfe von Floats 391–394
 - mit CSS3 415–418
- mehrzeilige Formulare 285–287
- Meiert, Jens O. 149, 504
- Merrill, Randy 472
- Meyer, Eric A. 116, 120, 193, 523
- meyerweb-reset, Reset-Stylesheet 509
- Mikroformate 526
 - Adressen 349
- Millimeter 58
- min-height
 - hasLayout und 508
- min-height-Eigenschaft 410
- minimum-scale, Direktive, für viewport-meta-Element 365
- min-width-Eigenschaft 228
 - hasLayout und 508
- Mobile First 366, 373
- Mobilgeräte 355–383
 - anhand von Merkmalen unterscheiden mit Media Queries 356
- Ansichtsbereich
 - Größe und Skalierung steuern 361–365
- Auflösung
 - ermitteln mit Media Queries 356
 - intern höher als angegeben 362
- Ausrichtung ermitteln 356
- Bildschirmgröße 356
- CSS testen in 383
- CSS3-Unterstützung 369
- Darstellung zu klein 361
- Darstellungsrollen anpassen für 379
- Elemente ausblenden in 370–371
- keine Unterstützung für :hover-Pseudoklasse 423
- Laden von Seiten mit CSS3
 - beschleunigen 368–370
- Längs- oder Querformat ermitteln 360
- Layoutbreite an Breite von Ansichtsbereich anpassen 362
- Layouthöhe an Höhe von Ansichtsbereich anpassen 364

- Links als Buttons darstellen 379–382
- mit hoher Auflösung
 - Stylesheets für 358
- Positionierung zurücksetzen 371
- relative Maßeinheiten 377
- SDK zum Testen 383
- Seitenlayouts linearisieren 371–373
- Skalierung steuern 364
- Skalierungsfaktor angeben 364
- Stile zuerst laden 366
- Stylesheets validieren für 383
- Stylesheets, Laden beschleunigen 367–368
- Tabellen darstellen 377
- Tabellen linearisieren 373–378
- Testen
 - Online-Simulator 383
 - SDK 383
- unterschiedliche Bedeutung von Pixeln 362
- viewport-meta-Element
 - Direktiven kombinieren 365
- Werbebanner ausblenden 371
- Zoom 362
 - ausdrücklich erlauben 364
 - selbst steuern 362
- Modell der separaten Rahmen (für Tabellenzellen) 304
- Modell der zusammengefassten Rahmen (für Tabellenzellen) 303
- modulare Skalen (Design) 463–465
- modularscale.com 463
- monochromes Farbschema 425
- monospace 110, 112
- move (Wert der Eigenschaft cursor) 462
- ::-moz-placeholder-Pseudoklasse 269
- ms-text-size-adjust-Eigenschaft 372
- Multimedia-Inhalte
 - nicht mit ausdrucken 350
- Musik
 - Gemeinsamkeiten mit Webdesign 465, 469

N

- Nachbarn, direkte *siehe* Selektoren, benachbarte Geschwisterelemente
- Nachfahren, direkt *siehe* Kindelemente
- Nachfahrenselemente 16–18
 - Beschreibung 12
 - CSS-Regeln für 16
 - Selektor 309

- Nachfahrenselektoren 9, 17, 145, 170, 198
 - Spezifität 18
 - verwenden anstelle von Kindselektoren 246
- Namen von CSS-Eigenschaften 7
- Navigation 211–250
 - Breadcrumb 232–235
 - horizontale Menüs 206
 - Karteireiter 235, 238
 - Links
 - Rollover-Effekte
 - Transparenz 434
 - Listen 226–230
 - Tabs 235, 238
- Navigation, horizontal 235–238
- Navigationslisten, in Mobilgeräten als Buttons darstellen 379
- Navigationsmenüs 169
 - aktuelle Seite hervorheben 230–232
 - horizontal, mit mehreren Ebenen 243–250
 - Karteireiter, Probleme 238
 - vertikal 226–230
 - vertikal, mit mehreren Ebenen 239–243
- negative Außenabstände 163, 403, 409
 - für horizontale Zentrierung 104
- ne-resize (Wert der Eigenschaft cursor) 462
- Newhouse, Mark 230, 523
- nicht besuchte Links, Pseudoklasse
 - :link-Pseudoklasse 32
- nicht umbrechende Leerzeichen 137
- nicht-semantisches Markup 263
- Nielsen, Jakob 211
- normalize.css, Reset-Stylesheet 510
- :not()-Pseudoklasse 34
- :not-Selektor 222–224
- n-resize (Wert der Eigenschaft cursor) 462
- :nth-child()-Pseudoklasse 33, 273, 310, 318, 320, 324, 326, 329, 375
 - Intervall 38
 - Schlüsselwörter 36
 - Startwert 38
 - und :nth-of-type, Unterschiede 37
 - Werte 35–38
- :nth-child()-Selektor 377
- :nth-last-child()-Pseudoklasse 33
- :nth-last-of-type()-Pseudoklasse 33
- :nth-of-type()-Pseudoklasse 33
- Nummerierung
 - automatisch, mit content-Eigenschaft 487
- nw-resize (Wert der Eigenschaft cursor) 462

O

- odd/even-Schlüsselwörter für :nth-child() 36
- Off-Left-Technik 148, 178, 180, 277, 497
- Online-Validator, für CSS 475
- :only-child-Pseudoklasse 33
- :only-of-type()-Pseudoklasse 33
- opacity-Eigenschaft 423, 434–438
- Open Source, Schriften 130
- :optional-Attribut für optionale Formular-elemente 284
- option-Element 272–274
 - abwechselnd gestalten 273
- orphans-Eigenschaft 342
- outline-color-Eigenschaft
 - zum Hervorheben von Elementen mit Fokus 214
- outline-Eigenschaft 293
 - als Ersatz für border-Eigenschaft in Formularen 495
 - bei Auswahl von Links über Tastatur 213
 - statt border verwenden 266
 - Texteingabefelder 266
- overflow-Eigenschaft 98–99, 327, 498
 - Clearfix 491–494
 - für Bilder mit abgerundeten Ecken 441
- overflow-x-Eigenschaft 98–99
 - Zeilenumbrüche und 135, 136
- overflow-y-Eigenschaft 98–99
 - für textarea-Elemente in Internet Explorer 269
 - vertikale Zentrierung und 108

