PROGRAMMER'S CHOICE

Michael Jendryschik

Einführung in XHTML, CSS und Webdesign

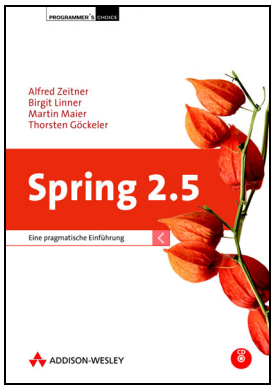
Standardkonforme, moderne und barrierefreie Websites erstellen

2., aktualisierte und erweiterte Auflage

ADDISON-WESLEY

Einführung in XHTML, CSS und Webdesign

Hier eine Auswahl:



Spring 2.5

Alfred Zeitner, Birgit Linner, Martin Maier, Thorsten Göckeler
416 Seiten
€ 39,95 [D], € 41,10 [A]
ISBN 978-3-8273-2622-5

Spring ist ein Open Source Applikationsframework für die Java Plattform mit dem Ziel, Java Anwendungen zu vereinfachen und guten Programmierstil zu fördern. Dabei steht die Entkopplung einzelner Komponenten im Vordergrund. Ziel des Buches ist es, dem Leser einen schnellen und praxisnahen Einstieg im Umgang mit dem Spring-Framework zu ermöglichen. Anhand eines durchgängigen Fallbeispiels werden schrittweise Spring-Techniken eingeführt und jeweils an einem Codebeispiel erläutert. So entsteht anhand von immer komplexeren Anforderungen eines fiktiven Kunden aus einem einfachen Lagerservice ein Lagerverwaltungssystem mit Bestellfunktionalität.

Suchmaschinen-Optimierung

Daniel Koch
352 Seiten
€ 29,95 (D) € 30,80 (A)
ISBN 978-3-8273-2469-6

Suchmaschinen sind für den Großteil der Online-User DAS zentrale Mittel für die Informationssuche. Umso wichtiger ist es daher, die eigenen Seiten möglichst weit oben in den Trefferlisten zu platzieren. Dieses Buch zeigt, wie Suchmaschinen arbeiten und wie man seinen Webauftritt für Google & Co. optimieren kann. Als besonderen Anreiz soll das Buch einen SEO-Wettbewerb vorantreiben. Der orientiert sich an der »Hommingberger Gepardenforelle« von Heise. Dazu werden im Buch die entsprechenden SEO-Methoden anhand eines fiktiven Suchbegriffs vorgestellt. Die Leser können so den Erfolg ihrer Optimierungsversuche in direkter Konkurrenz zueinander verfolgen.

Michael Jendryschik

Einführung in XHTML, CSS und Webdesign

Standardkonforme, moderne und barrierefreie Websites erstellen

2., aktualisierte und erweiterte Auflage

ADDISON-WESLEY

An imprint of Pearson Education

München • Boston • San Francisco • Harlow, England
Don Mills, Ontario • Sydney • Mexico City
Madrid • Amsterdam

eBook

Die nicht autorisierte Weitergabe dieses eBooks an Dritte ist eine Verletzung des Urheberrechts!

Bibliografische Information Der Deutschen Nationalbibliothek

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

Die Informationen in diesem Produkt werden ohne Rücksicht auf einen eventuellen Patentschutz veröffentlicht.
Warennamen werden ohne Gewährleistung der freien Verwendbarkeit benutzt.
Bei der Zusammenstellung von Abbildungen und Texten wurde mit größter Sorgfalt vorgegangen.
Trotzdem können Fehler nicht vollständig ausgeschlossen werden.
Verlag, Herausgeber und Autoren können für fehlerhafte Angaben und deren Folgen weder eine juristische Verantwortung noch irgendeine Haftung übernehmen.
Für Verbesserungsvorschläge und Hinweise auf Fehler sind Verlag und Herausgeber dankbar.

Alle Rechte vorbehalten, auch die der fotomechanischen Wiedergabe und der Speicherung in elektronischen Medien.
Die gewerbliche Nutzung der in diesem Produkt gezeigten Modelle und Arbeiten ist nicht zulässig.

Fast alle Hardware- und Softwarebezeichnungen und weitere Stichworte und sonstige Angaben, die in diesem Buch verwendet werden, sind als eingetragene Marken geschützt.
Da es nicht möglich ist, in allen Fällen zeitnah zu ermitteln, ob ein Markenschutz besteht, wird das ®-Symbol in diesem Buch nicht verwendet.

Umwelthinweis:
Dieses Produkt wurde auf chlorfrei gebleichtem Papier gedruckt. Um Rohstoffe zu sparen, haben wir auf die Folienverpackung verzichtet.

10 9 8 7 6 5 4 3 2 1

11 10 09

ISBN 978-3-8273-2739-0

© 2009 by Addison-Wesley Verlag,
ein Imprint der Pearson Education Deutschland GmbH,
Martin-Kollar-Straße 10–12, D-81829 München/Germany
Alle Rechte vorbehalten
Lektorat: Brigitte Bauer-Schiewek, bbauer@pearson.de
Sprachlektorat: Sybille Feldmann
Fachlektorat: Manuela Hoffmann
Herstellung: Martha Kürzl-Harrison, mkuerzl@pearson.de
Coverkonzeption und -gestaltung: Marco Lindenbeck, webwo GmbH, mlindenbeck@webwo.de
Satz: Reemers Publishing Services GmbH, Krefeld, www.reemers.de
Druck und Verarbeitung: Kösel, Krugzell (www.KoeselBuch.de)
Printed in Germany

Inhalt

1 Einleitung ... 13

1.1 Vorwort ... 13
1.2 Über dieses Buch ... 14
1.2.1 Entstehung ... 14
1.2.2 Aufbau ... 17
1.2.3 Fehlermeldungen und Verbesserungsvorschläge ... 19
1.2.4 Koautoren ... 19
1.2.5 Vorveröffentlichungen ... 20
1.3 Weiterbildung, Diskussionen und Problemlösungen ... 21
1.4 Danksagung ... 22

2 Grundlagen ... 25

2.1 Begrifflichkeiten: Website, Webseite, Homepage ... 25
2.2 Textauszeichnung: Die erste Komponente des Webs ... 27
2.2.1 Dokumente ... 28
2.2.2 Strukturorientiert schreiben ... 29
2.2.3 Standards: SGML und XML ... 30
2.2.4 Zusammenfassung ... 34
2.3 Hypertext: Die zweite Komponente des Webs ... 34
2.3.1 Eine kurze Geschichte des Hypertexts ... 35
2.4 Das World Wide Web ... 39
2.4.1 Gewirr, Verknüpfungen und Netze ... 40
2.4.2 Das World Wide Web Consortium ... 42
2.5 Technische Fundamente des World Wide Webs ... 44
2.5.1 URI ... 44
2.5.2 HTTP ... 49
2.5.3 MIME-Typen ... 52
2.6 Sprachen des World Wide Webs ... 54
2.6.1 HTML ... 54
2.6.2 XHTML ... 56
2.6.3 Warum sollten Webautoren heute auf XHTML setzen? ... 57
2.6.4 Ausblick: XHTML 2.0, (X)HTML 5 ... 59
2.6.5 CSS ... 61
2.6.6 Spezifikationen, Empfehlungen und Arbeitsentwürfe ... 64
2.7 Browser ... 65
2.7.1 Die Entwicklung bis zum Browserkrieg ... 65
2.7.2 Browserkrieg ... 69
2.7.3 Windows Internet Explorer ... 71
2.7.4 Firefox ... 73
2.7.5 Opera ... 74
2.7.6 Konqueror ... 76
2.7.7 Safari ... 77

2.7.8 Lynx ... 78
2.7.9 Browsertests ... 79
2.7.10 Bezugsquellen und Testumgebungen ... 82
2.8 Webstandards ... 85
2.8.1 Was ist ein Standard? ... 85
2.8.2 Der Sinn von Standards ... 86
2.8.3 Die Realität der Browserhersteller ... 86
2.8.4 Die Realität der Standards ... 87
2.8.5 Standards sind mehr als nur pure Regeln ... 88
2.8.6 Was also sind Webstandards? ... 88
2.9 Barrierefreiheit ... 89
2.9.1 Warum dies auch eine Einführung in barrierefreies Webdesign ist ... 89
2.9.2 Welche Arten von Behinderungen müssen Sie berücksichtigen? ... 91
2.9.3 Verordnungen und Richtlinien ... 97
2.10 Zeichenkodierung ... 104
2.10.1 00101010 – Am Anfang waren null und eins ... 104
2.10.2 Von Bits und Bytes, dezimal und hexadezimal ... 104
2.10.3 Begrifflichkeiten ... 106
2.10.4 Zeichenkodierungen ... 106
2.10.5 Das Problem mit Zeichenkodierungen ... 111
2.10.6 Zeichendarstellung ... 113
2.10.7 Zeicheneingabe ... 114
2.10.8 Die Wahl der »richtigen« Zeichenkodierung ... 118
2.11 Typografie ... 119
2.11.1 Leerzeichen und Festabstände ... 120
2.11.2 Waagerechte Striche ... 121
2.11.3 Anführungszeichen, Apostroph und falsche Freunde ... 125
2.11.4 Punkte ... 128
2.11.5 Zahlen, Einheiten und Währungszeichen ... 129
2.11.6 Mathematische Symbole und Formeln ... 131
2.11.7 Pfeile ... 133

3 Hilfsmittel ... 135
3.1 Editoren ... 135
3.1.1 WYSIWYG-Programme ... 135
3.1.2 Texteditoren ... 138
3.1.3 Fazit ... 142
3.2 Validatoren ... 143
3.2.1 Natürliche und formale Sprachen ... 143
3.2.2 Syntaktische Fehler und proprietäre Erweiterungen ... 145
3.2.3 Gültigkeit und Wohlgeformtheit ... 146
3.2.4 W3C Markup Validation Service ... 147
3.2.5 Schema-Validator ... 150
3.2.6 Grenzen ... 152
3.2.7 CSS-Validator ... 152
3.3 Bildbearbeitung ... 153
3.3.1 Rastergrafiken ... 153
3.3.2 Vektorgrafiken ... 157

3.4 Nützliche Werkzeuge ... 160
3.4.1 Tidy ... 160
3.4.2 FTP ... 164
3.4.3 Xenu's Link Sleuth ... 165
3.4.4 Color Cop ... 166
3.5 Browsererweiterungen ... 167
3.5.1 Firebug ... 168
3.5.2 Web Developer Toolbar ... 169
3.5.3 Html Validator (based on Tidy) ... 170
3.5.4 Fangs ... 170
3.5.5 MeasureIt ... 172
3.5.6 LinkChecker ... 172
3.5.7 Right-Click Lynx viewer ... 172
3.5.8 WAVE-Toolbar ... 173

4 Einführung in XHTML ... 175
4.1 Syntax und Vokabular ... 175
4.1.1 Elementtypen, Elemente, Tags, Elementinhalte ... 175
4.1.2 Attribute ... 177
4.1.3 Kommentare ... 177
4.1.4 Zeichenreferenzen ... 178
4.2 Die globale Struktur eines XHTML-Dokuments ... 180
4.2.1 Die XML-Deklaration ... 181
4.2.2 Die Dokumenttyp-Deklaration ... 181
4.2.3 Wurzelelement: Der Elementtyp html ... 183
4.2.4 Dokumentkopf: Der Elementtyp head ... 184
4.2.5 Dokumentrumpf: Der Elementtyp body ... 185
4.2.6 Hallo-Welt-Beispiel ... 186
4.3 Das Document Object Model ... 188
4.4 Kompatibilitätsrichtlinien ... 191
4.5 Metaangaben: Der Elementtyp meta ... 193
4.5.1 Metaangaben und Suchmaschinen ... 194
4.6 Universalattribute ... 195
4.6.1 Element-Identifikatoren: Die Attribute id und class ... 195
4.6.2 Angabe der natürlichen Sprache: Die Attribute lang und xml:lang ... 197
4.6.3 Inzeilige Formatierung: Das Attribut style ... 199
4.6.4 Kommentierende Informationen: Das Attribut title ... 199

5 Auszeichnung der Inhalte ... 201
5.1 Überschriften und Absätze ... 201
5.1.1 Überschriften: Die Elementtypen h1, h2, h3, h4, h5 und h6 ... 201
5.1.2 Absätze: Der Elementtyp p ... 202
5.1.3 Zeilenumbrüche: Der Elementtyp br ... 204
5.2 Links und Anker ... 205
5.2.1 Anker: Der Elementtyp a ... 205
5.2.2 Dokumentbeziehungen: Der Elementtyp link ... 209

5.3 Bilder ... 212
5.3.1 Grafiken einbinden: Der Elementtyp img ... 212
5.4 Objekte ... 214
5.4.1 Objekte einbetten: Der Elementtyp object ... 214
5.5 Auszeichnung im Text ... 220
5.5.1 Logische Textauszeichnung: Die Elementtypen em, strong, dfn, code, samp, kbd und var ... 220
5.5.2 Kennzeichnen von Dokumentänderungen: Die Elementtypen del und ins ... 223
5.5.3 Abkürzungen und Akronyme: Die Elementtypen abbr und acronym ... 226
5.5.4 Physische Textauszeichnung: Die Elementtypen i, b, tt, big und small ... 227
5.6 Listen ... 229
5.6.1 Ungeordnete Listen: Die Elementtypen ul und li ... 230
5.6.2 Geordnete Listen: Die Elementtypen ol und li ... 230
5.6.3 Definitionslisten: Die Elementtypen dl, dt und dd ... 233
5.7 Tabellen ... 234
5.7.1 Tabellen, Zeilen, Kopf- und Datenzellen: Die Elementtypen table, tr, th und td ... 235
5.7.2 Tabellenüberschrift, Zusammenfassung und Abkürzungen: Der Elementtyp caption und die Attribute summary und abbr ... 236
5.7.3 Zeilengruppen: Die Elementtypen thead, tfoot und tbody ... 237
5.7.4 Spaltengruppen: Die Elementtypen colgroup und col ... 238
5.7.5 Zellen miteinander verbinden: Die Attribute colspan und rowspan ... 243
5.8 Zitate ... 244
5.8.1 Inzeilige Zitate: Der Elementtyp q ... 245
5.8.2 Zitatblöcke: Der Elementtyp blockquote ... 248
5.8.3 Referenz auf die Quelle eines Zitats: Der Elementtyp und das Attribut cite ... 249
5.9 Struktur und Gruppierung ... 250
5.9.1 Gruppierung: Die Elementtypen div und span ... 250
5.10 Formulare ... 252
5.10.1 Formularcontainer: Der Elementtyp form ... 253
5.10.2 Steuerelemente ... 254
5.10.3 Beschriftungen: Der Elementtyp label ... 262
5.10.4 Struktur: Die Elementtypen fieldset und legend ... 263
5.11 Weitere Elemente ... 264
5.11.1 Kontaktinformationen: Der Elementtyp address ... 264
5.11.2 Präformatierter Text: Der Elementtyp pre ... 264
5.11.3 Programmcode eines Scripts: Die Elementtypen script und noscript ... 265

6 Einführung in CSS ... 269
6.1 Design und Styling ... 269
6.1.1 Design ... 269
6.1.2 Styling ... 271
6.1.3 Design und Styling in der Webentwicklung ... 272
6.2 Entwurfskonzepte und Vorteile ... 272
6.3 Einbindung von CSS in ein XHTML-Dokument ... 274
6.3.1 Einbindung per style-Attribut ... 274
6.3.2 Einbindung per style-Element ... 274

6.3.3 Einbindung per link-Element ... 276
6.3.4 Ausgabemedien ... 279
6.4 CSS: Syntax und Vokabular ... 280
6.4.1 Regeln ... 280
6.4.2 Maskierung ... 281
6.4.3 Kommentare ... 282
6.4.4 At-Regeln ... 286
6.5 Selektoren ... 288
6.5.1 Universalselektor ... 289
6.5.2 Typselektor ... 290
6.5.3 Klassen- und ID-Selektor ... 290
6.5.4 Attributselektoren ... 291
6.5.5 Pseudoklassen ... 293
6.5.6 Pseudoelemente ... 296
6.5.7 Kombinatoren ... 299
6.6 Vererbung und Initialwerte ... 300
6.7 Die Kaskade ... 302
6.7.1 Reihenfolge ... 303
6.8 Browserkompatibilität ... 306
6.8.1 Browserunterstützung ... 306
6.8.2 Doctype Switching ... 307
6.8.3 Der Grund für viele IE-Bugs: hasLayout ... 310
6.8.4 CSS-Hacks für den Internet Explorer ... 311

7 Eigenschaften und Werte ... 315
7.1 Werte ... 315
7.1.1 <Farbe> ... 315
7.1.2 <Integer> ... 318
7.1.3 <Zahl> ... 319
7.1.4 <Länge> ... 319
7.1.5 <Prozent> ... 322
7.1.6 <URI> ... 322
7.1.7 <Zeichenkette> ... 323
7.1.8 <inherit> ... 323
7.2 Farben und Hintergründe ... 323
7.2.1 Vordergrundfarbe: Die Eigenschaft color ... 323
7.2.2 Hintergrundfarbe: Die Eigenschaft background-color ... 324
7.2.3 Hintergrundbild: Die Eigenschaft background-image ... 325
7.2.4 Wiederholung eines Hintergrundbilds: Die Eigenschaft background-repeat ... 326
7.2.5 Scrolleigenschaft eines Hintergrundbilds: Die Eigenschaft background-attachment ... 327
7.2.6 Position eines Hintergrundbilds: Die Eigenschaft background-position ... 328
7.2.7 Hintergrund: Die zusammenfassende Eigenschaft background ... 330
7.3 Schrift ... 331
7.3.1 Schriftfamilie: Die Eigenschaft font-family ... 331
7.3.2 Schriftstil: Die Eigenschaft font-style ... 334
7.3.3 Schriftvariante: Die Eigenschaft font-variant ... 335
7.3.4 Schriftgewichtung: Die Eigenschaft font-weight ... 336

7.3.5 Schriftgröße: Die Eigenschaft font-size ... 337
7.3.6 Schrift: Die zusammenfassende Eigenschaft font ... 340
7.4 Text ... 341
7.4.1 Texteinzug: Die Eigenschaft text-indent ... 341
7.4.2 Horizontale Ausrichtung: Die Eigenschaft text-align ... 341
7.4.3 Ausschmückung: Die Eigenschaft text-decoration ... 343
7.4.4 Laufweite: Die Eigenschaft letter-spacing ... 344
7.4.5 Wortabstände: Die Eigenschaft word-spacing ... 345
7.4.6 Groß- und Kleinschreibung: Die Eigenschaft text-transform ... 346
7.4.7 Behandlung von Leerraum: Die Eigenschaft white-space ... 347
7.5 Das CSS-Boxmodell ... 347
7.5.1 Polsterung, Rahmen und Randabstand ... 348
7.5.2 Boxtypen ... 349
7.5.3 Polsterung: Die Eigenschaft padding ... 354
7.5.4 Rahmenstil: Die zusammenfassende Eigenschaft border-style ... 356
7.5.5 Rahmenbreite: Die Eigenschaft border-width ... 358
7.5.6 Rahmenfarbe: Die zusammenfassende Eigenschaft border-color ... 358
7.5.7 Rahmen: Die zusammenfassende Eigenschaft border ... 359
7.5.8 Randabstand: Die Eigenschaft margin ... 360
7.5.9 Boxtyp eines Elements: Die Eigenschaft display ... 363
7.6 Ausrichtung und Elementfluss ... 365
7.6.1 Vertikale Ausrichtung: Die Eigenschaft vertical-align ... 365
7.6.2 Floatierte Boxen: Die Eigenschaft float ... 368
7.6.3 Steuerung des Elementflusses um floatierte Boxen: Die Eigenschaft clear ... 372
7.6.4 YAML ... 375
7.7 Positionierung ... 376
7.7.1 Positionierung: Die Eigenschaft position ... 376
7.7.2 Boxabstände: Die Eigenschaften top, right, bottom und left ... 382
7.7.3 Angabe der Ebene: Die Eigenschaft z-index ... 383
7.8 Breite und Höhe ... 386
7.8.1 Content-Breite: Die Eigenschaft width ... 386
7.8.2 Mindestbreite: Die Eigenschaft min-width ... 389
7.8.3 Maximalbreite: Die Eigenschaft max-width ... 390
7.8.4 Content-Höhe: Die Eigenschaft height ... 390
7.8.5 Mindesthöhe: Die Eigenschaft min-height ... 392
7.8.6 Maximalhöhe: Die Eigenschaft max-height ... 393
7.8.7 Der Boxmodell-Bug ... 394
7.8.8 Höhe der Zeilenbox: Die Eigenschaft line-height ... 396
7.9 Sichtbarkeit und visuelle Effekte ... 398
7.9.1 Sichtbarkeit: Die Eigenschaft visibility ... 398
7.9.2 Überlauf: Die Eigenschaft overflow ... 401
7.9.3 Deckkraft: Die Eigenschaft opacity ... 403
7.9.4 Mauszeiger: Die Eigenschaft cursor ... 406
7.10 Gestaltung von Listen ... 407
7.10.1 Stil der Listenelementmarkierung: Die Eigenschaft list-style-type ... 407
7.10.2 Grafiken als Listenelementmarkierung: Die Eigenschaft list-style-image ... 409
7.10.3 Position der Listenelementmarkierung: Die Eigenschaft list-style-position ... 410
7.10.4 Listenelementmarkierungen: Die zusammenfassende Eigenschaft list-style ... 410

7.11 Das CSS-Tabellenmodell ... 411
7.11.1 Virtuelle Tabellenschichten und Transparenz ... 411
7.11.2 Algorithmen für die Tabellenbreite: Die Eigenschaft table-layout ... 413
7.11.3 Algorithmus für die Tabellenhöhe ... 414
7.11.4 Rahmenmodell: Die Eigenschaft border-collapse ... 415
7.11.5 Rahmenabstand: Die Eigenschaft border-spacing ... 419
7.11.6 Rahmen um leere Zellen: Die Eigenschaft empty-cells ... 421
7.11.7 Ausrichtung von Tabellenüberschriften: Die Eigenschaft caption-side ... 422
7.12 Generierter Content ... 423
7.12.1 Generierter Inhalt: Die Eigenschaft content ... 423
7.12.2 Zähler: Die Eigenschaften counter-increment und counter-reset ... 425

8 Eine Beispielwebsite: Die Kochbar ... 429
8.1 Einleitung ... 430
8.2 Vorbereitung ... 434
8.3 Schritt 1: XHTML-Grundgerüst ... 436
8.4 Schritt 2: Logo ... 438
8.5 Schritt 3: Hauptnavigation ... 440
8.6 Schritt 4: Sidebar ... 442
8.7 Schritt 5: Content und Fußzeile ... 447
8.8 Schritt 6: Styling der Startseite ... 450
8.8.1 Vorgehensweise und Coderichtlinien beim Schreiben eines Stylesheets ... 451
8.8.2 Globale Einstellungen ... 453
8.9 Schritt 7: Regeln für allgemeine Elemente ... 456
8.10 Schritt 8: Styling des Kopfbereichs ... 459
8.11 Schritt 9: Illustrationen ... 465
8.12 Schritt 10: Sidebar ... 468
8.13 Schritt 11: Content ... 477
8.14 Schritt 12: Fußzeile ... 479
8.15 Schritt 13: Über die Kochbar ... 481
8.16 Schritt 14: Impressum ... 485
8.17 Schritt 15: Speisekarte ... 486
8.18 Schritt 16: Öffnungszeiten ... 491
8.19 Schritt 17: Kontaktformular ... 499
8.20 Schritt 18: Rezept des Tages ... 508
8.21 Schritt 19: Browsertests und Bugfixes ... 514
8.22 Das Ergebnis ... 519

9 Anhang ... 521
9.1 Glossar ... 521
9.2 Literaturverzeichnis ... 548

Index ... 557

1 Einleitung

1.1 Vorwort

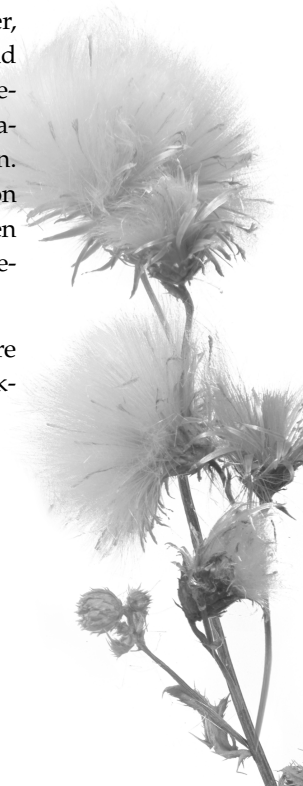
Wenn ich nicht gerade Bücher schreibe, dann schreibe ich **Webseiten**. Ich verdiene meinen Lebensunterhalt damit, Webseiten und Webanwendungen zu konzipieren sowie deren Informationsarchitektur, das **XHTML** und das **CSS** zu liefern. Und ich bin mir sicher, dass viele Leser dieser Einführung dies ebenfalls tun oder es zumindest in Zukunft vorhaben.

Mit den Dingen, die ich hier anspreche, habe ich täglich zu tun, von Projekt zu Projekt mit neuer, mit wachsender Begeisterung, Frustration und Hingabe. Ich hoffe, ein wenig meiner Begeisterung und Freude mit in diese Einführung getragen und die Frustration weggelassen zu haben. Mein Ziel ist es, Ihnen die Entwicklung von standardkonformen **Websites** näherzubringen und meine Erfahrungen und Kenntnisse in diesem Bereich an Sie weiterzugeben.

Ich wünsche Ihnen viel Spaß bei der Lektüre und viel Erfolg bei Ihren kommenden Projekten.

Michael Jendryschik

Dortmund



1.2 Über dieses Buch

1.2.1 Entstehung

Als das **World Wide Web** die Welt eroberte, gab es nur **HTML** – und solange es lediglich eine Plattform für den Austausch von Informationen war, reichte HTML, das als Sprache für einfache **Dokumente** wie Hilfedateien, Protokolle oder technische Dokumentationen entwickelt worden war, auch vollkommen aus. Die Probleme begannen mit dem großen Boom Mitte bis Ende der 1990er-Jahre, als HTML sich von einer strukturorientierten Sprache zunehmend hin zu einem Universalwerkzeug für die Veröffentlichung und visuelle Gestaltung von **Multimedia** wandelte.

Webautoren hatten, bevor es CSS und CSS-fähige **Browser** gab, keine andere Möglichkeit, als Inhalte und Anweisungen für deren Darstellung untrennbar und nicht unterscheidbar in ein Dokument zu schreiben. Zwischen Inhalt und Präsentation bestand kein Unterschied. An klassische Arbeitsteilung, wie sie heute gang und gäbe ist, war damals nicht zu denken. Es gab keine klaren Grenzen zwischen der Arbeit von Designern, Autoren, Entwicklern und Onlineredakteuren. Stattdessen führte die Entwicklung von Webseiten zu allgemeiner Frustration und Unzufriedenheit. Aufwendige, komplexe und verschachtelte Tabellenkonstruktionen zu verwenden, um einfachste Designvorstellungen umzusetzen, widersprach nicht nur dem Empfinden für Ästhetik und **Semantik** der Webentwickler, sondern der Aufwand stand auch in einem nicht vertretbaren Verhältnis zum Ergebnis. Leider gab es (und gibt es noch heute) Entwickler, die ein HTML-Dokument fast ausschließlich aus der Sicht des Layouts betrachteten und dieses in HTML nachzubilden versuchten. Dabei benutzten sie `font`, `br`, feste **Leerzeichen** (` `), transparente Grafiken sowie Layouttabellen im Überfluss und zweckentfremdeten **Elemente**. Hinzu kam, dass sich die Wege der Browserentwickler von den Standards mehr und mehr entfernten und dann auch noch inkompatibel zueinander wurden.

Mit der Entwicklung von CSS und der Aufteilung von HTML in mehrere Sprachvarianten unterschiedlichen Umfangs traf man sich zwar wieder, um einen gemeinsamen Weg zu beschreiten, aber es dauerte eine Weile, bis die Möglichkeiten, die CSS bot, erkannt wurden. In den Jahren davor war das Web jedoch enorm gewachsen und ein einziges Chaos geworden.

Nachdem CSS Ende 1996 als **W3C-Empfehlung** veröffentlicht wurde, war nichts mehr so wie vorher. Nun, genauer gesagt, es dauerte noch eine Weile, bis die Browserlandschaft weit genug war, Webentwicklern die neuen Möglichkeiten bewusst wurden und CSS wirklich genutzt werden konnte. Mit der

Trennung der Präsentation eines Dokuments von dessen Inhalt vereinfachte CSS die Erfassung von Dokumenten und die Verwaltung von Websites. Diese konnten deutlich schneller entwickelt und, was viel bedeutender ist, ungleich einfacher und somit kostengünstiger gewartet und umgestaltet werden.

Anfänge

Ich hatte das Glück, genau zum richtigen Zeitpunkt mit dem Webdesign anzufangen. Im Jahr 1999 begann ich, mich für die Erstellung von Webseiten zu begeistern und erste kleine Websites zu erstellen, zunächst natürlich nur für mich selbst. Ohne wirkliches Verständnis für die Zusammenhänge, ohne das nötige Basiswissen und ohne ein ausreichendes Repertoire benötigter Sprachen – vor allem natürlich HTML, XHTML und CSS – musste ich zwangsläufig scheitern. Das Ergebnis war unbrauchbar und der Bedarf nach hilfreicher Lektüre offensichtlich.

Für viele war SELFHTML damals (und ist es auch heute noch) *die* Standardlektüre zum Thema, zudem die einzig deutschsprachige. Ich fühlte mich jedoch von der Informationsflut, die dieses Werk bot, erschlagen, fand mich nicht zurecht und entdeckte auch nur selten die Lösung für Probleme, die mich gerade beschäftigten.

Usenet

Ganz andere Erfahrungen machte ich im **Usenet**. Das Usenet ist ein textbasiertes Diskussionsforum im Internet. Besser gesagt: Es ist ein Medium, in dem sich Tausende von unterschiedlichen Diskussionsforen zu jedem vorstellbaren Thema in Dutzenden von Sprachen befinden. Man verfasst mithilfe eines Newsreaders einen Beitrag in einem dieser Foren, der dann weltweit von Millionen von Menschen gelesen und beantwortet werden kann.

<dciwam/>

Die **Newsgroup** *de.comm.infosystems.www.authoring.misc* wurde fortan zu meinem virtuellen Zuhause. Die Basis meines heutigen Wissens habe ich mir in zahllosen Diskussionen mit Teilnehmern dieser Gruppe erarbeitet.

Version 1.0

Das vorliegende Buch ist das Ergebnis einer Arbeit, mit der ich im Jahr 2000 begonnen habe. Um das erworbene Wissen zu vertiefen und an andere weiterzugeben, die mit den gleichen Startschwierigkeiten zu kämpfen hatten wie ich, beschloss ich, selbst ein kleines Tutorial zu schreiben und dieses auf meiner Website zu veröffentlichen. Das Ergebnis nannte ich »Einführung in (X)HTML, CSS und Webdesign«, und es war nicht mehr als eine kommentierte Linksammlung, didaktisch und chronologisch aufbereitet und auf das Wesentliche beschränkt. Die Einführung sollte den Zweck erfüllen, dem Anfänger zu sagen, was er für seinen Internetauftritt alles braucht, wie er vorzugehen hat, wie er eine Webseite schreibt und wo er wichtige Artikel zu einzelnen Fragen finden kann. Das Ergebnis können Sie noch heute online betrachten unter *http://archiv.jendryschik.de/einfuehrung/1/*.

Version 2.0 Die Veröffentlichung der zweiten Version nur zwei Jahre später, am 19. Oktober 2002, war eine Dokumentation meiner eigenen Fortschritte. Ich konnte mittlerweile von einer zweijährigen Erfahrung in der Erstellung von CSS-basierten Layouts profitieren, und all diese Erfahrungen flossen in die Einführung mit ein. Ich vollzog den Wandel weg von HTML und hin zu XHTML, darüber hinaus habe ich alle Kapitel stark erweitert, vor allem den CSS-Teil. Die Einführung war im Vergleich zur ersten Version nunmehr ein vollwertiges, für sich allein stehendes Tutorial und zudem eines der ersten, die den Leser wegführten vom klassischen Webdesign mit Layouttabellen, transparenten Grafiken und `font`-Elementen hin zu flexiblen, modernen und schlanken CSS-Layouts. Die Version 2.5.01 finden Sie bei Interesse unter *http://archiv.jendryschik.de/einfuehrung/2/*.

Version 3.0 Mit der Arbeit an der dritten Version dieser Einführung begann ich im September 2005. Ursprünglich war das Ziel, die bestehende Einführung punktuell zu korrigieren und zu erweitern – das Ergebnis war eine komplette Neufassung. Ich habe den grundlegenden Aufbau beibehalten, jedoch alle Inhalte neu geschrieben und stark erweitert. Das CSS-Kapitel habe ich auf den Stand von CSS 2.1 gebracht. Neu hinzugekommen war auch ein durchgängiges Beispiel, die Entwicklung der Website eines kleinen Restaurants mit Biergarten. Es sollte helfen, die mühsam erworbene Theorie in der Praxis anzuwenden. Erstmals erschien die Einführung nicht mehr nur online, sondern im Dezember 2006 auch als Buch.

Version 4.0 In Händen halten Sie nun die zweite Auflage des Buchs, die der Version 4.0 des Onlinetutorials entspricht. Die erste Auflage verkaufte sich gut und beständig, ist allerdings mittlerweile ein wenig in die Jahre gekommen. Die vorliegende Version habe ich vollständig aktualisiert und an einigen Stellen erweitert. Dadurch hat das Buch etwas an Umfang zugelegt. Unter anderem neu hinzugekommen ist das achte Kapitel, »Eine Beispielwebsite: Die Kochbar«, das das in der ersten Version über das ganze Buch verstreute Kochbar-Beispiel in einem Kapitel bündelt. Dadurch erhält es einen stärkeren Projektcharakter und somit einen größeren Praxisbezug.