P

- padding-bottom-Eigenschaft 349, 386, 409
- Padding-Box 77
 - als Ursprung für top- und left-Eigenschaften 244
- padding-Eigenschaft 39, 69–70, 147, 166, 173, 203, 329, 371
 - Auswirkungen von box-sizing 490
 - Formulare und 289
 - für Tabellen 301
 - für Tabellenzellen 301–305
 - Texteingabefelder 266
 - zurücksetzen 509
 - zurücksetzen für zweispaltige Layouts 386
- padding-left-Eigenschaft 70, 200, 403
 - für Listen 191
- padding-right-Eigenschaft 70, 403

- padding-top-Eigenschaft 70
- grafische Unterstreichungen und 141
- page-break-after-Eigenschaft 341–342
- page-break-before-Eigenschaft 341–342
- page-break-inside-Eigenschaft 341–342
- Palette 426
- Panorama-Präsentation von Bildern
 - erstellen 182–183
- Pfadangaben in CSS-Dateien, relativ zu Datei 5
- Phark-Textersetzung 178, 180
- PHP 326, 512
- Pica (pc) 58
- Pipe-Zeichen *siehe* senkrechter Strich
- Pixel, Maßeinheit 58
- placeholder-Attribut
 - gestalten in Gecko-Browsern 269
 - gestalten in Webkit-Browsern 269
 - in Druckversion ausblenden 347
- Pluszeichen (+), Selektoren
 - benachbarte Geschwisterelemente 20
- PNG-Grafiken 177, 290
 - als Hintergrundbilder 87
 - für Transparenz 436
- pointer (Wert der Eigenschaft cursor) 462
- Polyfills 513–515
 - Chrome Frame 515
 - CSS3 PIE 266, 439, 513, 515
 - CSS3 PIE, für mehrfache Hintergründe 473
 - css3-mediaqueries-js 515
 - Definition 514
 - Flexie 515
 - für abgerundete Ecken 440
 - für Media Queries 359
 - html5.js 505
 - jepnope.js 515
 - Modernizr.js 515
 - Prefixfree 515
 - selectivizr 34, 210, 224, 331, 500, 514, 515
- Porträtmodus, in Mobilgeräten 360
- position, absolute 497
- position-Eigenschaft 84, 89–96, 152, 158, 178, 262, 346, 406–408
 - horizontale Zentrierung und 103
 - Werte 90
- Positionierung 89–90
 - absolut 90–92, 176, 220, 262, 497
 - absolut, für Listeneinträge in Dropdown-Navigationsmenüs 245
 - bottom 89, 90, 92, 95
 - fest 95–96, 406–408
 - fest, und transform-Eigenschaft 453
 - ineinander 93–95
 - Kontext 90–95, 176
 - left 89, 90, 92, 95
 - relativ 92–93, 158, 248, 262, 346
 - right 89, 90, 92, 95
 - top 89, 90, 92, 95
 - und Quellcode-Reihenfolge 89
 - z-index 96–98
 - zurücksetzen für Mobilgeräte 371
- Positionierungskontext 90–95, 152, 158, 176, 241, 261, 262, 446
 - bei Schattenwürfen in IE 445
 - in Navigationsmenüs 240
 - Textersetzung (Gilder/Levin) 178
- Präfix-Eigenschaften
 - für Animationen 458
 - Warnung vor ausschließlicher Verwendung von -webkit-Präfix 516
- Präfix-Schreibweise 247
 - doppelt, für Webkit-Browser 430
 - für border-radius 438
 - für CSS3-Eigenschaften 155, 298, 369, 382, 401, 415, 516–517
 - für Schattenwürfe 229
 - im Internet Explorer 8+ 158
 - Version ohne Präfix immer mit angeben 517
- Primzahlen 470–473
- print-Medientyp 55, 333–335
- Print-Stylesheets 333–354
- Programmiersprachen
 - clientseitig, JavaScript 326
 - PHP, für CSS-Präprozessoren 512
 - Ruby, für CSS-Präprozessoren 512
 - serverseitig, PHP 326
- progress (Wert der Eigenschaft cursor) 462
- Progressive Enhancement 247, 249, 273, 319, 329, 376, 377
 - abgerundete Ecken 439
 - CSS3-Eigenschaften und 230
 - für Formulare 288–300
 - Transparenz und 437
- projection-Medientyp 55
- Prozentwerte 59, 170–172
 - Bezug auf Breite des umgebenden Elements 172
 - für column-gap- und column-rule, nicht erlaubt 417
 - für Druckversion 336
 - für flexible Layouts 392

- für Linearisierung 373
- für vertikale Zentrierung 107
- horizontale Zentrierung und 103
- Pseudoelemente 30–34
 - ::marker 198
 - ::selection 153
 - ::slot 419
 - :after 32
 - :before 32
 - :first-letter 32
 - :first-line 31, 32
 - in CSS 2.1 32
 - in CSS3 32
- Pseudoklasse 30–34
 - :active 32
 - :checked 34
 - :disabled 34
 - :empty 232, 376
 - :enabled 34
 - :first-child 20, 32
 - :first-of-type 33
 - :focus 32
 - :hover 31, 32
 - :hover, keine Unterstützung in Mobilgeräten 423
 - :lang 32
 - :last-child 32, 33
 - :last-of-type 33
 - :link 32
 - :not() 34
 - :nth-child() 33, 35–38
 - Intervall 38
 - Intervall und Startpunkt 36
 - nummerierte Elemente 35
 - Schlüsselwörter 36
 - Startwert 38
 - und :nth-of-type(), Unterschiede 37
 - :nth-last-child() 33
 - :nth-last-of-type() 33
 - :nth-of-type() 33
 - :only-child 33
 - :only-of-type 33
 - :optional 284
 - :required 284
 - :root 33
 - :target 33
 - :visited 32
 - CSS 2.1 31

- dynamisch, Spezifikation 270
- in CSS3 32
- Pseudoklassen kombinieren 282
- Pseudoklassen, für Sprachattribute 161
- Punkt (.), zum Kennzeichnen von Klassen-Selektoren 27
- Punkt (pt) 58

Q

- q-Element 145
- quasi zufällige Hintergrundmuster 470–473
- Quellcode-Reihenfolge 372, 446
 - in mehrspaltigen Layouts 403
 - und Positionierung 89
- Quirksmode-Website 12
- Quirks-Modus für Browser 52
- Quirks-Modus, box-sizing-Eigenschaft und 491
- quotes-Eigenschaften 146

R

- radiale Farbverläufe 432–433
 - elliptische Form 433
 - Kreisform 433
 - Mittelpunkt angeben 432
- radial-gradient()-Funktion 432–433
- Radiobuttons 270–272
- Rahmen 70–73
 - Boxmodell und 38
 - Einzeleigenschaften für 71
 - Farbe ändern bei :hover 169
 - für Bilder 165–166
 - für Bilder entfernen 167–168
 - Stil, notwendig für Definition 168
- Rahmenbreite
 - Werte 72
- Rahmenfarbe
 - ändern 168–170
 - Werte 72
- Rahmenstil
 - Werte 72
- Rapid Prototyping 415
- Real World Style 523
- Regeln
 - !important 51
 - @font-face 128–132
 - Konflikte lösen 46–48
 - Spezifität 49, 167, 168

- relative Längeneinheiten 58
- relative Maßeinheiten, für Linearisierung 373
- relative Positionierung 92–93, 158, 248, 262, 346
 - Spezifikation 93
- rem-Einheiten 59, 394
- repeating-linear-gradient()-Funktion 433
- repeating-radial-gradient()-Funktion 433
- required-Attribut 283–284
 - für erforderliche Formularelemente 284
- :required-Pseudoklasse 283–284
- Reset-Buttons 274–277
 - eigene Stile 276
 - grafisch 277–279
 - Text 279–281
- Reset-Stylesheets 509–511
 - meyerweb-reset 509
 - normalize.css 510
 - universeller Selektor und 13
 - YUI 2
 - Reset CSS 510
- Responsive Webdesign
 - und Media Queries 356, 359
- Ressourcen, extern, einbinden 64–65
- Retina-Display 362
 - Media Queries für 358
- rgb() 318
- rgb()- und rgba()-Farbwerte 62
- rgba() 177, 299, 318, 330, 370, 378
- Rhythmus, vertikal 465–470
- right-Eigenschaft 89, 90, 92, 95
- Rollover-Effekte 168–170, 212–214, 275
 - für Tabellenspalten 323
 - für Tabellenzeilen 315–317, 330
 - ohne Bilder 421–423
- Root-Links 339
- :root-Pseudoklasse 33
- rotate-Funktion (transform) 450
- Ruby 512

S

- Safari 146
 - Hintergrundbilder für Tabellenspalten 323
- sans-serif (Wert der Eigenschaft
 - font-family) 110, 112
- scale-Funktion (transform) 449
- scaleX-Funktion (transform) 449
- scaleY-Funktion (transform) 449

- Schattenwürfe 298, 442–445
 - box-shadow
 - Farbangabe (optional) 443
 - Weichzeichnungsradius 443
 - Einfluss auf Element-Layout 444
 - Elemente 330
 - Ersatz in IE 7 und 8 445–448
 - für Druckversion deaktivieren 337
 - innerhalb eines Elements 248, 298, 443
 - Text 330
 - Texttransparenz 156
 - Transparenz definieren 442
- schließende spitze Klammer (>), Selektor für
 - Kindelemente 375
- Schlüsselwörter
 - für font-size 163, 292
 - für font-weight 118
 - inset 298
- Schrägstrich (/), zum Trennen von font-family
 - und line-height in font-Kurzschrift-
 - eigenschaft 126
- Schreibrichtung festlegen 160–161
- Schriftarten
 - festlegen 109–113
 - Namen in UTF-8-Zeichenkodierung
 - angeben 485
- Schriften
 - Betriebssystem 127
 - Buchstabenabstände, Blocksatz und 133
 - Eigenschaften
 - font 125–127
 - ersetzen 110
 - Fallback 130
 - Familien 112, 329
 - Farbe 116–117
 - feste Breite 110
 - geprägt, mit text-shadow 156
 - Gewicht 117–118
 - Groß- und Kleinschreibung 121–122
 - Größe, in harmonisches Verhältnis zu
 - Elementgrößen setzen 463
 - Größen und Maße 113–116
 - in Stylesheet einbetten, per Daten-
 - URI 517–520
 - Kapitälchen 120–121
 - kursive 110, 119
 - lizenzfrei 130, 180
 - mit Serifen 110
 - ohne Serifen 110

- Silbentrennung 133
- Standardgröße 114
- Stil 118–120
- System, angeben mit font-Kurzschrifteigenschaft 126
- Webfonts 128–132
- websicher 110
 - Linux 112
 - Liste erweitern 113
- Wortabstände, Blocksatz und 133
- Zeilenhöhe 122–123
- Schriftenglättung, im Internet Explorer 157
- Schriftfarben 116–117
- Schriftgewicht 117–118
 - numerische Werte 118
- Schriftgröße
 - absolute Werte 114
 - als Referenz für Skalierung 171
 - relative Werte 114
 - rem 116
 - Schlüsselwörter 115
 - Schlüsselwörter, relativ 115
 - Vererbung 114
- Schriftstil 118–120
- screen-Medientyp 55
- Screenreader 181, 279, 498
 - für Browser versteckte Elemente darstellen 497
- JAWS 499
- VoiceOver 499
- Window Eyes 499
- Scrollbar, Erzeugung steuern 99
- Scrollen, fest positionierte Elemente und 95
- Scrollen, overflow-Eigenschaften und 98
- Seiten, nicht ausdrucken 351
- Seitenhierarchie, Navigation, Breadcrumb 232
- Seitenlayout 372, 384–420
 - linearisieren, für Mobilgeräte 371–373
 - zweispaltig 384
 - Spalten mit fester Breite 388
- Seitenumbrüche, anlegen in Druckversion 341–342
- Seitenverhältnis, von Bildern 171
- select-Element 272–274
- ::selection-Pseudoelement 154
- selectivizr (Polyfill) 514
- SelectORacle 527
- Selektoren 6
 - allgemeine Geschwisterelemente 22–23
 - Attribut 23–26, 146, 161, 217, 276, 278, 499–501
 - allgemein 486
 - CSS3 25
 - für benötigte Formularfelder 283
 - für input-Elemente 265
 - für Substrings 339
 - für verschiedene label-Elemente 264
 - gruppieren 25
 - lang 24
 - Name 24
 - reduziert 286
 - Teilwert 24
 - Wert 24
 - Zeichenkette in 25
 - Attributwert, beliebige Zeichenkette in 25
 - Attributwert, Zeichenkette am Ende 25
 - Bedeutung von Position im Dokumentenbaum 12
 - beliebig, zusammenfassen 13
 - benachbarte Geschwisterelemente 20–22, 210, 255, 291, 295, 303, 309, 314, 321, 324, 349
 - benachbarte Geschwisterelemente (+) 318
 - Beziehung zwischen HTML-Elementen und 12
 - Browserunterstützung 26
 - class 264, 273
 - CSS 6
 - CSS3 22
 - nachrüsten mit selectivizr (Polyfill) 514
 - Pseudoklassen für Formulare 284
 - Element 9, 15–16, 207, 254, 329
 - auswählen mit 6
 - für Auswahllisten 272
 - für Formulare 252–254
 - Spezifikation 270
 - th 313
 - für Auswahl nach Eigenarten von HTML-Elementen 12
 - für Elemente 9
 - für Kindelemente 18
 - für unbenannte Dokumententeile 30–34
 - für Zuweisung von Stilen 9–12
 - gruppieren 12–13, 232, 299, 318
 - gruppiert, auf mehrere Zeilen verteilen 13
 - HTML-Elemente auswählen mit 9