> > > HINWEIS

Die stets aktuelle Version der Einführung finden Sie unter http://jendryschik.de/wsdev/einfuehrung/. Aktuelle Informationen, die sich speziell auf das Buch beziehen, veröffentliche ich auf der Seite http://jendryschik.de/wsdev/einfuehrung/buch/. Dort finden Sie zudem alle Originalabbildungen (in Farbe) und das komplette Kochbar-Beispiel zum Download.

Diese Einführung in XHTML, CSS und Webdesign richtet sich

Für wen ist diese Einführung?

- vor allem an Anfänger, die das Erstellen von Webseiten mit Webstandards erlernen möchten,
- an Fortgeschrittene, die bereits einige Websites erstellt haben und ihr Wissen nun weiter vertiefen möchten,
- aber auch an Profis, die schnell etwas nachschlagen möchten, beispielsweise Informationen zu einem bestimmten **Elementtyp** oder die erlaubten Werte einer **Eigenschaft**.

Die Lektüre dieser Einführung setzt grundlegendes Interesse voraus, sich mit Webtechnologien und Webstandards auseinandersetzen zu wollen, vor allem mit den Sprachen XHTML und CSS. Falls Sie eher an fertigen Homepage-Vorlagen oder einer Einführung in das schnelle »Zusammenklicken« von Webseiten interessiert sind, werden Ihre Erwartungen hier *nicht* erfüllt. Möchten Sie jedoch lernen, wie Sie standardkonforme, zugängliche (**Zugänglichkeit**) und suchmaschinenfreundliche Webseiten erstellen, die darüber hinaus auch noch gut aussehen, sind Sie hier genau richtig und herzlich willkommen.

1.2.2 Aufbau

Grundlagen

Im Kapitel »Grundlagen« werden Sie Basiswissen erwerben. In der Architektur ist eine Basis (im Sinne von Grundlage, Fundament) der unterste Bauteil einer Säule, auf der der Schaft aufsitzt. Die Basis selbst steht häufig auf einem Sockel, der die Form eines Quaders besitzt. Dieses Bild lässt sich auf dieses Buch übertragen. Der Sockel, auf dem der erste Teil des Buchs aufbaut, sind Ihre grundlegenden Computerkenntnisse. Sie sollten bereits wissen, was ein Ordner beziehungsweise ein Verzeichnis und was eine Datei ist. Sie sollten wissen, wie Sie Ordner und Dateien erstellen und dass es verschiedene Arten von Dateien gibt, die Sie mit unterschiedlichen Programmen öffnen, bearbeiten und erstellen. Es ist nicht unbedingt notwendig, dass Sie bereits einen Internetzugang haben, aber Sie sollten wissen, woher Sie einen solchen bekommen, was das Internet ist und wie Sie einen **Webbrowser** bedienen. Hilfreich wäre es, wenn Sie schon mit unterschiedlichen Browsern gearbeitet hätten, am besten auf unterschiedlichen Systemen, und bereits seit einiger Zeit im World Wide Web unterwegs wären.

Die Basis, die Sie sich in diesem Kapitel erarbeiten, ist das Fundament all dessen, was Sie im weiteren Verlauf dieses Buchs lernen werden. Dabei verstehe ich das Lernen als einen iterativen Prozess. Das bedeutet, das Erlernte festigt und erweitert sich mit jeder Wiederholung des Stoffs. Es mag sein, dass Sie beim Lesen der ersten Kapitel einige Zusammenhänge nicht verstehen können, weil Ihnen das Wissen über XHTML und CSS fehlt und Sie dieses erst in den folgenden Kapi-

teln erwerben werden. Das lässt sich leider nicht vermeiden, wenn man alle Themen mit der notwendigen Tiefe behandeln möchte. Wenn Sie das Buch durchgelesen haben und noch einmal von vorn anfangen, werden Sie feststellen, dass Ihnen auf einmal vieles viel klarer ist, als es Ihnen zu Beginn war, und Sie Neues entdecken, das Ihnen zunächst verborgen geblieben ist.

Hilfsmittel

Zum Handwerkszeug eines Webautors gehören nicht nur viel Erfahrung, eine gesunde Motivation, viel Zeit und viel Kaffee, sondern auch einige Programme und Werkzeuge, die Ihnen die Arbeit an der eigenen Website erleichtern beziehungsweise erst ermöglichen. Das Kapitel »Hilfsmittel« erläutert, woraus Ihre Entwicklungsumgebung mindestens bestehen sollte, bevor Sie richtig loslegen.

XHTML

Die Kapitel »Einführung in XHTML« und »Auszeichnung der Inhalte« erläutern den Aufbau von XHTML-Dokumenten und stellen eine Vielzahl von Elementtypen und **Attributen** vor, die im XHTML-Sprachumfang enthalten sind. Es ist nicht das Ziel dieses Buchs, Ihnen einen vollständigen Überblick über XHTML zu bieten. Diesen erhalten Sie in den entsprechenden Empfehlungen und **Spezifikationen** (siehe Tabelle 2.7). Stattdessen lernen Sie die wichtigsten Elementtypen und Attribute kennen, die Sie zur Erstellung standardkonformer Webseiten benötigen. Viele Teile der Sprache, die nur der Darstellung von Dokumenten dienen und nicht ihrer inhaltlichen Struktur, werden hier nicht erwähnt oder nur am Rande behandelt.

Es ist das Ziel dieser Kapitel, Ihnen zu zeigen, wie Sie zugängliche, suchmaschinenfreundliche XHTML-Dokumente erstellen, die so strukturiert sind, dass sie über ein **Stylesheet** in die gewünschte visuelle Form gebracht werden können. Wichtig ist dabei der Gedanke der Trennung von Struktur und Darstellung, der vielen Webautoren anfänglich Schwierigkeiten bereitet, speziell wenn sie nicht gewohnt sind, über die semantische Struktur von Informationen nachzudenken. Struktur setzt sich aus den Abschnitten eines Dokuments sowie der semantischen und strukturierten Auszeichnung der Dokumentinhalte zusammen. Die Darstellung ist die (visuelle) Gestaltung eines Dokuments. Idealerweise enthalten XHTML-Dokumente keine Informationen über die Darstellung der Inhalte.

CSS

Die Kapitel »Einführung in CSS« und »Eigenschaften und Werte« führen Sie in Cascading Style Sheets (CSS) ein und beschreiben den Aufbau von CSS-Stylesheets entsprechend der Version 2.1. Analog zu den XHTML-Kapiteln wird auch an dieser Stelle nicht der vollständige Sprachumfang erläutert, sondern nur die Eigenschaften und Werte, die im Alltag eines Webautors eine praktische Rolle spielen – wobei dies mittlerweile der Großteil der Empfehlung ist. Dadurch erhalten Sie einen umfangreichen, praxisrelevanten Überblick über die Möglichkeiten der Sprache.

Obwohl CSS es ermöglicht, Stylesheets für unterschiedliche Medien zu schreiben und Webseiten somit beispielsweise für den Ausdruck auf Papier oder für Handhelds zu optimieren, beschränkt sich diese Einführung zum großen Teil auf die Ausgabe auf einem Computerbildschirm. Auf Eigenschaften für die Sprachausgabe wird ebenso wenig eingegangen wie auf Eigenschaften für die Ausgabe in einem konkret bemessenen Medium (Druckausgabe). Darüber hinaus lasse ich alle Eigenschaften und Werte außen vor, die bis heute noch von keinem oder kaum einem Browser richtig interpretiert werden.

Die CSS-Kapitel führen den Gedanken der Trennung von Struktur und Darstellung fort. Sie werden lernen, wie Sie Ihre sauber strukturierten und semantisch reichhaltigen XHTML-Dokumente visuell aufbereiten und wie CSS Sie dabei unterstützen kann, ansprechend aussehende Webseiten zu erstellen.

Im letzten Kapitel werden wir gemeinsam Schritt für Schritt die Website eines kleinen Restaurants aufbauen, der »Kochbar«. Anhand dieses Beispiels werden Sie das erworbene Wissen praktisch vertiefen und lernen, wie die Trennung von **Markup** und Layout, von der ich in diesem Buch häufig spreche, im Projektgeschäft funktionieren kann.

Glossar

Ein Glossar ergänzt diese Einführung. Falls Sie über Fachbegriffe stolpern, die ich nicht näher erläutert habe, finden Sie dort höchstwahrscheinlich eine kurze Erklärung.

1.2.3 Fehlermeldungen und Verbesserungsvorschläge

Ein Buch wie dieses enthält trotz aller Anstrengungen immer noch Fehler und Verbesserungsmöglichkeiten. Kritik und Anregungen sind daher jederzeit willkommen. Bitte verwenden Sie meinen Bugtracker (**Bug**) auf der Seite *http://jendryschik.de/misc/bugtracker/*, um Fehler, die Ihnen auffallen, zu melden. Ich werde mich bemühen, jeden gemeldeten Fehler schnellstmöglich in die Errata *http://jendryschik.de/wsdev/einfuehrung/buch/errata* aufzunehmen und, zumindest in der Onlinevariante, kurzfristig zu beheben. So wird gewährleistet, dass diese Einführung besser wird und aktuell bleibt.

1.2.4 Koautoren

Nicht alle Abschnitte des vorliegenden Buchs habe ich komplett allein geschrieben. An einigen Stellen habe ich die Hilfe von Freunden und Kollegen in Anspruch genommen, die allesamt Experten auf ihrem Gebiet sind. Im Folgenden werden alle am Buch direkt oder indirekt beteiligten Autoren aufgeführt. Ich bedanke mich bei euch für eure Unterstützung und Hilfe!

- Jens Grochtdreis[1] ist Senior Frontend Developer bei SinnerSchrader in Frankfurt. Als »Webstandards Evangelist« hat er 2005 die Webkrauts[2] gegründet und wirbt mit ihnen für ein modernes Web. Ich habe Jens gebeten, für die zweite Auflage dieses Buchs einen Text über Webstandards zu schreiben. Das Ergebnis ist Kapitel 2.8.
- Lars Kasper[3] ist Experte für Webstandards wie (X)HTML und CSS, Zugänglichkeit, Benutzerfreundlichkeit und Informationsarchitektur seit Mitte der 1990er. Für ein gemeinsames Projekt, aus dem aus Zeitmangel leider nichts geworden ist, hat Lars eine ganze Reihe von hervorragenden Artikeln geschrieben. Ein paar davon habe ich mit freundlicher Erlaubnis in dieses Buch übernommen, an einigen Stellen gekürzt und ein wenig verändert. Dazu gehört vor allem das Kapitel 2.10 »Zeichenkodierung«.
- Sierk Bornemann[4] arbeitet als Webentwickler in Hannover und betreibt nebenher das *mod_tidy*-Projekt[5]. Zusammen mit Sierk habe ich einen Artikel über **Tidy** verfasst, von dem einiges in Kapitel 3.4.1 wiederzufinden ist. Für die zweite Auflage hat Sierk das Kapitel überarbeitet und aktualisiert.
- Auf einer CSS-Einführung, die Thomas Scholz ursprünglich für ein gemeinsames Projekt verfasst hat, basieren die Kapitel 6.3 bis 6.7 des CSS-Teils. Ich habe seine Einführung mit freundlicher Erlaubnis zum Teil wörtlich übernommen. Thomas Scholz ist freiberuflicher Webdesigner mit den Schwerpunkten **Barrierefreiheit**, Interfacedesign, CSS und Schreiben fürs Web.

1.2.5 Vorveröffentlichungen

In Teilen dieses Buchs greife ich auf vergangene Veröffentlichungen zurück, vor allem auf Fachartikel, die ich für das iX Magazin für professionelle Informationstechnik verfasst habe, zum Teil als Koautor. Mein Dank gilt Jürgen Seeger, Chefredakteur der iX, und Henning Behme, stellvertretender Chefredakteur, die mir die Wiederverwendung meiner Artikel im Rahmen dieses Buchs gestatteten, sowie den aufgeführten Koautoren.

- Kapitel 2.7.1 enthält Teile aus »Instandsetzung einer Ruine – **Internet Explorer** zwischen 6 und 7«, iX Magazin für professionelle Informationstechnik, Ausgabe 10/2006, S. 131–135.

1 *http://grochtdreis.de/weblog*
2 *http://www.webkrauts.de*
3 *http://www.larskasper.de*
4 *http://www.sierkbornemann.de*
5 *http://mod-tidy.sourceforge.net/*

- Kapitel 3.2 basiert auf den Artikeln »Vom Großen zum Kleinen – EBNF: die erweiterte Backus-Naur-Form verstehen«, erschienen im iX Magazin für professionelle Informationstechnik, Ausgabe 02/2004, S. 116–117, und »Qualitätskontrolle – (X)HTML-**Validatoren** und ihre Grenzen«, veröffentlicht im c't Magazin für Computertechnik, Ausgabe 12/2005, S. 190–191.
- Kapitel 3.4.1 »Tidy« basiert auf dem Artikel »Saubermänner – Tidy/mod_tidy: Webseiten reinigen«, den ich zusammen mit Sierk Bornemann für das iX Magazin für professionelle Informationstechnik, Ausgabe 10/2005, S. 162–165, verfasst habe.
- Die Abschnitte über CSSDOC in Kapitel 6.4.3 basieren auf dem Artikel »Anmerkungen zum Stil – Stylesheets mit CSSDOC kommentieren«, iX Magazin für professionelle Informationstechnik, Ausgabe 03/2008, S. 132–134.
- Kapitel 6.5 »Selektoren« enthält Teile aus dem Artikel »Richtige Ansprache – CSS3-**Selektoren**: Was die Browser (nicht) umsetzen«, erschienen im iX Magazin für professionelle Informationstechnik, Ausgabe 12/2004, S. 122–125. Koautor dieses Artikels ist Thomas Scholz.
- Für Kapitel 6.8.2 »Doctype Switching« konnte ich auf den Artikel »Pinselführung – Doctype-Switching: Wie Browser über die Darstellung von (X)HTML entscheiden« zurückgreifen, erschienen im iX Magazin für professionelle Informationstechnik, Ausgabe 03/2004, S. 136–139.
- Der Artikel »Abwehrzauber – Browserspezifische CSS-Tricks«, veröffentlicht im iX Magazin für professionelle Informationstechnik, Ausgabe 06/2005, S. 138–141, diente mir als Quelle für Kapitel 6.8.4 »CSS-Hacks für den Internet Explorer«.

1.3 Weiterbildung, Diskussionen und Problemlösungen

Wenn Sie inhaltliche Fragen zu diesem Buch haben, sind Sie herzlich eingeladen, diese zu stellen. Denn womöglich habe ich einen Sachverhalt nicht ausreichend erklärt, sodass sich Verbesserungsmöglichkeiten für diese Einführung ergeben. Fragen dieser Art an *michael@jendryschik.de* sind also willkommen.

Bei Fragen zu praktischen Problemen, die sich nicht konkret auf dieses Buch beziehen, wenden Sie sich am besten an die einschlägigen Newsgroups, Mailinglisten oder Webforen.

Usenet

- Zum Thema Usenet finden Sie auf der Seite *http://jendryschik.de/usenet/linkliste/* eine umfangreiche Linkliste, die Ihnen eine Hilfe sein wird, falls Sie nicht wissen, wie Sie Zugang zum Usenet erhalten. Empfehlenswert ist – wie bereits erwähnt – die Gruppe *de.comm.infosystems.www.authoring.misc*, eine umfangreiche Anlaufstelle zu den Themen XHTML, CSS und Webstandards. Diese Gruppe unterhält ebenfalls eine FAQ unter *http://dciwam.de/faq/*.

Mailinglisten

- Auch für Anfänger interessant ist die Mailingliste *css-design*, unterhalten von Claudia Klinger und Michael Charlier. Sie erreichen die Website der Mailingliste unter *http://de.groups.yahoo.com/group/css-design/*.

Webforen

- Die besten Webforen sind das XHTMLforum[6] und das SELFHTML-Forum[7].

Beachten Sie, dass diese Diskussionsmedien keine Supportforen mit Antwortgarantie sind – alles ist freiwillig.

1.4 Danksagung

Zunächst danke ich meiner Familie und meinen Freunden dafür, dass sie mir zum einen die nötige Zeit zum Schreiben dieses Buchs ließen, zum anderen aber hartnäckig genug waren, mich hin und wieder vom Rechner wegzulocken. Vor allem meine wundervolle Freundin Sylvia Schmidt, die sehr viel unter meiner Doppelbelastung leiden musste, zeigte diesbezüglich Durchsetzungsvermögen und ungeahnte Kreativität.

Mein Dank gilt Klaus Langenberg, dessen Website *http://www.thestyleworks.de* zu den besten CSS-Websites gehört, die ich kenne. Sie ist als technischer Ratgeber zum Thema CSS konzipiert und wendet sich an alle, die schon Erfahrung mit der Anwendung von Stylesheets haben und tiefer in die Materie einsteigen möchten. Ich kann jedem meiner Leser empfehlen, sich im Anschluss an diese Einführung auf dieser Site umzuschauen. Klaus Langenberg hat alle CSS 2.1-Eigenschaften im Rahmen seiner Referenz in unterschiedlichen Browsern getestet und mir erlaubt, seine Tests in dieser Einführung zu verwenden. Darüber hinaus waren mir seine Ausführungen Inspiration und Quelle.

Ich bedanke mich bei den Vorständen der itemis AG[8], Jens Wagener, Wolfgang Neuhaus und Dr. Georg Pietrek – nicht nur dafür, dass ich mir ein Dach über dem Kopf und jeden Tag eine warme Mahlzeit leisten kann, sondern auch für ihre Unterstützung. Die itemis AG gibt jedem Mitarbeiter einen Tag der

6 *http://xhtmlforum.de*
7 *http://forum.de.selfhtml.org*
8 *http://www.itemis.de*

Arbeitswoche die Gelegenheit, sich weiterzubilden. Meine Kollegen lernen neue Sprachen, bilden sich technisch fort, schreiben Fachartikel, besuchen Konferenzen oder arbeiten an Open-Source-Projekten mit. Ich habe die Zeit genutzt, um an diesem Buch zu arbeiten.

Danken möchte ich auch der Geschäftsführung der GETIT GmbH Gesellschaft für Technologie und Informationstransfer[9], Dr. Joachim Janoth und Dr. Thomas Krämerkämper, die mich während der Arbeit an der ersten Auflage dieses Buchs unterstützt haben.

Bei der Erstellung des Kochbar-Beispiels unterstützten mich Manuela Hoffmann[10], Ralf Koyro[11] und Konrad Scheller[12]. Manuela hat das Screendesign entworfen und das Fachlektorat des Kochbar-Kapitels übernommen, aus Ralfs Feder stammen die Texte, und Konni erlaubte mir, mich von seiner Speisekarte zu bedienen, um die entsprechende Seite auf der Kochbar-Website mit Leben zu füllen. Ich danke euch für eure Hilfe.

Stefan Mintert[13] und allen, die sich an der Arbeit der edition-W3C.de[14] beteiligt haben, danke ich für ihre Übersetzungen der **W3C**-Empfehlungen, durch die es mir erst möglich wurde, die Standards zu verstehen. Den Webkrauts[15] danke ich dafür, dass sie die Webstandards in Deutschland vorantreiben.

Für besonders lebhaften Gedankenaustausch über Jahre hinweg, vor allem zu Beginn meiner Tätigkeit als Webentwickler, danke ich vor allem Steffi Abel[16], Björn Höhrmann[17] und Michael Nahrath[18].

Last, but not least bedanke ich mich bei allen Lesern der ersten Auflage dieses Buchs und all denen, die regelmäßig meine Websites besuchen. Ohne Sie, Ihre Korrekturvorschläge und Ihre Unterstützung hätte ich dieses Buch niemals veröffentlicht.

9 *http://www.getit.de*
10 *http://www.pixelgraphix.de*
11 *http://www.lindah.de*
12 *http://www.roterochs.de*
13 *http://www.mintert.com*
14 *http://www.edition-w3c.de*
15 *http://www.webkrauts.de*
16 *http://www.abel-consulting.de*
17 *http://bjoern.hoehrmann.de*
18 *http://michael.nahrath.de*

2 Grundlagen

2.1 Begrifflichkeiten: Website, Webseite, Homepage

DEFINITION

*Eine **Website** (auch: Webangebot, Webauftritt, Webpräsenz) ist ein zusammenhängendes Onlineangebot eines Anbieters, das zumeist aus mehreren Webseiten besteht, die sich nicht zwangsläufig auf einem einzigen **Server** befinden beziehungsweise unter einer einzigen Domain erreichbar sein müssen. Eine Website zeichnet sich im Allgemeinen durch einheitliche Navigation und grafische Gestaltung aus.*

DEFINITION

*Eine **Webseite** (auch: Internetseite) ist ein Dokument, das mit einem Benutzerprogramm (zum Beispiel einem Browser) von einem **Webserver** abgerufen und auf einem Bildschirm dargestellt werden kann. Webseiten bestehen zumeist aus strukturiertem Text und Multimedia-Elementen wie Bildern oder Animationen. Jede Webseite ist über mindestens einen **URI** (siehe Kapitel 2.5.1) adressierbar.*

Webseiten können statisch oder dynamisch sein. Ein Kriterium ist die zugrunde liegende Technik.

DEFINITION

*Eine **statische Webseite** besteht aus (sowohl client- als auch serverseitig) unveränderlichen Inhalten. Sie wird in Sprachen realisiert, die in Kapitel 2.6 »Sprachen des World Wide Webs« beschrieben werden. Technisch gesehen, handelt es sich um eine einfache (XHTML-)Datei, die auf einem Webserver oder einem lokalen Laufwerk liegt und andere statische **Ressourcen** einbindet, etwa ein Stylesheet oder Grafiken.*

DEFINITION

*Eine **dynamische Webseite** besteht zusätzlich zu unveränderlichen aus veränderlichen (dynamischen) Inhalten. Dabei unterscheidet man zwischen serverseitiger und clientseitiger Dynamik:* Clientseitige Dynamik *findet im Browser des Benutzers statt, nachdem die Webseite heruntergeladen wurde; bei* serverseitiger Dynamik *wird die Webseite im Moment der Anforderung auf einem Webserver erzeugt.*[1]

JavaScript

Bestimmt haben Sie auch schon einmal animierte Menüs auf Webseiten gesehen. Diese sind oft mit **JavaScript** erstellt, einer clientseitigen **Skriptsprache** (ähnlich einer **Programmiersprache**). Im XHTML-Dokument befindet sich dabei neben dem XHTML-Quellcode noch JavaScript, das der Browser interpretieren muss.

Formulare

Wenn Sie ein Kontaktformular oder ein Gästebuch auf Ihrer Webseite verwenden, reichen die Fähigkeiten des Browsers und clientseitige Techniken im Allgemeinen nicht mehr aus. Die Gästebucheinträge müssen auf dem Server gespeichert werden, also ist hier der Webserver zuständig. Der Webserver muss wissen, was er mit den erhaltenen Daten anfangen soll, die Sie ihm mit dem Absenden eines Kontaktformulars oder Gästebucheintrags geschickt haben. Es gibt etliche Lösungen für solche Aufgabenstellungen. Prinzipiell wird auf dem Server ein Programm (eine Applikation) ausgeführt, wenn etwas gespeichert oder generiert werden soll. Es gibt Webserver, die »von Haus aus« eine oder mehrere Arten von Applikationen unterstützen. Vielen Webservern können, analog zu den Browsern, zusätzliche Funktionen hinzugefügt werden.

Chats

Es gibt ebenfalls Anwendungen, für die sowohl server- als auch clientseitig eine Applikation installiert sein muss. Das bekannteste Beispiel sind Chats. Der Server koordiniert den Chat, und auf dem Client muss ein **Java**-Applet installiert sein, das mit der serverseitigen Applikation zusammenarbeitet (und die Anzeige sofort aktualisiert, wenn jemand im Chat eine Aktion ausgeführt hat). Java-Applets sind spezielle clientseitige Java-Programme, die später noch genauer erläutert werden.

1 Natürlich kann eine serverseitig dynamisch erzeugte Webseite Techniken enthalten, die für eine clientseitige Dynamik sorgen. So kann beispielsweise ein JavaScript serverseitig generiert werden – das ist in der Praxis durchaus üblich.

Nach der Lektüre dieser Einführung werden Sie in der Lage sein, statische Webseiten zu erzeugen. Damit werden Sie Kenntnisse erworben haben, über die auch jeder Programmierer server- oder clientseitiger Applikationen verfügen sollte – schließlich ist das, was ein Browser anzeigt, letztendlich nichts anderes als (X)HTML und CSS. Speichern Sie eine dynamische Webseite lokal ab und betrachten Sie den **Quelltext**, dann verstehen Sie, was ich damit meine. An einigen Stellen werden Sie auch mit JavaScript in Berührung kommen, allerdings nur am Rande, wenn es darum geht, die Benutzerfreundlichkeit zu erhöhen.

DEFINITION

*Die **Homepage** (auch: Startseite, Leitseite) ist die »erste Seite« einer Website, also die Seite, die von einem Benutzer in der Regel als Erstes aufgerufen wird, zu der er immer wieder zurückkehrt und die in der Regel deutlich öfter betrachtet wird als irgendeine andere Seite der Website. Dadurch wird die Homepage zum zentralen Orientierungspunkt einer Website und trägt wesentlich dazu bei, diese als eigenständige Einheit im World Wide Web zu identifizieren. Die Homepage sollte vermitteln, welche Inhalte, Produkte oder Dienstleistungen auf der Website angeboten werden, was der Anbieter macht, den die Website repräsentiert, und wodurch es sich von der Konkurrenz abhebt. Ziel ist die Beantwortung folgender Fragen des Nutzers: Wo bin ich? Was kann ich hier tun?*[2]

Egal ob als multimediale Darstellung seiner selbst oder als Visitenkarte einer Unternehmung im Web, eine Website gehört mittlerweile zum Menschen des 21. Jahrhunderts. Die Denic-Datenbank führte im September 2008 mehr als 12 Millionen .de-Domains auf[3]. Und täglich kommen Tausende hinzu.

2.2 Textauszeichnung: Die erste Komponente des Webs

Viele Webdesigner fühlen sich beim Begriff »Textauszeichnung« überhaupt nicht angesprochen, dabei wären sie überrascht, erklärte man ihnen, dass damit ein wesentlicher Teil ihrer Arbeit gemeint sei. Keine zugängliche Website kommt ohne sinnvolle Textauszeichnung aus, keine Website kann hohe Suchmaschinenpositionen erreichen, wenn an der Auszeichnung ihrer Inhalte nur nebenbei gearbeitet würde. Warum das so ist, wird klar, wenn man einen Blick auf die Geschichte der elektronischen Textauszeichnung wirft[4].

2 siehe auch ***[Storrer 1999]***
3 *http://www.denic.de/de/domains/statistiken/domainentwicklung/*
4 vgl. ***[Behme, Mintert 2000]***

2.2.1 Dokumente

»Das ist ein wichtiges Dokument«, sagen wir gern zu Verträgen, Geschäftsbriefen, Zeugnissen oder beglaubigten Schriftstücken. Wir alle kennen den Begriff »Word-Dokument«. Webentwickler sprechen von »HTML-Dokumenten«. Aber was sind Dokumente eigentlich?

DEFINITION

*Ein **Dokument** ist in der elektronischen Datenverarbeitung eine (nicht ausführbare) Datei mit Inhalt, der in der Regel von Menschen gelesen werden kann. Es besteht aus Inhalt, Struktur, Informationen zur visuellen Darstellung und semantischen Informationen. Der Inhalt eines Dokuments setzt sich vor allem aus Texten zusammen, aber auch – gerade im Web – aus Bildern, Animationen oder Videos.*

Da Inhalte für gewöhnlich nicht einfach willkürlich aneinandergereiht, sondern in irgendeiner Form strukturiert sind, kann man auch von *strukturierten Inhalten* sprechen.

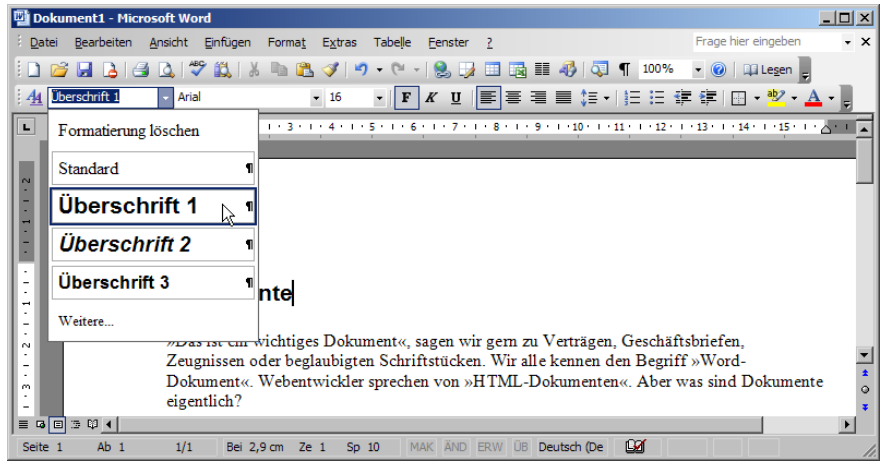


Abbildung 2.1: Verwendung von Formatvorlagen in Microsoft Word

Semantische Auszeichnung

Um Inhalte visuell ansprechend zu gestalten, bieten moderne Textverarbeitungsprogramme wie zum Beispiel Word zwei Vorgehensweisen an, eine »richtige« und eine »falsche«. Gehen wir von folgendem Beispiel aus: Eine Überschrift soll in einer größeren Schritt und fett formatiert werden. Unerfahrene Anwender wählen oftmals den scheinbar einfacheren, aber falschen Weg und formatieren die entsprechenden Textstellen direkt. Damit treffen sie zwar eine Aussage über die visuelle Darstellung des Texts und erreichen so ihr pri-

märes Ziel, allerdings fehlt eine Aussage über die Bedeutung des Texts, die *Semantik*, an dieser Stelle. Ein Text wird nicht dadurch zur Überschrift, dass er groß und fett ist, sondern dadurch, dass man ihn als Überschrift auszeichnet. Der richtige Weg ist, eine semantische Formatierung zuzuweisen – in diesem Fall *Überschrift 1* (siehe Abbildung 2.1) – und visuelle Einstellungen für diese Formatvorlage vorzunehmen. Diese Art des strukturorientierten Schreibens hat viele Vorteile.

2.2.2 Strukturorientiert schreiben

generic coding, generic markup

Der Grundgedanke des strukturorientierten Schreibens liegt in der Trennung des Informationsgehalts eines Dokuments von dessen äußerer Form. Um bei unserem Beispiel zu bleiben: Die semantische Auszeichnung eines Texts als Überschrift und dessen Formatierung (groß und fett) sind zwei unterschiedliche Paar Schuhe. Das Konzept wird als **generic coding** (allgemeine Kodierung) bezeichnet und stammt bereits aus den 1960er-Jahren. Ebenfalls aus dieser Zeit stammt der Begriff **generic markup**, der auf den New Yorker Buchdesigner Stanley Rice zurückgeht. »Markup« bedeutet auf Deutsch »Textauszeichnung« und bezeichnete ursprünglich die handschriftlichen Anmerkungen eines Layouters am Rand eines Dokuments, die später im Satz berücksichtigt wurden. Auf diese Weise wurden das Seitenformat bestimmt und Seiteninhalte angeordnet sowie typografische Festlegungen getroffen. Textauszeichnung beschränkte sich damals fast ausschließlich auf visuelle Aspekte eines Texts. Beim generic markup steht die semantische Auszeichnung der Inhalte im Vordergrund. Markierungen machen dabei eine Aussage über die Art der markierten Stelle, zum Beispiel »dies ist eine Überschrift« oder »dies ist ein Zitat«.

Vorteile

Der Vorteil solcher Auszeichnungen liegt auf der Hand: Bei rein visueller Auszeichnung gehen Informationen schnell verloren, weil sie nicht maschinenlesbar sind. Ein Zitat, das so ausgezeichnet wird (beispielsweise kursiv und eingerückt), ist nicht mehr als Zitat erkennbar, wenn der Text vorgelesen wird. Eine semantische Auszeichnung hingegen könnte ein Sprachsynthesizer erkennen und eine entsprechende Anmerkung voranstellen.

Ein anderer Aspekt ist ebenfalls nicht zu vernachlässigen: Eine visuelle Auszeichnung ist unflexibel. Es lassen sich nur schwer Änderungen am Layout durchführen. Wenn beispielsweise Überschriften nicht mehr fett, sondern kursiv dargestellt werden sollen, müssen an allen Überschriften innerhalb eines Dokuments entsprechende Änderungen vorgenommen werden. Werden Überschriften hingegen als solche ausgezeichnet, reicht eine Änderung an der Formatvorlage, um alle Überschriften eines Dokuments anders zu formatieren.

2.2.3 Standards: SGML und XML

Basierend auf den Ideen *generic coding* und *generic markup* entwickelten Charles Goldfarb, Edward Mosher und Raymond Lorie bei IBM im Jahr 1969 die **GML** (Generalized Markup Language[5]), eine **Metasprache**, mit deren Hilfe Sie unterschiedliche Auszeichnungssprachen für Dokumente definieren können. Das Ziel war es, mit auf GML basierenden Sprachen die logische Struktur von Dokumenten zu beschreiben, beispielsweise die Einteilung eines Buchs in Kapitel, Abschnitte, Überschriften und Absätze, ohne das Aussehen des Dokuments zu spezifizieren. Dadurch waren auf GML basierende Dokumente unabhängig von einer bestimmten Plattform oder Anwendung.

Im Laufe der Jahre arbeite Goldfarb daran, GML zu standardisieren. In Zusammenarbeit mit dem **ANSI**, einer US-amerikanischen Stelle zur Normung industrieller Verfahrensweisen, und der **ISO**, einer internationalen Vereinigung von Normungsorganisationen, gelang dies schließlich acht Jahre später im Jahr 1986 als SGML.