- id 9, 26–29, 188, 215, 227, 303, 309, 350, 370
 - Alternative 23
 - Benennung 27
 - Doppelkreuz (#) 29
 - kombinieren 231
 - Verwendung 28
 - zur Auswahl der aktuellen Seite 231
 - zur Gestaltung verschiedener Formulare 252
- Kind 375
 - in CSS 2.1-Spezifikation 20
 - Nachfahren, Unterschied 19
 - Spezifität 195
- Kindelemente 10, 18–20
- Klassen 9, 26–29, 188, 309, 319, 321, 325, 350, 370
 - Alternative 23
 - auf bestimmte Elemente beschränken 27
 - mit Punkt kennzeichnen 27
 - Parallele zu Attributselektor für Teilwerte 24
 - Verwendung 27
- Linkziel 33
- Liste 12
 - Website für 521
- Nachfahren 9, 145, 170, 198
 - Spezifität 18
 - verwenden anstelle von Kind-selektoren 246
- Pseudoelemente
 - CSS3, Kennzeichnung durch doppelten Doppelpunkt (::) 488
 - und Internet Explorer 153
- Pseudoklassen
 - :focus
 - animieren 454
 - :hover
 - animieren 454
 - verdoppeln in Navigationsmenüs 242
 - dynamisch 213, 280
 - für Buttons 276
 - gleiche Spezifität 213
 - Spezifikation 214
 - für Sprachattribute 161
 - kombinieren 223, 282
 - proprietär
 - ::-moz-placeholder 269
 - ::-webkit-placeholder 269
- Spezifikation 12, 14, 16
- Trennen mit Komma 12, 13
- Typ *siehe* Selektoren, Element
- universell 9, 13–14, 337, 345
 - andere Selektoren, kombinieren mit 14
 - für Reset-Regeln 392
 - häufige Verwendung, Reset 13
- verwenden, um Dokumententeile nicht auszudrucken 350
- Semikolon, zur Trennung von Deklarationen 6
- senkrechter Strich (|), horizontale Menüs 207
- se-resize (Wert der Eigenschaft cursor) 462
- serif (Wert der Eigenschaft font-family) 110, 112
- serifenlose Schriften 110
- Serifenschriften 110
- Sharp, Remy 514
- Silbentrennung 133
- Skalen
 - harmonisch, kombinieren 464
 - harmonisch, Online-Werkzeug 463
 - modular (Design) 463–465
- Skalierung
 - anfänglich, in Mobilgeräten anpassen 361
 - von Bildern 170–172
 - von Bildern beschränken 172
- Skalierungsfaktor
 - Zeilenhöhe, für Grundlinienraster 470
- ::slot-Pseudoelement (CSS3, noch nicht unterstützt) 419
- small-Schlüsselwort 115
- smaller-Schlüsselwort 115
- Smartphones 355–383
- Smashing Magazine 420
- Soft Hyphen *siehe* weiche Trennzeichen
- Sonderzeichen
 - hexadezimale Entsprechung (Unicode) 340
 - Unicode 196
 - vor Link einfügen, in Druckversion 340–341
- Souders, Steve 5
- Spalten
 - Hintergrund definieren für 399–402
 - mit unterschiedlicher Höhe 401
 - unecht 399–402
 - zweispaltige Layouts 384
 - Spalten mit fester Breite 388
- span-Element 142, 198, 218
 - anstelle von Radiobuttons und Checkboxes gestalten 271
 - Darstellungsrolle ändern 297