DEFINITION

SGML *(Standard Generalized Markup Language) ist eine Metasprache zur Definition von Auszeichnungssprachen für Dokumente. SGML ist als ISO 8879* ***[ISO-8879 1986]*** *standardisiert und darüber hinaus in der europäischen* ***Norm*** *EN 28879:1990 und der* ***DIN*** *EN 28879:1991 beschrieben.*

Behörden, Firmen, Bildungseinrichtungen und Institutionen begannen schnell damit, ihre Daten in SGML zu speichern. Mit HTML entstand Anfang der 1990er der bis heute bekannteste SGML-**Dokumenttyp**. Den großen Durchbruch auf breiter Front schaffte SGML aber nicht. Das liegt daran, dass hinter SGML zwar viele gute Ideen stecken und die Sprache sehr flexibel ist, diese Flexibilität aber mit einer erheblichen Komplexität erkauft wird. Das macht es Softwareentwicklern schwerer, Programme zu schreiben, die korrekt mit allen Facetten von SGML umgehen können.

Mitte der 1990er, als das Web zu boomen begann, bemühte sich eine SGML-Arbeitsgruppe beim **World Wide Web Consortium** (W3C, siehe Kapitel 2.4.2) darum, SGML zu vereinfachen und für SGML so einen Weg ins Web zu finden. Das Ergebnis war XML (Extensible Markup Language) ***[REC-XML10 2004] [REC-XML11 2004]***, das 1998 als W3C-Empfehlung veröffentlicht wurde.

5 Nicht etwa »Goldfarb Mosher Lorie«, wie GML später häufig scherzhaft genannt wurde.

DEFINITION

***XML** ist eine Teilmenge von SGML, man könnte auch sagen: eine abgespeckte Variante von SGML. XML soll es ermöglichen, Daten über das Web auszuliefern, zu empfangen, zu verarbeiten und zu verwalten, ähnlich wie es heute problemlos plattformübergreifend mit HTML möglich ist. XML wurde entworfen, um eine einfache Implementierung und Zusammenarbeit sowohl mit SGML als auch mit HTML zu gewährleisten. Dabei ist XML wie SGML eine Metasprache, mit der andere Auszeichnungssprachen definiert werden können.*

Heute basieren sehr viele Sprachen auf XML, das SGML in nahezu allen Bereichen abgelöst hat. Auch HTML liegt als XHTML mittlerweile in einer XML-**Syntax** vor. In anderen Bereichen gibt es ähnliche Entwicklungen, beispielsweise bei DocBook, einem Dokumentformat zur Erstellung von Büchern, Artikeln und Dokumentationen im technischen Umfeld, das mittlerweile sowohl als SGML als auch als XML vorliegt. Vielfacher Grund für den Einsatz von XML ist die einfachere Syntax: die Definition von SGML umfasst 500 Seiten, jene von XML nur 26.

Die Dokumenttyp-Definition (DTD)

DEFINITION

*Eine **Dokumenttyp-Definition** (DTD) ist ein Satz an Regeln, der benutzt wird, um SGML- oder XML-Dokumenttypen zu definieren. In einer DTD geben Sie unter anderem an, welche Elementtypen in Dokumenten dieses Typs verwendet werden dürfen, wie Elemente ineinander verschachtelt werden müssen, welche Attribute es für welches Element gibt und welche Attributwerte jeweils zulässig sind.*

Um sich ein Bild vom Aufbau einer solchen DTD zu verschaffen, werfen Sie bitte einen Blick auf Listing 2.1. Dieses definiert einen Dokumenttyp zur Auszeichnung eines Literaturverzeichnisses und ist ausführlich kommentiert, sodass Sie das Wesentliche verstehen werden, ohne zu tief in den Aufbau von DTDs einsteigen zu müssen.[6]

```
<!-- Das Wurzelelement heißt 'literaturverzeichnis'. Es darf
  beliebig viele (*) Elemente 'buch' enthalten. -->
<!ELEMENT literaturverzeichnis (buch)*>

<!-- 'buch'-Elemente bestehen aus einem Element 'titel',
  einem Element 'untertitel' (optional), beliebig vielen
  Elementen 'autor', einem Element 'link' (optional) und
```

6 Wenn Sie gern mehr erfahren möchten, empfehle ich Ihnen die DTD-Einführung in SELFHTML ***[Münz et al.]***.

```
  einem Element 'preis'. -->
<!ELEMENT buch (titel, untertitel?, (autor)*, link?, preis)>

<!-- 'buch'-Elemente haben ein Pflichtattribut 'isbn' -->
<!ATTLIST buch
  isbn CDATA #REQUIRED
>

<!-- Die Elemente 'link', 'titel' und 'untertitel' haben
  weder Attribute noch Kindelemente und dürfen lediglich
  Text enthalten. -->
<!ELEMENT link (#PCDATA)>
<!ELEMENT titel (#PCDATA)>
<!ELEMENT untertitel (#PCDATA)>

<!-- Das Element 'autor' besteht aus einem Element 'name' und
  einem Element 'vorname'. Darüber hinaus gibt es ein
  optionales Attribut 'typ', das den Wert 'herausgeber'
  annehmen kann. -->
<!ELEMENT autor (name, vorname)>
<!ATTLIST autor
  typ (herausgeber) #IMPLIED
>
<!ELEMENT name (#PCDATA)>
<!ELEMENT vorname (#PCDATA)>

<!-- Das Element 'preis' darf nur Text enthalten und verfügt
  über ein Pflichtattribut 'waehrung'. -->
<!ELEMENT preis (#PCDATA)>
<!ATTLIST preis
  waehrung CDATA #REQUIRED
>
```

Listing 2.1: Eine DTD für ein Literaturverzeichnis

Folgendes Listing zeigt ein Beispiel für ein Dokument, das auf dieser DTD basiert. Das Dokument ist so geschrieben, dass es sowohl ein SGML- als auch ein XML-Dokument sein könnte.[7] Die erste Zeile stellt einen Verweis auf die DTD dar. Anschließend folgt das Literaturverzeichnis, das aus zwei Büchern besteht. Vergleichen Sie den Aufbau mit den Element- und Attributdefinitionen in der DTD.

7 Ich möchte Sie an dieser Stelle nicht mit Erklärungen darüber aufhalten, inwieweit sich SGML und XML voneinander unterscheiden. Im weiteren Verlauf dieses Buchs, wenn es um den Unterschied zwischen HTML und XHTML geht, werde ich noch einmal darauf zurückkommen.

```
<!DOCTYPE literaturverzeichnis SYSTEM literaturverzeichnis.dtd">
<literaturverzeichnis>
<buch isbn="3-8273-1872-6">
  <titel>XHTML, CSS und Co.</titel>
  <untertitel>Die W3C-Spezifikation für das
    Web-Publishing</untertitel>
  <autor typ="herausgeber">
  <name>Mintert</name>
  <vorname>Stefan</vorname>
  </autor>
  <preis waehrung="EUR">59.95</preis>
</buch>
<buch isbn="3-8273-2477-7">
  <titel>Einführung in XHTML, CSS und Webdesign</titel>
  <untertitel>Standardkonforme, moderne und barrierefreie
    Websites erstellen</untertitel>
  <autor>
  <name>Jendryschik</name>
  <vorname>Michael</vorname>
  </autor>
  <preis waehrung="EUR">39.95</preis>
</buch>
</literaturverzeichnis>
```

Listing 2.2: Beispiel eines Literaturverzeichnisses

HTML-Dokumente sehen übrigens ähnlich aus wie dieses Literaturverzeichnis, nur heißen die Elemente und Attribute anders. Das liegt daran, dass HTML ebenfalls ein SGML-Dokumenttyp ist. Auch für HTML gibt es eine DTD, genauer gesagt mehrere, nur dass diese viel umfangreicher sind als die Literaturverzeichnis-DTDs. In Kapitel 4.1 »Syntax und Vokabular« werden wir den Aufbau von (X)HTML-Dokumenten etwas genauer betrachten und uns auch mit den Begriffen Elementtyp, Element und Attribut näher auseinandersetzen.

Die **Gültigkeit** des Dokuments überprüfen Sie mit sogenannten **Parsern**; diese lesen zunächst die DTD ein und prüfen dann, ob das eigentliche Dokument konform mit der DTD ist (dazu mehr in Kapitel 3.2 »Validatoren«).

DTDs werden sowohl in der SGML- als auch in der XML-Definition beschrieben. Zur Definition von XML-Dokumenttypen stehen heute mit Schemasprachen wie **XML Schema** oder RELAX NG mächtigere Werkzeuge bereit, die wesentlich komplexere Zusammenhänge beschreiben können, als dies innerhalb einer DTD möglich ist. Auf diese Schemasprachen wird im Rahmen dieser Einführung jedoch nicht weiter eingegangen.

2.2.4 Zusammenfassung

Tabelle 2.1 fasst die Entwicklung der Textauszeichnung noch einmal zusammen. Sie können sich sicher vorstellen, dass eine Zusammenfassung in dieser Kürze unvollständig sein muss. Tatsächlich werden lediglich die Eckdaten genannt, die wichtig für die Entwicklung der Textauszeichnung aus der Sicht des World Wide Webs waren.

Zeitraum	Entwicklung	Beteiligte
1960er	generic coding, generic markup	William Tunnicliffe, Stanley Rice
1969	GML	Charles Goldfarb, Edward Mosher und Raymond Lorie (IBM)
1978–1986	SGML	ANSI, ISO und Charles Goldfarb
1989	HTML	Tim Berners-Lee (**CERN**, später W3C)
1998	XML	Tim Bray (Textuality, Netscape), Jean Paoli (Microsoft) und C. M. Sperberg-McQueen (University of Illinois at Chicago, Text Encoding Initiative) für das W3C
2000	XHTML	Steven Pemberton (CWI), Dave Raggett (W3C, HP) u. a. für das W3C

Tabelle 2.1: Zusammenfassung der Entwicklung der Textauszeichnung

2.3 Hypertext: Die zweite Komponente des Webs

Von jeher wurden Schriftstücke so geschrieben, dass sie linear von vorne nach hinten gelesen werden können[8]. Das war bei alten Steintafeln der Fall, und ist auch bei heutigen Druckwerken nicht anders. Dieses Buch, das Sie gerade lesen, ist ebenfalls so aufgebaut, dass es sinnvoll ist, auf der ersten Seite zu beginnen und sich Seite für Seite zum Ende des Buchs voranzuarbeiten. Dies funktioniert in diesem Fall sehr gut, da die Lerneinheiten aufeinander aufbauen. Autoren anderer Bücher hingegen fühlen sich durch die traditionelle lineare Abfolge von Schriftstücken eingeengt. Frederic Vester, der Erfinder des vernetzten Denkens und vielfach ausgezeichnet für seine Bücher, beschreibt die Einschränkungen, mit denen ein Buchautor leben muss:

> *»Das Medium Buch mit der linearen Anordnung eines in diesem Falle ca. 1,9 Kilometer langen Buchstabenwurms ist, wie letztlich die Sprache überhaupt, nur bedingt geeignet für das Thema, das ich mir hier vorgenommen habe. Der Ver-*

8 vgl. ***[Münz 1997]***

such, ein Gesamtbild der vernetzten Wirklichkeit zu geben, die eigentlich nur simultan erfasst werden kann, muss daher zwangsläufig unvollkommen bleiben.« ***[Vester 1980]***

Die Idee des Hypertexts bricht mit den traditionellen Konventionen.

DEFINITION

Wenn man es kompliziert ausdrücken möchte, ist ***Hypertext*** *die nicht lineare Organisation heterogener Objekte, deren netzartige Struktur durch logische Verbindungen zwischen atomisierten Wissenseinheiten hergestellt wird. Einfacher formuliert, bezeichnet Hypertext die nicht lineare Präsentation von Inhalten. Die einzelnen Inhalte werden Knoten genannt. Über Verweise navigiert man von dem einen Knoten zum anderen.*

Gegenüber der linearen Darstellung von Inhalten bietet Hypertext den Vorteil, eine größere Komplexität mit geringerer Redundanz vermitteln zu können. Zudem setzt sich in der Literatur[9] zunehmend die Erkenntnis durch, dass unser vernetztes Denken ähnlich abläuft wie die Strukturen eines Hypertexts. Demnach entspricht die assoziative Struktur von Hypertext eher der Funktionsweise des menschlichen Denkens, was die Informationsaufnahme im Vergleich zu traditionellen Präsentationsformen beschleunigt.

2.3.1 Eine kurze Geschichte des Hypertexts

Hypertextuelle Strukturen sind seit Jahrhunderten bekannt. Dazu gehören Erschließungshilfen wie Inhaltsverzeichnisse, Indizes oder Fußnoten. Aber auch innerhalb eines Buchs ist es möglich, von einer Stelle zu einer anderen zu verweisen – denken Sie nur an Solo-Rollenspiele.

Solo-Rollenspiele

Ein Solo-Rollenspiel ist ein Buch, das in zahlreiche Abschnitte unterteilt ist. Die Abschnitte sind durchnummeriert. Der Spieler beginnt beim ersten Abschnitt und versucht, sich bis zum letzten Abschnitt durchzuschlagen. Am Ende jedes Abschnitts trifft der Spieler Entscheidungen, die jeweils auf unterschiedliche Folgeabschnitte verweisen, sodass sich verschiedene Handlungsstränge ergeben.

Falls Sie mit Rollenspielen im Allgemeinen oder Solo-Rollenspielen im Speziellen nicht vertraut sind, hier ein typisches Beispiel[10]:

»Du eilst eine halbe Stunde oder länger durch den Wald und bahnst dir einen Weg durch Unterholz und Farn. Du stößt auf einen kleinen Bach. Du legst eine kurze Rast ein, wäscht dir das Gesicht und trinkst von dem klaren, kalten Wasser.

9 unter anderem ***[Schulmeister 2007]***
10 entnommen aus ***[Dever, Chalk 1984]***

> *Du fühlst dich wie neugeboren, watest durch den Bach und gehst weiter. Nicht lange danach steigt dir der Geruch von Holzrauch in die Nase. Er scheint aus nördlicher Richtung zu kommen.*
> *Willst du dem Geruch nachgehen und der Sache auf den Grund gehen, lies weiter bei* **128**.
> *Willst du lieber einen Umweg machen, lies weiter bei* **347**.«

Abhängig davon, wie der Spieler sich entscheidet, muss er in diesem Beispiel entweder zu Abschnitt 128 oder zu Abschnitt 347 blättern. Dort wird die Geschichte weitererzählt bis zu einem Punkt, an dem der Spieler sich erneut entscheiden muss. Rollenspielbücher sind eine klassische Hypertextanwendung.

Enzyklopädien

Ein anderes Beispiel sind Enzyklopädien. Die *Encyclopédie* wurde Mitte des 18. Jahrhunderts von Denis Diderot, Jean Baptiste le Rond d'Alembert und über 100 weiteren Autoren verfasst. Sie sollte die Sammlung des gesamten Wissens der damaligen Zeit darstellen und ist die wohl berühmteste frühe Enzyklopädie im heutigen Verständnis. Im Vorwort schreibt d'Alembert:

> *»Bei der lexikalischen Zusammenfassung alles dessen, was in die Bereiche der Wissenschaften, der Kunst und des Handwerks gehört, muss es darum gehen, deren gegenseitige Verflechtungen sichtbar zu machen und mithilfe dieser Querverbindungen die ihnen zugrunde liegenden Prinzipien genauer zu erfassen [...] es geht darum, die entfernteren und näheren Beziehungen der Dinge aufzuzeigen, aus denen die Natur besteht und die die Menschen beschäftigt haben [...].«*

Automatisierter Hypertext

Um einem Querverweis zu folgen, muss der Leser im Buch vor- oder zurückblättern. Dies ist jedoch umständlich, und man findet nicht immer sofort den Abschnitt, den man sucht. Damit Hypertext wirklich funktioniert, muss die Verfolgung der Verweise mechanisiert beziehungsweise automatisiert erfolgen.

Eines der ersten Systeme dieser Art ist eine Maschine, die Raymond Roussel in den 1930er-Jahren als Lesehilfe für seine verschachtelten Texte entwickelte. Dabei handelte es sich um eine Art Wechselrad für Notizzettel, an deren Randfärbung die Verschachtelungstiefe der Notiz abzulesen war. Zwischen den Zetteln wechselt der Leser mit einer Kurbel, sodass er die farblich als zusammenhängend gekennzeichneten Textteile gesondert lesen kann.

Das moderne Hypertextkonzept wurde erstmals von Vannevar Bush im Jahr 1945 in dem Artikel »As We May Think« ***[Bush 1945]*** in der US-amerikanischen Zeitschrift »The Atlantic Monthly« veröffentlicht. Bush war damals Leiter des »US Office of Scientific Research and Development« und überwachte in dieser Position die Entwicklung der ersten Atombombe. Der Artikel erscheint aus heutiger Sicht visionär, denn er nimmt wesentliche Züge der Computerentwicklung vorweg und wird inzwischen als einer der wichtigsten Texte in der Theoriegeschichte der Computer betrachtet.

Bush beschreibt darin seine Vision eines Systems, das er **Memex** nennt und das das Wissen eines bestimmten Gebiets elektronisch aufbereitet leicht zugänglich darstellen kann. Das Besondere an seinem System sind Verweise, denen Nutzer mit elektronischer Hilfe leicht folgen können und die so auch Bücher und Filme aus Bibliotheken einfach aufrufen können. Die Maschine soll die Form eines Schreibtischs haben und eine Kombination aus elektromechanischen Kontrollen und Mikrofilmgeräten beinhalten. Auf zwei nebeneinanderliegenden, berührungssensitiven Bildschirmen sollen Informationsinhalte projiziert werden. Der Benutzer würde in diesen Informationen mit Hebeln vor- und zurückblättern sowie Dokumente speichern und wieder aufrufen können. Außerdem würde es die Möglichkeit geben, Seiten durch »Verknüpfungen« aufeinander verweisen zu lassen.

Memex

Bush schreibt in seinem Text:

> *»Ganz neue Arten von Enzyklopädien werden entstehen, bereits versehen mit einem Netz assoziativer Pfade, bereit, in den Memex eingebaut und dort erweitert zu werden. Der Anwalt hat auf Tastendruck die gesammelten Gutachten und Entscheidungen seines gesamten Berufslebens zur Verfügung, ebenso wie die von Freunden und Autoritäten. Für den Patentanwalt stehen Hunderttausende ausgegebener Patente bereit, mit vertrauten Pfaden zu jedem Punkt, der für seinen Klienten von Interesse sein könnte. Der Arzt, verwundert über die Reaktion eines Patienten, folgt dem Pfad, den er bei der Erforschung eines früheren, ähnlichen Falls angelegt hat, und kann rasch andere Fallgeschichten durchgehen, mit Hinweisen auf die relevanten Klassiker der Anatomie und Histologie. Der Chemiker, der sich mit der Synthese einer organischen Verbindung müht, hat alle Fachliteratur in seinem Labor vor sich, mit Pfaden, die sich mit Vergleichen zwischen Verbindungen befassen, und Seitenpfaden über ihre physikalischen und chemischen Eigenschaften.*
> *Der Historiker, der eine ausführliche Chronologie eines Volkes anlegt, versieht diese parallel mit einem Schnellpfad, der nur die wichtigsten Punkte berührt, und kann jederzeit von dort aus anderen, über die gesamte Zivilisation einer bestimmten Epoche führenden Pfaden folgen. Es wird ein neuer Berufszweig von Fährtensuchern entstehen, die sich damit beschäftigen, nützliche Pfade durch die ungeheure Menge von Aufzeichnungen und Dokumenten anzulegen. Ein Meister kann seinen Schülern nicht nur die eigenen Ergänzungen zu den Aufzeichnungen dieser Welt hinterlassen, sondern das gesamte Gerüst, mit dessen Hilfe sie entstanden sind.«*

Die Begriffe Hypertext und Hypermedia prägte Ted Nelson, US-amerikanischer Gesellschafts- und Computerwissenschaftler, im Jahr 1965 im Rahmen seines Hypertextprojekts **Xanadu**. Nelson konzipierte Xanadu als dezentrales Speichersystem für Dokumente, von denen jedes eine vom Speicherort unabhängige, eindeutige Adresse besitzt. Auch jedes **Zeichen** innerhalb eines Dokuments ist eindeutig adressierbar. Nelson brachte in sein Projekt zahlreiche richtungsweisende Ideen ein, unter anderem:

Xanadu

- **Versionierung**: Jedes in Xanadu abgelegte Dokument stellt einen unlöschbaren Eintrag in einer Datenbank dar. Nach Veröffentlichung einer neuen Version desselben Dokuments bleiben die alten Versionen weiterhin verfügbar. Nutzer können Versionen miteinander vergleichen und Versionsunterschiede einfach sichtbar machen.
- **Mehrfenstertechnik**: Zusammengehörende Dokumente werden in parallelen Fenstern angezeigt. Zwischen diesen Fenstern werden die Verbindungen zwischen den Dokumenten (**Hyperlinks** etc.) sichtbar gemacht.
- **Bidirektionale Verweise**: Wird in Xanadu auf ein Dokument verwiesen, kennt auch das verlinkte Dokument diese Verbindung. So kann das Benutzerprogramm stets auch alle Dokumente anzeigen, die auf das aktuell betrachtete Dokument verweisen.
- **Inhalte einbetten**: Um beispielsweise aus einem Buch zu zitieren, müssen Nutzer das Zitat nicht abtippen, sondern können die entsprechende Stelle einfach verlinken. Benutzerprogramme betten den Text als Zitat ein.

Der Informationswissenschaftler Rainer Kuhlen schreibt über Xanadu:

> *»Das auch heute noch utopisch anmutende Endziel ist dabei die Verwaltung des gesamten Weltwissens über ein riesiges, computerunterstütztes Begriffsnetz, das den Zugriff auf die entsprechenden informationellen Einheiten gestattet. Durch die Möglichkeit der simultanen und kollektiven Bearbeitung eines Dokuments soll der tendenzielle Gegensatz zwischen Autor und Leser aufgehoben werden. Die Aufgabe solcher Systeme beschränkt sich nicht allein auf die Verwaltung der vielfältigen und komplexen Beziehungen zwischen einzelnen Hypertexteinheiten. Es sind darüber hinaus auch Mechanismen erforderlich, die bei extensivem Mehrbenutzerbetrieb die Integrität, Aktualität und auch das Rückverfolgen der Entstehungshistorie eines Dokuments gewährleisten sowie den Schutz von Urheber-, Nutzungs- und Vervielfältigungsrechten unterstützen.« **[Kuhlen 1991]***

Das Xanadu-Projekt fand in der breiten Öffentlichkeit wenig Anklang, was hauptsächlich an seiner Komplexität liegt. Bis heute existieren nur Prototypen, obwohl oder gerade weil Nelsons Ideen und Konzepte weit über das hinausgehen, was heutige Hypertextsysteme, wie Wikis oder das World Wide Web, bieten. Nelson steht dem Web daher bis heute skeptisch gegenüber:

> *»Viele Leute denken, Xanadu war ein Versuch, das World Wide Web zu errichten. Ganz im Gegenteil: Das World Wide Web war exakt das, was wir zu vermeiden versuchten.«*

Augment

1968 präsentierte Douglas C. Engelbart zusammen mit einer Gruppe von 19 Forschern, die gemeinsam am *Augmentation Research Center* des *Stanford Research Institute* im US-Bundesstaat Kalifornien arbeiteten, ein Hypertextsystem namens Augment, das sie seit 1962 entwickelten und das sich mithilfe

eines Holzklotzes mit Sensoren, den Engelbart Mouse nannte, bedienen ließ. Diese Präsentation war die öffentliche Geburt der Maus als Eingabegerät. Darüber hinaus präsentierte Engelbart Innovationen wie Hypertext, Mehrfenstertechnik sowie die Verknüpfung heterogener Informationen über Zeiger. Im Gegensatz zu Xanadu konnte hier eine real existierende Software bestaunt werden. Das Video können Sie sich noch heute anschauen[11].

XLink

Eine der neueren Entwicklungen im Bereich des Hypertexts ist **XLink** (XML Linking Language) ***[REC-XLink10 2001]***. XLink entstand im Jahr 2001 und erlaubt, Elemente in XML-Dokumente einzufügen, um Links zwischen Ressourcen zu erzeugen und zu beschreiben. XLink nutzt XML-Syntax. Links können unidirektional sein wie im heutigen HTML, also ein Link von genau einem Quell- zu genau einem Zielanker (**Anker**), oder aber multidirektional, das heißt unterschiedliche Wege zwischen beliebig vielen Ressourcen beschreiben.

Tabelle 2.2 fasst die Geschichte des Hypertexts kurz zusammen.

Zeitraum	Entwicklung	Beteiligte
1945	Memex	Vannevar Bush
seit 1960	Xanadu	Ted Nelson
1965	»Hypertext«, »Hypermedia«	Ted Nelson
1968	Augment	Douglas C. Engelbart u. a. (Augmentation Research Center)
1989	HTML	Tim Berners-Lee (CERN, später W3C)
2001	XLink	Steve DeRose (Brown University Scholarly Technology Group), Eve Maler (Sun Microsystems) und David Orchard (Jamcracker) für das W3C

Tabelle 2.2: Zusammenfassung der Entwicklung von Hypertext und Hypertextsystemen

2.4 Das World Wide Web

Tim Berners-Lee, derzeit Inhaber des 3Com Founders-Lehrstuhls am Laboratory for Computer Science des Massachusetts Institute of Technology (MIT) und im Juli 2004 für seine Verdienste bei der globalen Entwicklung des Internets von der englischen Königin zum Ritter geschlagen (also *Sir* Tim Berners-Lee), kam »zufällig zur rechten Zeit und mit den passenden Interessen und Neigungen, nachdem der Hypertext und das Internet ihre Volljährigkeit erreicht hatten« ***[Berners-Lee 1999]***. Er verband die Ideen *generic coding*, Hypertext und Internet miteinander und schuf das heute so populäre World Wide Web.

11 *http://sloan.stanford.edu/mousesite/1968Demo.html*

2.4.1 Gewirr, Verknüpfungen und Netze

Enquire

Verfolgt man den Weg, der schließlich in das World Wide Web mündete, zurück an seinen Anfang, gelangt man in das Jahr 1980 zu einem Hypertextprojekt namens *Enquire*, das der damals am CERN-Institut angestellte Informatiker Tim Berners-Lee in seiner freien Zeit programmierte.

Das *CERN* ist eine europäische Organisation für Kernforschung in der Nähe von Genf in der Schweiz, dessen Gelände sich über die Grenze bis nach Frankreich erstreckt. Berners-Lee setzte Enquire (auf Deutsch so viel wie »nachfragen«, »sich erkundigen«) ein, um aufzuzeichnen, welcher der zahlreichen fest oder vorübergehend Angestellten welches Programm geschrieben hatte, welche Programme auf welchen Rechnern ausgeführt wurden und wer an welchem Projekt mitarbeitete. In Enquire konnte er Seiten mit Informationen über Personen, Geräte oder Programme erstellen. Jede Information entsprach einem Knoten, der eine zugehörige Liste mit Links zu anderen Knoten enthielt. Um eine neue Seite anzulegen, musste eine Verknüpfung ausgehend von einer anderen Seite erstellt werden; diese Verknüpfungen wurden bidirektional dargestellt, das heißt, wenn ein Link von A nach B gesetzt war, tauchte A in der Linkliste von B als Rückverweis auf. Das ursprüngliche Enquire existierte jedoch nur auf einer 8-Zoll-Diskette und ging irgendwann verloren.

1988 griff Berners-Lee die Ideen, die in Enquire steckten, wieder auf und begann mit der Arbeit an einem neuen Hypertext-Dokumentationssystem. Es sollte dezentralisiert sein und so weit wie möglich auf Einschränkungen verzichten, um von möglichst vielen Nutzern auf möglichst vielen Systemen verwendet werden zu können. Am CERN gab es Dutzende von Systemen, die parallel verwendet wurden und über keinerlei Schnittstellen verfügten. Systemübergreifender Datenaustausch war somit so gut wie unmöglich. Ein weiteres System einzuführen, wäre zum Scheitern verurteilt gewesen, ebenso der Versuch, auf einem der Systeme aufzubauen und alle anderen fallen zu lassen (und somit all jene vor den Kopf zu stoßen, die sich an eines der fallen gelassenen Systeme gewöhnt hatten). Berners-Lees Ziel war es, eine gemeinsame Basis für die Kommunikation zu schaffen, während jedes System seine Eigenheiten behalten kann. Globaler Hypertext und Internettechnologien sollten dies ermöglichen.

»WWW«

Lassen wir Tim Berners-Lee selbst berichten, wie es zu der Namensgebung für das Projekt kam:

> *»Ich suchte nach Begriffen, die seine neue Art von Struktur verdeutlichen würden. Mesh (Geflecht) oder Information Mesh war eine Idee, aber es klang zu sehr nach Mess (Durcheinander). Ich dachte an Mine of Information (Fundgrube an Informationen), kurz MOI, aber moi bedeutet auf Französisch ich, und das*

*war zu egozentrisch. Eine Alternative war The Information Mine (Die Informationsfundgrube), aber das Akronym TIM war ja noch egozentrischer! Außerdem war die Metapher einer Fundgrube auch nicht ganz korrekt, weil sie nicht den globalen Aspekt des Konzepts beinhaltete, ebenso wenig die Idee des Hypertexts – und sie repräsentierte nur den Erhalt von Informationen, nicht deren Bereitstellung. [...] Dann tauchte ein anderer Name auf, ebenfalls eine einfache Möglichkeit, globalen Hypertext zu repräsentieren. Dieser Name wurde in der Mathematik benutzt, um eine Sammlung von Knoten und Verknüpfungen zu bezeichnen, in der jeder Knoten mit anderen verknüpft werden kann. Der Name spiegelte den Grundzug verteilter Anwendungen und Computer wider, die das System verknüpfen konnte. Er drückte die Hoffnung auf ein möglicherweise globales System aus, und er legte auch einige interessante Symbole nahe. [...] Ich würde mein System World Wide Web nennen.« **[Berners-Lee 1999]***

Anfänge

1990 begann Berners-Lee trotz einiger Widerstände am CERN damit, den Code für das WWW zu schreiben. Er startete mit der Programmierung eines Webclients, der die Erstellung, Betrachtung und Bearbeitung von Hypertextdokumenten sowie später den Zugriff auf Newsgroups erlauben würde. Anschließend schrieb er den Code für das *Hypertext Transfer Protocol* (**HTTP**), über das Computer über das Internet kommunizieren würden, sowie den *Universal Resource Identifier* (URI, später *Uniform Resource Identifier*), das Schema, nach dem Dokumentadressen erstellt und aufgefunden werden können (siehe Kapitel 2.5 »Technische Fundamente des World Wide Webss«). Im Dezember 1990 arbeitete er an seiner *Hypertext Markup Language* (HTML), die beschreibt, wie Seiten mit Hypertextverknüpfungen ausgezeichnet werden (siehe Kapitel 2.6.1). Darüber hinaus programmierte er einen eigenen kleinen Webserver.

Die erste Website

Auf dem Server *info.cern.ch*[12] wurde Weihnachten 1990 die erste globale Hypertextwebsite mit den Spezifikationen von HTTP, URI und HTML sowie allen projektbezogenen Informationen veröffentlicht und von Berners-Lee konsequent weiter mit Inhalten versorgt. Interessierte konnten auf dieser Website erfahren, was das World Wide Web sein sollte, woher man einen Webbrowser und somit Zugriff auf das World Wide Web bekommt und wie man einen eigenen Webserver aufsetzt, um selbst Inhalte zu veröffentlichen.

Das World Wide Web war eröffnet!

In den folgenden Monaten (und Jahren) war Berners-Lee unermüdlich damit beschäftigt, das Web zunächst am CERN und schließlich weltweit bekannt zu machen und zu etablieren – anfangs ziemlich auf sich allein gestellt, später mit immer größer werdender Unterstützung aus der wachsenden Gemeinde der Webnutzer. Immer mehr Server wurden an das Web angeschlossen, die Ber-

12 Heute nur noch als Kopie unter *http://www.w3.org/History/19921103-hypertext/hypertext/WWW/TheProject.html* verfügbar.

ners-Lee konsequent auf *info.cern.ch* verlinkte, aber der ganz große Boom ließ zunächst auf sich warten, da das Web nicht so einfach jedem zugänglich war, weil entsprechende Benutzerprogramme fehlten. Die Browser waren anfänglich rein textorientiert; erst 1992 entstanden die ersten Browser für grafische Benutzeroberflächen – dazu jedoch später mehr.

2.4.2 Das World Wide Web Consortium

Abbildung 2.2: Logo des World Wide Web Consortium

Je größer das World Wide Web wurde, desto wichtiger wurde es, eine Institution zu schaffen, die sich um die technischen Grundlagen und Standards im Web kümmern und die Entwicklung begleiten und vorantreiben sollte. Das CERN, wie erwähnt eine Kernforschungsorganisation, konnte diese Aufgabe nicht leisten. Daher gründete Tim Berners-Lee zusammen mit Al Vezza, damals stellvertretender Direktor des LCS (Laboratory für Computer Science) am MIT (Laboratory for Computer Science des Massachusetts Institute of Technology), im Oktober 1994 das World Wide Web Consortium.

DEFINITION

*Das **World Wide Web Consortium** (W3C) ist ein internationales Konsortium, in dem Mitgliedsorganisationen, ein fest angestelltes Team und die Öffentlichkeit gemeinsam daran arbeiten, Webstandards zu entwickeln. Ziel des W3C ist es, dem World Wide Web dadurch seine vollen Möglichkeiten zu erschließen, dass Protokolle und Richtlinien entwickelt werden, die ein langfristiges Wachstum des Webs sichern*[13].

Aufbau und Organisation

Das W3C-Kernteam besteht heute aus mehr als 60 Forschern und Ingenieuren überall auf der Welt, die die technischen Aktivitäten leiten und die Arbeitsabläufe koordinieren. Der Großteil des Teams arbeitet an den drei Niederlassungen des W3C: am MIT in den Vereinigten Staaten, am ERCIM in Frankreich und an der Keio-Universität in Japan.

Überall auf der Welt und aus vielen verschiedenen Arbeitsgebieten treten Organisationen dem W3C bei, um an einem herstellerneutralen Forum zur Schaffung von Webstandards mitzuwirken. Zurzeit (Mitte 2008) gibt es welt-

13 siehe ***[Birkenbihl 2005]*** und ***[Fischer, Lischke 2003]***

weit über 400 Mitglieder aus Industrie und Forschung, insbesondere Hard- und Softwareunternehmen, Telekommunikationsanbieter sowie Regierungsstellen und Universitäten. Eine vollständige Liste aller Mitglieder finden Sie stets aktuell auf der Seite *http://www.w3.org/Consortium/Member/List*.