- für Textersetzung 178
- legend-Element umgeben mit 256
- spezielle Regeln für Ausdrücke 334–335
- Spezifikationen
 - CSS
 - Stylesheet überprüfen 476
 - zu Pseudoklassen und -elementen 34
 - CSS 2.1, Kindselektor 20
 - CSS3, :nth-child() und Verwandte 38
 - CSS3, Borders and Backgrounds 87
 - HTML 263
 - HTML 4.01 53
 - HTML, class-Attribut 30
 - HTML5 53
 - HTML5, Deaktivieren von Formular-
elementen 34
 - Selektoren 14
- Spezifität 48, 51, 167, 168, 332, 500
- Berechnung 49
- Druckversion und 338
- gleich bei Pseudoklassen für Link-
zustände 213
- Kindselektoren und 195
- Nachfahrenselektoren 18
- Rollover-Effekte in Tabellen und 316
- Sprites 187–189
 - als Daten-URI ins Stylesheet einbetten 518
- s-resize (Wert der Eigenschaft cursor) 462
- Standardschriftgröße 59, 114
- Standardwerte, für Eigenschaften 8
- Stapelungskontext 97
- Stapelungsreihenfolge 96, 353
- Startwert, für :nth-child() und Verwandte 38
- Statusleiste, Betriebssystemsschrift für 127
- Sticky Footer 408–411
 - Grundversion 410
 - Probleme mit IE8 411
- Stildefinitionen für Definitionslisten 207–209
- Stile
 - definieren für HTML5
 - Elemente erzeugen, in älteren Versionen
von IE 504–505
 - für Druckausgabe, Definition im gleichen
Stylesheet wie für Bildschirm-
darstellung 335
 - für erste Absatzzeile 150–151
 - für Linkziel definieren 222–224
 - Inline
 - im Browser anzeigen 486
 - überschreiben 499–501
 - Vorrang vor internen und externen
Stylesheets 500
 - Selektoren für Zuweisung von 9–12
- style-Attribut 5
 - überschreiben 499
- Stylesheets
 - auf Webserver komprimieren 368
 - Autor 47
 - Benutzer 47
 - Browser 46
 - Unterschiede 509
 - Daten einbetten 517–520
 - Druckausgabe 333–354
 - einbinden per link-Tag, für Druck-
ausgabe 334, 335
 - extern 2, 4
 - extern, url() 65
 - für Ausdrücke, einbinden 334–335
 - für bestimmte Medientypen 53–55
 - für iPad 357
 - für Mobilgeräte mit hoher Auflösung 358
 - für Mobilgeräte zuerst laden 366
 - im HTML-Dokument definieren 5
 - importieren 4
 - in einer Datei zusammenfassen 367
 - Inhalt organisieren für bessere Verwaltung 55
 - Inline 4, 5
 - Warnung 47
 - intern 3–4, 5
 - Laden beschleunigen 367–368
 - mehrere in einer Datei zusammenfassen 367
 - Notfall-Stylesheet, Polyfills und 514
 - Reihenfolge 46–48
 - separat für Bildschirm- und Druck-
ausgabe 335
- Spezifikation 5
- strukturieren 55–56
- validieren 475–477
- validieren, für Mobilgeräte 383
- verschlanken 367
- verschlanken, Online-Werkzeuge 368
- sub-Element 158
- Submit-Buttons 274–277
 - eigene Stile 276
 - grafisch 277–279
 - Text 279–281
- super-Element 158
- sw-resize (Wert der Eigenschaft cursor) 462

Systemadministrator, freundlich stimmen 499
Systemschriften, angeben mit font-Kurzschrift-
eigenschaft 127

T

- Tabellen 301–332
 - Beschriftungen 314–315, 329
 - Bilder, Leerraum entfernen 306–308
 - Breite, fest 326–327
 - colspan-Attribut 323
 - Darstellungsmodell
 - separate Rahmen 303, 312
 - zusammengefasste Rahmen 303, 312
 - dynamisch aus Datenbank erzeugt,
 - Stile für 319
 - Innen- und Außenabstände anpassen 329
 - linearisieren, für Mobilgeräte 373–378
 - Rahmen 301
 - color-Eigenschaft und 303
 - zusammenfassen 329
 - zusammengefasst, Darstellungsregeln 304
 - Spalten
 - eingeschränkte Gestaltung mit CSS 323
 - hervorheben 315
 - in Druckversion auf jeder Seite
 - anzeigen 343–344
 - individuell gestalten 320–324
 - Rollover-Effekte 323
 - Spaltengruppen (colgroup-Element), ein-
geschränkte Gestaltung mit CSS 323
 - Spaltenüberschriften, ausblenden in Mobil-
geräten 377
 - spezielle Eigenschaften für Darstellung 331
 - Überschriften
 - ausblenden in Mobilgeräten 375
 - eigene Stile definieren 313–314
 - th 314
 - Überschriftenzellen 329
 - Zeilen
 - abwechselnde Hintergründe 324–326
 - abwechselnde Gestaltung, mit
JavaScript 326
 - als Blockelemente definieren, für
Mobilgeräte 376
 - hervorheben 315
 - individuell gestalten 317–320
 - Rollover-Effekte 315–317, 330
 - Spezifität und 316
 - zählen 318
 - Zählungskontext 311
 - Zebra 375
 - Zellen 329
 - Abstände definieren 304
 - individuell gestalten 308–311
 - Innenabstände 301–305
 - leere 304
 - leere verstecken 312
 - Rahmen 301–305
 - zusammenfassen 303, 312
 - über mehrere Spalten (colspan) 323
 - Zwischenräume definieren 305–306
 - Zwischenräume entfernen 306
 - table-Element 374
 - table-layout-Eigenschaft 326–327
 - Tab-Navigation 235, 238
 - Tags
 - div 215
 - link 4
 - style, kein HTML-Code 5
 - :target-Pseudoelement 495–497
 - :target-Pseudoklasse 222–224
 - :target-Pseudoklasse für Linkziel 33
 - tbody-Element 311, 330, 331
 - td-Element 311, 329
 - Template Layout, CSS3-Modul 418
 - Testen
 - auf mehreren Plattformen 477–479
 - Browsercam 477
 - Browsererweiterungen 479–481
 - für Mobilgeräte (per SDK) 383
 - Google Analytics als Hilfsmittel 479
 - mit Textbrowser 483
 - Webserver-Logdateien als Hilfsmittel 479
 - Website
 - auf mehreren Plattformen 477–479
 - Zugänglichkeit 483
 - Text
 - ausgewählt, Gestaltung ändern 154
 - Ausrichtung 133–134, 346
 - Blocksatz 133
 - linksbündig 133
 - rechtsbündig 133
 - zentriert 133
 - durch Bilder ersetzen 178–181
 - hochgestellt 158–160
 - Lesbarkeit bei Druckversion, weiß auf
schwarz 337
 - mit Schatten versehen 155–158