Die Arbeiten des W3C werden von einer Kombination aus Mitgliedsbeiträgen, Forschungsmitteln und anderen Quellen öffentlicher und privater Finanzierung getragen. Daneben unterhält es eine Reihe von Büros rund um die Welt. Diese arbeiten mit ihren lokalen Web-Communities daran, W3C-Technologien in den Landessprachen zu verbreiten, die geografische Basis des W3C zu erweitern und internationale Beteiligung an den Aktivitäten zu fördern. Das deutsch-österreichische Büro sitzt in Sankt Augustin.

Recommendation Process

Um sicherzustellen, dass die verabschiedeten Empfehlungen einen möglichst breiten Konsens finden, durchläuft jede Empfehlung ein bestimmtes Verfahren, den sogenannten *Recommendation Process*. Stationen dieses Verfahrens sind unter anderem:

- Working Draft (**Arbeitsentwurf**),
- *Candidate Recommendation* (eine Phase, in der die Ergebnisse von Implementationen verarbeitet werden sollen),
- *Proposed Recommendation* und schließlich
- *Recommendation*.

An jeder dieser Phasen sind sowohl die Mitglieder als auch die interessierte Öffentlichkeit beteiligt, Letztere hauptsächlich über die Teilnahme an den zahlreichen Mailinglisten.

Arbeitsbereiche und Aktivitäten

Die Arbeit des W3C unterteilt sich in fünf Bereiche:

- Architecture Domain (**DOM**, Jigsaw, XML, URI)
- Document Formats Domain (HTML, Stylesheets, Math, Grafik, **Internationalisierung**)
- Interaction Domain (Geräteunabhängigkeit, synchronisierte Multimedia-Anwendungen, Voice Browser)
- Technology and Society Domain (digitale Signaturen, Metadaten, elektronisches Geld, Datenschutz und Datensicherheit)
- Web Accessibility Initiative (Zugänglichkeit, Barrierefreiheit)

Jeder dieser Bereiche ist in Aktivitäten (*Activities*) unterteilt, die wiederum aus Arbeitsgruppen (*Working Groups*, beispielsweise die CSS-Arbeitsgruppe) und Interessengruppen (*Interest Groups*) bestehen, die sich aus Teilnehmern der Mitgliedsorganisationen, dem W3C-Team und den vom W3C eingeladenen Experten zusammensetzen. Arbeitsgruppen befassen sich mit der eigentlichen Entwicklung und Ausarbeitung der technischen Empfehlungen, während die Interessengruppen eher allgemeinere Arbeiten übernehmen. Die Gruppen sichern auch die Koordination mit anderen Standardisierungsgremien und technischen Vereinigungen.

2.5 Technische Fundamente des World Wide Webs

2.5.1 URI

DEFINITION

*Ein **Uniform Resource Identifier** (URI) ist eine Zeichenfolge, die zur Identifizierung einer abstrakten oder physikalischen Ressource dient.*

DEFINITION

***Ressourcen** sind Dokumente und Daten aller Art, wie zum Beispiel Grafiken, Dateien, HTML-Dokumente, Usenet-Beiträge, Telefonnummern, IRC-Channels oder (XML-) Namensräume.*

Das Akronym URI wurde mit **RFC** 1630 ***[RFC-1630 1994]*** im Juni 1992 von Tim Berners-Lee im Rahmen seines Projekts »World Wide Web« am CERN-Institut eingeführt. Aktuell ist RFC 3986 ***[RFC-3986 1998]*** aus dem Jahr 2005.

URIs sind entweder absolut, oder sie sind relativ zu einem Basis-URI.

Absolute URIs

Absolute URIs sind wie folgt aufgebaut:

```
<Schema>:<schemenspezifischer Teil>
```

Der erste Teil eines URI gibt das Schema an, das die Interpretation des folgenden Teils festlegt. Unter *http://www.iana.org/assignments/uri-schemes.html* finden Sie eine Liste aller erlaubten URI-Schemata. Tabelle 2.3 führt die Schemata auf, mit denen Sie es im Webentwickleralltag am häufigsten zu tun haben.

URI-Schema	Bedeutung
`file`	Dateien im lokalen Dateisystem
`ftp`	File Transfer Protocol (**FTP**)
`http`	Hypertext Transfer Protocol
`https`	Hypertext Transfer Protocol Secure
`mailto`	E-Mail-Adresse
`news`	Newsgroup oder Newsartikel

Tabelle 2.3: Häufig verwendete URI-Schemata

Der schemenspezifische Teil vieler URI-Schemata, beispielsweise `http` oder `ftp`, besitzen einen hierarchischen Aufbau. Die in eckige Klammern eingeschlossenen Teile sind optional.

```
<Schema>://[<Benutzer>[:<Passwort>]@]<Server>[:<Port>]/
<Pfad>?<Parameter>
```

DEFINITION

Ports *(Anschlüsse) sind Adresskomponenten, die in Netzwerkprotokollen wie HTTP eingesetzt werden, um Datenpakete den richtigen Diensten (Protokollen) zuzuordnen. Damit können unterschiedliche Dienste unter derselben Adresse und somit auf demselben Rechner betrieben und einzeln angesprochen werden. Ein Webserver beispielsweise ist im Allgemeinen unter Port 80 zu erreichen, FTP-Server für Dateitransfers unter Port 21.*

Relative URIs

Bei relativen URIs werden Schema, Server und Port sowie gegebenenfalls Teile des Pfads weggelassen. Den genauen Aufbau eines URI entnehmen Sie bei Interesse RFC 3986, Abschnitt 3.

Ein URI ist entweder ein URL, ein URN oder beides.

URL

DEFINITION

Der Begriff ***Uniform Resource Locator*** *(*URL*), der im allgemeinen Sprachgebrauch oftmals anstelle von URI verwendet wird, bezeichnet eine Unterart des URI und identifiziert eine Ressource über ihren primären Zugriffsmechanismus.*

Wie Tabelle 2.4 zeigt, ist *http://example.com/verzeichnis/dokument* ein klassischer URL (vergleiche oben angeführten Aufbau):

Schema	`http`
Benutzer	keine Angabe
Passwort	keine Angabe
Server	`example.com`
Port	keine Angabe
Pfad	`verzeichnis/dokument`
Parameter	keine Angabe

Tabelle 2.4: Die Bestandteile des URL *http://example.com/verzeichnis/dokument*

Der URL lässt sich wie folgt lesen:

- Es gibt ein via HTTP verfügbares Dokument,
- das sich auf einem Rechner (Server) `example.com` befindet
- und dort über den Pfad `verzeichnis/dokument` erreichbar ist.

Schemata

Andere Schemata, die Ihnen häufig begegnen werden, sind *ftp://user:password@example.com/* für FTP und *mailto:webmaster@example.com* für die Angabe von E-Mail-Adressen.

Fragmentlinks

Einige URLs verweisen auf eine Stelle innerhalb einer Ressource. Diese Art von URI endet mit #, gefolgt von einem Bezeichner für einen Anker (Fragmentbezeichner). Zum Beispiel zeigt dieser URL auf einen Anker namens `fragment`: *http://example.com/verzeichnis/dokument#fragment*.

Relative URLs

DEFINITION

*Ein **relativer URL** enthält keinerlei Schema-, Server- oder Portinformationen. Sein Pfad verweist auf eine Ressource auf demselben Server wie das aktuelle Dokument, vom dem es ausgeht. Ein relativer URL kann Pfadkomponenten und auch Fragmentbezeichner enthalten.*

Einige Beispiele sollen dies verdeutlichen:

Beispiele

Relative URLs	Erläuterung
datei.zip	Verweis auf eine Ressource `datei.zip`, die sich im selben Verzeichnis befindet wie die Ursprungsdatei.
./datei.zip	Wie oben. Die Zeichenkette `./` verweist auf das aktuelle Verzeichnis und kann im Allgemeinen weggelassen werden.
downloads/datei.zip	Verweis auf eine Ressource `datei.zip` im Unterverzeichnis `downloads`.
../datei.zip	Verweis auf eine Ressource `datei.zip`, die sich eine Verzeichnisebene über der Ursprungsdatei befindet.
../../downloads/files/datei.zip	Verweis auf eine Ressource `datei.zip`, zunächst zwei Verzeichnisebenen nach oben, dann in das Unterverzeichnis `downloads`, von dort in das Unterverzeichnis `files`.

Tabelle 2.5: Beispiele für relative URLs

Verzeichnislistings

Sie können auch auf Verzeichnisse anstatt auf Dokumente verweisen. In dem Fall wird entweder ein sogenanntes Verzeichnislisting präsentiert, das Ihnen den Inhalt des Verzeichnisses anzeigt, oder es befindet sich eine Datei im Verzeichnis, die dann standardmäßig vom Browser angezeigt wird. Welche Dateien das sein können, ist abhängig von der Serverkonfiguration, generell sind das aber zumindest die Dateien `index.html` und/oder `index.htm`. Bei den beiden folgenden Verweisen würde also jeweils dieselbe Ressource angezeigt werden; in diesem Fall eine Webseite.

```
wsdev/einfuehrung/index.html
wsdev/einfuehrung/
```

Verweis relativ zum Document Root

Eine weitere Möglichkeit für relative Verweise ist der Verweis relativ zum *Document Root*, also zum Hauptverzeichnis Ihrer Website, in dem alle anderen Dateien und Unterverzeichnisse liegen. Ein solcher Verweis hat immer einen Schrägstrich (`/`) am Anfang des Verweisziels. Dazu zwei Beispiele:

```
/downloads/datei.zip
/
```

Der erste URL verweist ausgehend vom *Document Root* auf eine `datei.zip` in dem Unterverzeichnis `downloads`. Der zweite URL verweist direkt zum *Document Root*. Es würde also entweder ein Verzeichnislisting oder die Datei `index.html` angezeigt werden.

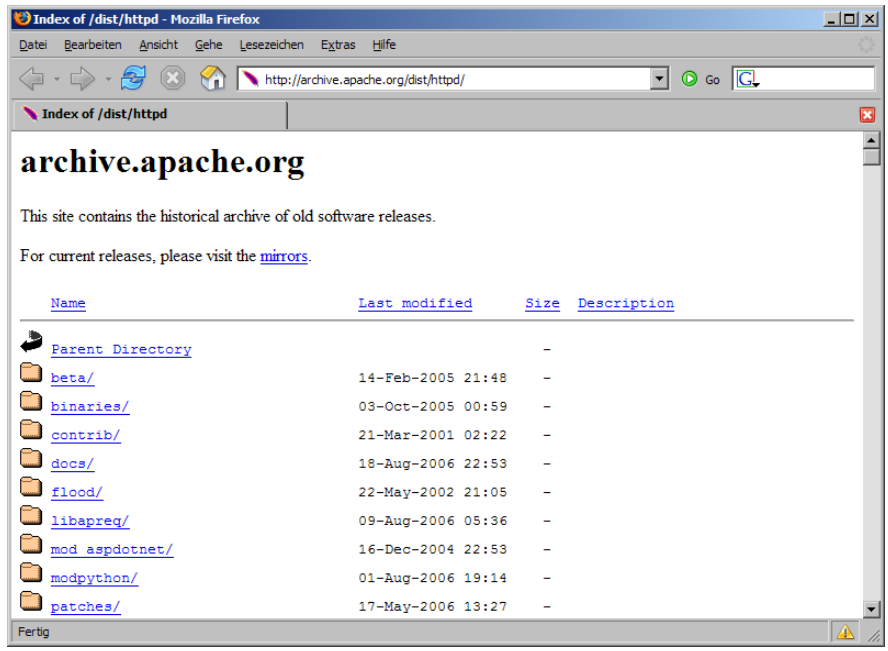


Abbildung 2.3: Beispiel eines Verzeichnislistings

Verweise auf Verzeichnisse und Verweise relativ zum *Document Root* funktionieren nur in einer Webserverumgebung (oder direkt im Stammverzeichnis eines Laufwerks).[14]

Probleme mit URLs

Das Konzept von URLs bringt einige Probleme mit sich. Wenn die Ressource verschoben wird, zum Beispiel auf einen anderen Server, ist sie über den ursprünglichen URL nicht mehr erreichbar. Auch gibt es stets mehrere Möglichkeiten, eine Ressource über URLs zu adressieren. Folgende URLs können unter Umständen zu derselben Ressource führen, und dabei habe ich relative URLs noch ausgelassen:

14 Das Installieren und Einrichten eines Webservers, beispielsweise des **Apache HTTP Server**, ist keineswegs schwierig. Wenn Sie eine lokale Entwicklungsumgebung ausprobieren möchten, empfehle ich Ihnen XAMPP. Dabei handelt es sich um eine Zusammenstellung verschiedener Programme, Dienste und Module, darunter der Apache Webserver, die Datenbank MySQL sowie die Programmier- beziehungsweise Skriptsprachen PHP und Perl. Alle Informationen sowie den Download finden Sie unter *http://www.apachefriends.org/de/xampp.html*.

```
http://example.com/verzeichnis/dokument
http://example.com/verzeichnis/dokument.xhtml
http://www.example.com/verzeichnis/dokument
http://www.example.com:80/verzeichnis/dokument
ftp://ftp.example.com/verzeichnis/dokument
```

URNs könnten dieses Problem beheben.

URN

DEFINITION

*Ein **Uniform Resource Name** (URN) mit dem URI-Schema `urn` identifiziert eine beliebige Ressource mittels eines vorhandenen oder frei zu vergebenden Namens und ist auf Lebensdauer fest mit der Ressource verbunden, ganz gleich, wo und wie diese erreichbar ist. Ein URN besteht im Wesentlichen aus Ziffern, beispielsweise urn:isbn:3-8273-2477-7 oder urn:zbl:0949.68057.*

Unveränderliche Kennung

Anders als URLs, die nicht die Ressource selbst angeben, sondern nur den Ort, an dem sie liegt, vergibt ein URN einer Ressource eine unveränderliche Kennung, anhand deren das Dokument unabhängig von seinem Ablageort identifiziert werden kann, ähnlich einer ISBN bei Büchern.

Übersetzung

URNs können heute nicht direkt aufgerufen werden. Vielmehr müssen sie erst in URLs übersetzt werden, die nicht unbedingt die direkte gesuchte Ressource bezeichnen müssen, sondern auch zu Metainformationen oder Bezugsquellen führen können. So wäre es beispielsweise denkbar, dass ein Benutzer beim Aufruf des oben aufgeführten URNs nicht zum Volltext des Buchs, sondern zur Produktseite seines bevorzugten Onlinebuchhändlers geleitet wird.

Zum Auffinden der Dokumente wird eine zentrale Instanz benötigt, die die URNs in gültige URLs auflöst. Diese Aufgabe übernehmen derzeit die nationalen Bibliotheken.

2.5.2 HTTP

DEFINITION

*Das **Hypertext Transfer Protocol** (HTTP) wurde 1990 vom Tim Berners-Lee im Rahmen seines Projekts »World Wide Web« am CERN-Institut entwickelt. Es ist ein Kommunikationsschema zum Austausch von Daten und noch heute* das *Standardprotokoll des Webs. Es gibt derzeit zwei Versionen: 1.0 **[RFC-1945 1996]** und 1.1 **[RFC-2616 1999]**.*

Verwendung

HTTP wird primär zur Anfrage und Übertragung von Webressourcen verwendet. Durch Erweiterung seiner Anfragemethoden, Header-Informationenen

und Fehlercodes im Rahmen von HTTP 1.1 wird es zunehmend zum Austausch beliebiger Daten zwischen Clients und Servern genutzt, beispielsweise auch zur Durchführung von Dateitransfers (Downloads und Uploads).

Protokollaufbau

Immer dann, wenn Sie einen URI in die Adresszeile eines Browsers tippen und die Ressource anfordern oder wenn Sie einem Link folgen, baut der Browser eine HTTP-Verbindung auf. Bei jeder Verbindung wird zwischen dem Browser (Client) und dem Server, auf dem die angeforderte Ressource liegen soll, ein HTTP-Nachrichtenpaar ausgetauscht, und zwar ein HTTP-Request (Anfrage) des Clients und ein HTTP-Response (Antwort) des Servers. Eine HTTP-Nachricht besteht aus mehreren Zeilen, die durch einen Zeilenumbruch (genauer: **Wagenrücklauf** und **Zeilenvorschub**, kurz CRLF für »Carriage Return« und »Line Feed«) abgeschlossen werden.

HTTP-Request

Ein HTTP-Request ist schematisch wie folgt aufgebaut:

```
Request-Zeile
[Message-Header CRLF]*
CRLF
[Message-Body]
```

Die Request-Zeile gibt die Request-Methode, einen URI sowie die Protokollversion an. Anschließend können einige Message-Header-Zeilen mit Request-Parametern und Clientinformationen folgen. Eine Leerzeile beendet den HTTP-Request oder trennt den Message-Header vom Message-Body, falls ein solcher folgt.