- richtige Kodierung angeben 483–486
- Schatten 157
 - für Internet Explorer 7-9 156
 - Richtung 157
 - und Zugänglichkeit 156
- Schattenwürfe 330
- Schreibrichtung 160–161
- Subskript 160
- Superskript 160
- tiefgestellt 158–160
- verbergen mit overflow-y 135
- zentrieren 133–134
- text (Wert der Eigenschaft cursor) 462
- text-align-Eigenschaft 104, 133–134, 290
 - Spezifikation 134
 - zum Zentrieren von Formularelementen 293
- textarea-Element 268–270
 - Darstellungsrolle ändern 269
 - Stildefinitionen für 268
- text-decoration-Eigenschaft 138–140, 346
 - line-through (Wert) 139
 - mehrere Werte 138
 - Nachfahrenelemente und 139
 - none (Wert) 139
 - overline (Wert) 139
 - underline (Wert) 139
- Texteingabefelder, border-Eigenschaft und 293
- Texteingabefelder, Schattenwürfe, innen 266
- Textersetzungs 178–181
 - FIR 179, 181
 - Gilder/Levin 178, 181
 - mit CSS3 content-Eigenschaft 180, 181
 - Off-Left-Technik 148
 - Phark (Off-Left-Technik) 178, 180
 - Urheberrecht und 180
- Textfarbe 61
 - Kontrast und 427
 - mit color-Eigenschaft definieren 294
- text-indent-Eigenschaft 150, 277
 - Textersetzungs 178
- text-shadow-Eigenschaft 155–158, 248, 330, 370
 - geprägte Schrift 156
- text-transform-Eigenschaft 121–122
- Textzeilen, am Seitenanfang, Druckversion 342
- Textzeilen, am Seitenende, Druckversion 342
- tfoot-Element 331
- thead-Element 311, 329, 331, 343
- th-Element 311, 313, 329
 - th-Elemente, eigene Stile für 313–314
- Tilde (~), Selektor für allgemeine Geschwister-
elemente 22
- Tipps und Tricks 474–520
- title-Attribut 220
- Tool-Tips 218–222
- top-Eigenschaft 89, 90, 92, 95, 160, 240, 262
- Tracking 124
- transform-Eigenschaft 370, 448–454
 - animieren 458
 - Funktionen 451
 - rotate-Funktion 450
 - simulieren in Internet Explorer 7 und 8 452
 - und position: fixed 453
 - Ursprung angeben mit transform-origin 452
- transform-origin-Eigenschaft 452
- transition-delay-Eigenschaft 455
- transition-duration-Eigenschaft 455
- transition-Eigenschaft 230, 247, 361, 370, 454–461
- transition-property-Eigenschaft 455
- transition-timing-function-Eigenschaft 455
- translate-Funktion (transform) 449
- translateX-Funktion (transform) 449
- translateY-Funktion (transform) 449
- transparente Hintergründe 177
- Transparenz 87–89, 434–438
 - Farbwerte und 62
 - für Schattenwürfe 442
 - Internet Explorer 7 und 8 437
 - mit Bildern 436
 - PNG-Grafikformat 290
 - rgba()-Schreibweise 330
- tr-Element 317–320
 - für Mobilgeräte als Block-Element darstellen 376
 - in Darstellung für Mobilgeräte einbeziehen 375
- Trennstriche, fehlend bei word-wrap 135
- Trennung von Darstellung und Verhalten 223
- Trennung von Deklarationen, mit Semikolon (;) 6
- Trennung von Inhalt und Gestaltung 5, 85, 279
- Trennzeichen, weich, Zeilenumbrüche und 135, 137
- TypeKit (Webfonts) 132
- Typetester 523
- TYPO3 (CMS) 415

- Typografie
 - erste Absatzzeile 150–151
 - Grundlinie, Bilder und 176
 - Grundlinie, Unterstreichungen und 140
 - Grundlinienverschiebung bei hoch-/tief-
gestelltem Text vermeiden 158–160
 - hängende Initialen
 - Grundlinienraster und 468
 - Initialen 146–148
 - grafisch 148–149
 - Pull Quotes 161–164
 - Sonderzeichen, aus anderem Zeichensatz
einfügen 142
 - Überschriften, mit Grafiken über-
lagern 151–153
 - Zeilenhöhe 159
 - Zeilenumbrüche 134–137
 - typografisch korrekte Anführungs-
zeichen 143–146
 - Typselektoren 15
 - Stildefinitionen für select-Elemente 272
 - Typselektoren *siehe* Elementselektoren
- ## U
- Überschriften
 - an Grundlinienraster anpassen 466
 - Überstreichungen, mit border-top 212
 - ul-Element 195
 - Außen- und Innenabstände anpassen 227
 - für Navigation 226–230
 - Umfließen von Elementen 100–102
 - verhindern 101
 - umgebender Block 108
 - Positionierung und 91
 - vertikale Zentrierung und 108
 - umgebendes Element
 - und Positionierungskontext 93
 - Unicode
 - hexadezimale Schreibweise 196, 485
 - hexadezimale Schreibweise, Backslash für 340
 - Zeichen für erzeugte Inhalte 207
 - Zeichenkodierung für Stylesheet 484
 - Unicode (UTF-8) 144
 - universeller Selektor 9, 14, 337
 - für Reset-Regeln 392
 - in Kombination mit anderen Selektoren 14
 - Verwendung 13
 - universeller Selektor (*) 13–14, 345, 510
 - Unterlänge, Darstellung von Bildern und 308
 - Unterschneidung (Kerning) 124
 - Unterstreichungen 138–140
 - für Links 140
 - für Links entfernen 212
 - grafisch 140–142
 - mit border-bottom 212
 - per border-top und border-bottom 140
 - Urheberrecht
 - Schriften 180
 - Webfonts 130
 - <uri> (Wert der Eigenschaft cursor) 462
 - URIs (Uniform Resource Identifiers)
 - angeben für Online-Validierung von
Stylesheets 475
 - Formate 65
 - Fragment-Identifizierer 496
 - in Druckausgabe mit darstellen 338
 - nach Link anzeigen, in Druckversion 338–339
 - url()-Funktion 64–65
 - url()-Funktion, für Daten-URIs 518
 - url()-Schreibweise 73
 - Ursprung
 - für top- und left-Eigenschaften 244
 - für transform-Eigenschaft 452
 - user-scalable, Direktive, für viewport-meta-
Element 365
 - UTF-8 *siehe* Unicode
- ## V
- Validierung 475–477
 - für HTML und CSS gemeinsam 476
 - Variablen
 - definieren mit CSS-Präprozessoren 511
 - Vererbung 16, 44, 280–281
 - Aufzählungszeichen für Listen 200
 - color-Eigenschaft und 228
 - in Formularen 253
 - Vermischung von Darstellung und Inhalt
 - vermeiden 378
 - Verneinung, Pseudoklasse 34
 - verschachtelte Listen, für Navigationsmenüs mit
mehreren Ebenen 246
 - Verschachtelung, Auswahl unabhängig von 16
 - verschiedene Arten, Werte anzugeben 8
 - vertical-align-Eigenschaft 106, 158, 160, 308
 - vertikale Ausrichtung steuern 308

- vertikale Zentrierung 106–108, 465–470
 - Grundprinzipien 106
 - mit absoluten Werten 108
- Verwendung von id-Attributen 27
- VGA-Standard 62
- Viewport *siehe* Ansichtsbereich
- viewport-meta-Element 361
- visibility-Eigenschaft 494
- :visited-Pseudoklasse 212–214
- Vordergrundfarbe 61, 116–117, 197, 303
 - Kontrast und 427
- Vorfahrenelement 16
 - Bedeutung 11
 - Beschreibung 12
- Vorrang *siehe* Spezifität

W

- W3C 356
 - CSS-Seite 525
 - HTML-Validator 526
 - Style-Mailingliste 523
 - Tabelle der CSS-Eigenschaften 8
 - Validatoren 53, 526
 - Validierungsdienst für Stylesheets 475–477
- Waisen, Druckversion 342
- wait (Wert der Eigenschaft cursor) 462
- Walker, Alex 472
- Wartungsaufwand, bei Verwendung von
 - Conditional Comments 502
- WCAG *siehe* Web Content Accessibility Guidelines
- Web Content Accessibility Guidelines 427, 428
- Webaim Color Contrast Checker 427
- Webdesign
 - bedarfsgerecht *siehe* Responsive Webdesign
 - Gemeinsamkeiten mit Musik 465, 469
 - Hintergründe, durchscheinend 88
- Webfarben 62
- Webfonts 128–132
 - @font-face-Generator 130
 - einbinden mit @font-face-Regel 128
 - einbinden per Google Fonts API 128
 - Flash of unstyled text (FOUT) 132
 - Font Squirrel 130
 - Alternativen 132
 - eigene Schriften hochladen 130
 - Fontdeck 132
 - Fontspring 132

- Google Fonts API 131, 132
- im Stylesheet speichern 518
- lizenzfrei 130
- per link-Tag einbinden 128
- rechtliche Probleme 129
- Textersetzung oder 180
- TypeKit 132
 - unterschiedliche Formate 129
- Webkit-Browser, alt-Text und 178
- ::-webkit-input-placeholder-Selektoren 347
- webkit-min-device-pixel-ratio
 - (Media Queries) 358
- ::-webkit-placeholder-Pseudoklasse 269
- webkit-text-size-adjust-Eigenschaft 372
- Webseiten, Layouts 384–420
- websichere Schriften 110
 - Linux 112
 - Liste erweitern 113
- Websites
 - Can I Use, zu Browserunterstützung für
 - CSS-Eigenschaften 481, 517
 - cssninja 361
 - css-wiki.com 521
 - Farbschema wählen 426
 - Quirksmode 361, 481
 - Testen mit Textbrowser 483
- weiche Trennzeichen 137
- Werbebanner, ausblenden in Mobilgeräten 371
- Werte 8
 - Abhängigkeit von Verwendungszweck der
 - Eigenschaft 8
 - Angaben auf verschiedene Weise 8
 - automatisches Zurücksetzen durch
 - Kurzschrifteigenschaft 8
 - background-image 65
 - Farben 8, 61–64
 - benannt 62
 - benannt, Beispiel 61
 - CSS3 62
 - hexadezimal 62
 - hexadezimal, Beispiel 61
 - hsla() 63
 - hsla() oder rgba() 64
 - hsla(), Beispiel 61
 - rgb() und rgba() 62
 - rgb(), Beispiel 61
 - rgb(), Prozentwerte
 - Beispiel 61
 - rgba(), Beispiel 61

- für :nth-child()-Pseudoklasse 35–38
 - für Eigenschaften 6
 - für transition-timing-function 458
 - hexadezimal 8
 - in CSS3-Spezifikation 8
 - inherit, für color-Eigenschaft 228
 - Längeneinheiten, für Positionierung 90
 - mehrmal im gleichen Stylesheet verwenden,
 - CSS-Präprozessoren und 511
 - Pixel und em, Rundungsfehler 245
 - Pixel, unterschiedliche Bedeutung in Mobilgeräten 362
 - Prozent 170–172
 - Rahmenbreite (border-width) 72
 - Rahmenfarbe (border-color) 72
 - repeat-x, für background-repeat 75
 - repeat-y, für background-repeat 75
 - Schlüsselwörter 8
 - für border-Eigenschaft 72
 - überschreiben, mit Kurzschrifteigenschaften 126
 - und Eigenschaften 8
 - url(), Anführungszeichen 65
 - url(), für background-image 73
 - URLs 65
 - white-space-Eigenschaft 292
 - für legend-Elemente in Firefox 256, 258
 - widows-Eigenschaft 342
 - width, Direktive, für viewport-meta-Element 365
 - width-Eigenschaft 60, 178, 353, 371, 379, 389
 - Auswirkungen von box-sizing 490
 - dreispaltiges Layout
 - bei Spalten mit flexibler Breite 391–394
 - horizontale Zentrierung und 103
 - zurücksetzen in Mobilgeräten 372
 - wiederholende Farbverläufe 433
 - Wiederverwenden von CSS-Regeln 5
 - Witwen, Druckversion 342
 - WordPress (CMS) 415
 - word-spacing-Eigenschaft 123–124
 - Spezifikation 124
 - word-wrap-Eigenschaft 134, 135
 - fehlende Trennstriche 135
 - World Wide Web Consortium *siehe* W3C
 - Wortabstand 123–124
 - w-resize (Wert der Eigenschaft cursor) 462
 - Wurzelement 11
 - Pseudoklasse 33
 - www-style-Mailingliste (W3C) 523
- ## X
- x-Achse 77
 - x-Höhe (ex) 58
 - x-Höhe, zum Definieren der Schriftgröße 59
 - XHTMLForum, CSS-Unterforum 523
 - x-large-Schlüsselwort 115
 - x-small-Schlüsselwort 115
 - xx-Schlüsselwort 115
 - xx-small-Schlüsselwort 115
- ## Y
- y-Achse 77
 - YUI 2
 - Reset CSS, Reset-Stylesheet 510
- ## Z
- Zahlenverhältnisse, harmonisch 465
 - Zähler
 - in Dokument verwenden 487
 - Zählungskontext für Tabellenzeilen 311
 - Zebratabellen 324–326
 - Zeichen
 - einfügen vor oder nach Inhalten 486
 - nicht sichtbar, in Tabellen 312
 - Zeichenkette
 - leer, für content-Eigenschaft 207
 - Zeilenbox 308
 - Zeilenhöhe 122–123, 294
 - als Basis für Grundlinienraster 465–470
 - Zeilenumbrüche 134–137
 - automatisch, bei Block-Elementen 40
 - mit weichen Trennzeichen 135, 137
 - word-wrap und 135
 - Zentimeter 58
 - Zentrieren
 - Bilder 105
 - Block-Elemente 103–104
 - horizontal 103–105
 - mit absoluten Werten 104
 - Prozentwerte 103
 - Inline-Elemente 104
 - vertikal 106–108
 - Grundprinzipien 106
 - mit absoluten Werten 108
 - mit display: table-cell 106
 - Prozentwerte 107
 - von Text 133–134

Zikaden-Prinzip 470–473
 Farbverläufe anstelle von Bildern
 verwenden 472
z-index-Eigenschaft 84, 96–98, 153, 220, 353,
 446
Zoll (in) 58
Zugänglichkeit
 :focus-Pseudoklasse für Links 214
 Braille-Zeile 279
 Elemente nur in Hilfsgeräten
 darstellen 497–499
Farbenblindheit 427
Farbkontrast 427–428
Farbschema 426
Rot-Grün-Schwäche 427
Screenreader 279
 Website mit Textbrowser testen 483
Zusammenfassen von Selektoren 12–13
zusammengefasste Außenabstände 67
zweispaltige Layouts 384
 flexibel 384–388
 Spalten mit fester Breite 388–391

Über die Autoren

Christopher Schmitt ist CEO von Heatvision.com, Inc., einem Unternehmen für neue Medien und Design mit Sitz in Tallahassee, Florida.

Christopher ist ein preisgekrönter Webdesigner, der seit 1993 mit dem Web arbeitet. Er ist der Autor von *Designing CSS Web Pages* (New Riders) und *HTML5 Cookbook* (O'Reilly). Außerdem ist er Koautor von *Photoshop CS in 10 Steps or Less* (Wiley) und *Dreamweaver MX Design Projects* (glasshaus) und steuerte vier Kapitel zu *XML, HTML, and XHTML Magic* (New Riders) bei. Christopher schrieb auch Beiträge für das *New Architect Magazine*, *A List Apart*, *Digital Web* und *Web Reference*.

Auf Konferenzen wie The Other Dreamweaver Conference und SXSW hielt Christopher Vorträge über die Verwendung und die Vorteile von CSS-basiertem Design. Er ist Listenbetreuer bei Babble (<http://www.babblelist.com>), einer Mailinglisten-Community, die sich mit fortgeschrittenen Webdesign-Themen beschäftigt.

Auf seiner privaten Website <http://www.christopher.org> zeigt Christopher sein wahres Gesicht und seine neuesten Unternehmungen.

Jorgen W. Lang arbeitet seit 1996 als freiberuflicher Webdesigner und -programmierer. Sein Arbeitsschwerpunkt liegt in der Entwicklung kreativer Ideen und deren Umsetzung mit den bestmöglichen Technologien.

Seit 2001 übersetzt Jorgen für den O'Reilly Verlag Fachbücher zu verschiedenen Webtechnologien, darunter *CSS - Das umfassende Handbuch*, das *CSS Kochbuch*, *PHP Hacks*, *jQuery - Von Kopf bis Fuß* sowie *Websites erstellen - Das Missing Manual*. Die vorliegende Ausgabe des CSS Kochbuchs hat er grundlegend überarbeitet, aktualisiert und erweitert.

Jorgen lebt in der Nähe von Oldenburg/Niedersachsen.

Kolophon

Das Tier auf dem Cover von *CSS Kochbuch* ist ein Grizzlybär (*Ursus arctos horribilis*). Die kennzeichnenden Merkmale eines Grizzlys sind seine buckligen Schultern, die lange Schnauze und die langen gekrümmten Krallen. Die Farbe des Fells variiert von blond über braun bis schwarz. Die äußersten Spitzen der Haare weisen häufig einen weißen oder silbernen Schimmer auf, was ihm eine graue Gesamterscheinung gibt. Ausgewachsene Grizzlys wiegen bis zu 400 kg, bei einer Schulterhöhe von 1,50 m. Wenn sie sich auf ihre Hinterbeine aufrichten, erreichen sie eine imposante Größe von bis zu 2,50 m. Trotz ihrer Masse laufen sie mit bis zu 60 km/h überraschend schnell.

Grizzlys sind Allesfresser, ernähren sich aber hauptsächlich von Nüssen, Beeren, Insekten, Lachs, Aas und kleinen Säugetieren. Der Speiseplan variiert je nach Jahreszeit und Lebensraum. In Alaska ernähren sie sich vor allem von Lachs, während sie in den Hochgebirgsregionen vor allem von Beeren und Insekten leben.

Grizzlys sind Einzelgänger. Man begegnet ihnen vor allem in den abgelegenen Bergen und Wäldern von British Columbia, Alberta, Yukon und den Northwest Territories (Kanada) und in Alaska, Idaho, Wyoming, Washington und Montana (USA).

Der Grizzly gehört zu den bedrohten Arten: Bevor die ersten Siedler in den Westen kamen, dürfte die Population des Kontinents laut Schätzungen etwa 100.000 Tiere umfasst haben. Durch die Zerstörung seines Lebensraums ist er heute in den Vereinigten Staaten fast ausgerottet; Abholzung, Bergbau, Wilderei und Zersiedelung gefährden die letzten Populationen. Nur in Kanada und Alaska sind noch größere Bestände vorhanden.

Der Umschlagsentwurf dieses Buches basiert auf dem Reihenlayout von Edie Freedman und stammt von Ellie Volckhausen, die hierfür einen Stich des *Dover Pictorial Archive* aus dem 19. Jahrhundert verwendet hat. Das Coverlayout der deutschen Ausgabe wurde von Hanna Dyer mit Quark XPress 4.1 unter Verwendung der Schriftart ITC Garamond von Adobe erstellt. Als Textschrift verwenden wir die Linotype Birka, die Überschriftenschrift ist die Adobe Myriad Condensed und die Nichtproportionalschrift für Codes ist LucasFont's TheSans Mono Condensed. Mary Anne Weeks Mayo hat das Kolophon geschrieben.