HTTP-Response

Ein HTTP-Response hat einen fast identischen Aufbau:

```
Status-Zeile
[Message-Header CRLF]*
CRLF
[Message-Body]
```

Die Statuszeile besteht aus der Protokollversion und einem Erfolgs- oder Fehlercode. Ihr folgen Message-Header-Zeilen, die Server- und Metainformationen enthalten. Auch hier beendet eine Leerzeile den HTTP-Response oder trennt gegebenenfalls Message-Header und Message-Body voneinander.

Mit einem HTTP-Sniffer wie zum Beispiel *http://web-sniffer.net* lassen sich Befehle und Daten, die bei einer Verbindung ausgetauscht werden, sehr einfach einsehen. Ein Aufruf der Adresse *http://www.example.com/* im Browser könnte folgenden HTTP-Request an den Server senden:

```
GET / HTTP/1.1[CRLF]
Host: www.example.com[CRLF]
Connection: close[CRLF]
Accept-Encoding: gzip[CRLF]
Accept: text/xml, application/xml, application/xhtml+xml,
  text/html;q=0.9, text/plain;q=0.8,
  image/png,*/*;q=0.5[CRLF]
Accept-Language: en-us,en;q=0.5[CRLF]
Accept-Charset: ISO-8859-1,utf-8;q=0.7,*;q=0.7[CRLF]
User-Agent: Mozilla/5.0 (Windows; U; Windows NT 5.1; en-US;
  rv:1.7.1) Gecko/20040707 [CRLF]
[CRLF]
```

Listing 2.3: Beispiel eines HTTP-Requests

In der Request-Zeile fordert der Client eine Ressource mit der Methode `GET` an und gibt zudem an, HTTP der Version 1.1 zu verstehen. In diesem Fall adressiert der URI kein konkretes Dokument, sondern den *Document Root*. Es ist abhängig von den Servereinstellungen, welche Ressource zurückgeliefert wird, in vielen Fällen ist es die Datei `index.html`. Als Nächstes folgen die Message-Header-Zeilen. Das Feld `Host` gibt den Domainnamen des Webservers und gegebenenfalls den Port an. Mit dem Feld `Connection` kann der Sender Optionen beschreiben, die nur für eine Verbindung gelten. Hier soll die Verbindung nach erfolgtem Response geschlossen werden. `Accept-Encoding` beschränkt die kodierbaren Inhalte, `Accept` spezifiziert, über `q`-Werte (*quality values*) relativ gewichtet, die akzeptierbaren **MIME-Typen**, `Accept-Language` die bevorzugte Sprache, `Accept-Charset` den akzeptierbaren **Zeichensatz**. Das Feld `User-Agent` übermittelt Angaben über den Browser, in diesem Beispiel ein **Mozilla** der Version 1.7.1 unter Windows.

Der HTTP-Response des Servers auf diesen Request könnte wie folgt aussehen:

```
HTTP/1.1 200 OK
Date: Wed, 25 Aug 2004 18:18:43 GMT [CRLF]
Server: Apache/1.3.27 (Unix) (Red-Hat/Linux) [CRLF]
Last-Modified: Wed, 08 Jan 2003 23:11:55 GMT [CRLF]
ETag: "3f80f-1b6-3e1cb03b" [CRLF]
Accept-Ranges: bytes [CRLF]
Content-Length: 438 [CRLF]
Connection: close [CRLF]
Content-Type: text/html [CRLF]
[CRLF]
<HTML>[CRLF]
<HEAD>[CRLF]
  <TITLE>Example Web Page</TITLE>[CRLF]
</HEAD>[CRLF]
```

```
<body>[CRLF]
<p>You have reached this web page by typing
  &quot;example.com&quot;,[CRLF]
  &quot;example.net&quot;,[CRLF]
  or &quot;example.com&quot; into your web
  browser.</p>[CRLF]
<p>These domain names are reserved for use in documentation
  and are not available [CRLF]
  for registration. See <a href="http://www.rfc-
  editor.org/rfc/rfc2606.txt">RFC [CRLF]
  2606</a>, Section 3.</p>[CRLF]
</BODY>[CRLF]
</HTML>[CRLF]
[CRLF]
```

Listing 2.4: Beispiel für einen HTTP-Response

Die Statuszeile gibt hier den Erfolgscode `200 OK` an, es ist also kein Fehler aufgetreten. Die Bedeutung der HTTP-Header-Felder erklärt sich von selbst oder ist – wenn Sie tiefer in das Thema einsteigen möchten – in RFC 2616 nachzulesen. Dort finden Sie in Abschnitt 10 auch sämtliche Statuscode-Definitionen aufgeführt. Mit einer Leerzeile vom HTTP-Header abgetrennt, folgt der 438 Byte lange Content-Body, wie im Feld `Content-Length` bereits angekündigt wurde. In diesem Fall handelt es sich um das zurückgelieferte HTML-Dokument, das der Browser nach Aufruf der Adresse *http://www.example.com/* anzeigen würde.

Angabe der Zeichenkodierung

Was leider fehlt, ist eine Angabe der Zeichenkodierung. Diese Information ist die »schwache Stelle« der Trennung von Ressource und Transfer, weil sie nicht technisch sicher in der Ressource selbst untergebracht werden kann. Das Feld `Content-Type` des Request-Headers der W3C-Homepage sieht besser aus, da es eine Angabe darüber enthält, welcher Zeichenkodierung das Dokument entspricht:

```
Content-Type: text/html; charset=utf-8
```

Die richtige Angabe der Kodierung verdient Ihre besondere Aufmerksamkeit und gilt als Kriterium Ihrer Kompetenz. Mehr dazu in Kapitel 2.10.

2.5.3 MIME-Typen

DEFINITION

Multimedia Internet Message Extensions (MIME) *legen die Struktur und den Aufbau von E-Mails fest, werden im Internet aber ferner dafür verwendet, die Art der zu übertragenden Daten bei Client/Server-Verbindungen festzulegen, unter anderem auch bei HTTP.*

Wie Sie bereits gesehen haben, tauschen Client und Server bei einem HTTP-Dialog Informationen über die Art der angefragten Ressource aus. Der Client sendet mittels `Accept-Header` eine Liste aller MIME-Typen an den Server, die er (bevorzugt) verarbeiten kann. Der Server wiederum versieht seine Antwort mit einem Content-Type-Header, in dem die Art der gesendeten Ressource und gegebenenfalls, bei textuellen Ressourcen, die verwendete Zeichenkodierung notiert sind.

Verarbeitung einer Ressource

Im Allgemeinen akzeptieren Browser Ressourcen jedes MIME-Typs. Was sie dann mit ihnen anstellen, ist abhängig von der Server-, Browser- und Systemkonfiguration, also ob sie die Ressource im Browserfenster darstellen, zum Herunterladen anbieten oder an ein anderes Programm übergeben, das diese Ressource dann anzeigt oder anderweitig verarbeitet.

- Jeder Browser kann eine unterschiedliche Menge an Dateitypen standardmäßig anzeigen; dazu gehören neben (X)HTML-Dokumenten in den meisten Fällen Textdateien, CSS-Stylesheets und Grafikformate wie **GIF**, **JPEG** oder **PNG**. Andere Dateitypen benötigen ein **Plug-in**, beispielsweise **Flash** oder **SVG**.
- Wenn der Browser die Zielressource nicht selbst verarbeiten kann, ihm aber ein Programm bekannt ist, das entsprechende Dateien verarbeitet, kann der Start dieses Programms veranlasst und die Datei an das Programm übergeben werden. Gegebenenfalls wird das Anzeigefenster des Fremdprogramms in das Browserfenster eingebettet. Dieses Verhalten ist oft bei Musik- oder Videodateien zu beobachten.
- Kann der Browser mit dem Dateityp der Zielressource nichts anfangen und ist ihm auch kein Programm bekannt, das die Datei verarbeiten kann, wird ein Dialog angeboten, in dem entschieden wird, was mit der Datei geschehen soll; auf diese Weise kann der Nutzer die Datei herunterladen. Ein solcher Dialog erscheint in den meisten Fällen bei gepackten Dateien, beispielsweise ZIP oder RAR.

Aufbau

Ein MIME-Typ wird in zwei Teilen notiert, die durch einen Schrägstrich (`/`) voneinander getrennt sind. Zunächst erfolgt die Angabe eines Medientyps, anschließend wird der Untertyp angeführt. Auf beiden Seiten können Sie einen Asterisk (`*`) als Platzhalter notieren; so steht `text/*` für alle Textdateien, `*/*` für Dateien eines beliebigen MIME-Typs.

Mittlerweile gibt es so viele MIME-Typen wie Sand am Meer. Eine sehr umfangreiche und häufig aktualisierte Liste finden Sie auf der Seite *MIME Media Types* der **IANA**[15], die für Webautoren wichtigsten sind in Tabelle 2.6 aufgelistet.

15 *http://www.iana.org/assignments/media-types/*

MIME-Typ	Dateinamens-erweiterung(en)	Erläuterung
`application/pdf`	pdf	**PDF**-Dateien
`application/xhtml+xml`	htm, html, xhtml, xml	XHTML-Dateien
`application/xml und text/xml`	xml	XML-Dateien
`application/x-httpd-php`	php, html	**PHP**-Dateien
`application/zip`	zip	ZIP-Dateien
`image/gif`	gif	GIF-Grafikdateien
`image/jpeg`	jpeg, jpg	JPEG-Grafikdateien
`image/png`	png	PNG-Grafikdateien
`multipart/form-data`		Daten aus einem Webformular mit Dateiupload
`text/css`	css	CSS-Dateien
`text/html`	htm, html, xhtml	(X)HTML-Dateien
`text/plain`	txt	Standardtextdateien
`video/mpeg`	mpeg, mpg	MPEG-Videodateien

Tabelle 2.6: MIME-Typen in der Webentwicklung

2.6 Sprachen des World Wide Webs

2.6.1 HTML

DEFINITION

*Die **Hypertext Markup Language** (HTML) bildet zusammen mit den Standards Hypertext Transfer Protocol (HTTP) und Uniform Resource Identifier (URI) die fundamentalen Mechanismen des World Wide Webs. HTML vereint semantische Auszeichnung von Inhalten, und Hypertext und ermöglicht Webautoren, unter anderem*

- *Dokumente mit Überschriften, Text, Tabellen, Auflistungen, Hyperlinks zu anderen Dokumenten und Ressourcen, Grafiken sowie anderen Arten von Auszeichnungen online zu veröffentlichen,*
- *Formulare zur Interaktion mit Benutzern zu entwerfen, über die diesen Suchfunktionen, Kontakt- und Bestellmöglichkeiten und vieles mehr angeboten werden können, sowie*
- *Multimedia-Elemente wie Videos, Musik und Animationen sowie Applikationen direkt in ihre Dokumente einzufügen.*

Entwicklung

SGML

Als Tim Berners-Lee im Jahr 1990 am CERN-Institut an der Entwicklung einer einfachen Hypertextsprache arbeitete, die in der Lage sein sollte, Basismenüs für die Hypertextnavigation und einfache Dokumente wie Hilfedateien, Protokolle oder technische Dokumentationen zur Verfügung zu stellen, landete er – vor allem aus politischen Gründen – bei SGML[16].

Berners-Lee verfolgte die Idee einer Sprache, die die Struktur eines Hypertextdokuments beschreiben sollte, aber keine Details von dessen Darstellung, um eine optimale Kompatibilität zu allen verschiedenen Bildschirmarten und Papiergrößen zu erzielen, die am CERN zu dieser Zeit verwendet wurden. Aber auf welche Art und Weise sollte die Auszeichnung semantischer Strukturen notiert werden? SGML galt in der damaligen Hypertextgemeinde als der einzige potenzielle Dokumentenstandard. Um sich die breite Unterstützung der Hypertext- und SGML-Anhänger für sein World Wide Web zu sichern, entwickelte Berners-Lee HTML so, dass es wie ein Mitglied der auf SGML basierenden Familie von Auszeichnungssprachen aussah – und es schließlich auch wurde. Das Web benötigte in seinen Anfängen Unterstützung von jeder Seite, die in dessen Einwicklung einbezogen werden konnte, und die SGML-Gemeinde bot in vielerlei Hinsicht eine wertvolle Hilfestellung. Darüber hinaus wurde auf den IBM-Rechnern am CERN ein besonderer Satz von **Tags** verwendet, die in spitze Klammern gesetzt wurden und der Notation von SGML-Dokumenttypen in dieser Hinsicht ähnelten. So sollte CERN-Mitarbeitern die Verwendung von HTML schmackhaft gemacht werden.

Dass HTML heute die Gestalt hat, in der es sich präsentiert, ist somit auf einige ganz pragmatische Entscheidungen zurückzuführen; es hätte auch vollkommen anders aussehen können.

HTML 2.0

Bereits in den ersten Jahren nach seiner Veröffentlichung wurde das Web und mit ihm HTML vor allem aufgrund seiner Einfachheit und seiner Möglichkeiten sehr schnell äußerst populär. Verschiedene Institutionen, vor allem natürlich Browserhersteller, erweiterten HTML auf vielfältige Art und Weise. Da das Web als gemeinsame Basis zur Kommunikation entworfen wurde, ist es allerdings darauf angewiesen, dass Autoren von Webseiten und Entwickler von Browsern und Editoren dieselben Konventionen für HTML einhalten. Andernfalls bestünde die Gefahr, dass das Web auf eine inkompatible Welt bestehend aus **proprietären** Formaten zusteuert und sich letzten Endes das kommerzielle Potenzial des Webs und die plattformübergreifende Zugänglichkeit seiner Inhalte für alle Beteiligten verringert. Diese Erkenntnis führte dazu,

16 vgl. ***[Berners-Lee 1999]***

dass HTML im Jahr 1995 unter der Leitung der **IETF** erstmalig als verbindliche Spezifikation standardisiert und als RFC 1866 »Hypertext Markup Language – 2.0« ***[RFC-1866 1995]*** veröffentlicht wurde.

HTML 3.2 HTML 2.0 umfasste nicht alle Erweiterungen, weder **Frames** oder Tabellen noch das Einbinden von JavaScript. Die meisten Webautoren setzten sich deshalb über den Standard hinweg, um die Möglichkeiten, die damals aktuelle Browser (vor allem der zu dieser Zeit marktführende **Netscape Navigator**) boten, ausschöpfen zu können. In dieser Zeit begann eine intensive Zusammenarbeit zwischen dem akademisch orientierten W3C und kommerziell orientierten Softwareherstellern. Früchte dieser Zusammenarbeit waren HTML+ und HTML 3.0, die umfangreichere Versionen von HTML vorschlugen, jedoch zu wenig Konsens fanden, um den Status einer Empfehlung zu erreichen. Im Jahr 1997 schließlich wurde die offizielle Version HTML 3.2 ***[REC-HTML32 1997]*** verabschiedet.

HTML 4.0 HTML 4.0 ***[REC-HTML40 1998]*** folgte im Jahr 1998 und erweiterte die Sprache um Mechanismen für Stylesheets und Scripting, Frames, eingebundene Objekte, Internationalisierung und bessere Zugänglichkeit für eingeschränkte Benutzer. Berners-Lee kommentierte den Versionssprung mit folgenden Worten: »Die Entwicklung von HTML nähert sich ihrem Ende. Jetzt ist es eher notwendig, einige lose Enden zusammenzufügen.«

HTML 4.01 Ende 1999 wurde HTML 4.01 ***[REC-HTML401 1999]*** verabschiedet, das einige Fehler aus HTML 4.0 beseitigte und diese Version ersetzte. HTML 4.01 ist noch heute die »Lingua Franca« des World Wide Webs und eine der populärsten Sprachen in der Welt der Nullen und Einsen, trotz aller weiteren Entwicklungen.

2.6.2 XHTML

DEFINITION

*Die **Extensible Hypertext Markup Language** (*XHTML*) ist eine Familie von XML-Dokumenttypen, die den SGML-Dokumenttyp HTML 4.01 in XML »übersetzen«. Einfach ausgedrückt, ist XHTML eine auf XML basierende Auszeichnungssprache, die wie HTML aussieht und funktioniert. Allerdings entspricht XHTML den strengeren Syntaxregeln von XML. XHTML 1.0 entstand im Januar 2000 und enthält mit sehr wenigen Unterschieden den gleichen Sprachumfang wie HTML 4.01.*

XHTML 1.0 Im August 2002 erschien XHTML 1.0 in der zweiten Ausgabe ***[REC-XHTML10 2002]***, die einige Fehler korrigiert und die Errata in die erste Ausgabe integriert hat.

Die Spezifikation beschreibt XHTML 1.0 als ersten Dokumenttyp einer Familie aktueller und zukünftiger XHTML-Dokumenttypen und -Module, die sowohl XML-konform sind und somit mit speziellen XML-Benutzerprogrammen angezeigt und verarbeitet werden können als auch – wenn einige einfache Richtlinien befolgt werden – im HTML 4-konformen Benutzerprogrammen funktionieren, also in jedem Browser angezeigt werden können.

Modularisierung von XHTML

Im April 2001 wurde die W3C-Empfehlung »Modularisierung von XHTML« ***[REC-ModXHTML 2001]*** veröffentlicht. Darin werden die in XHTML 1.0 verfügbaren Elementtypen in eine Sammlung von abstrakten Modulen aufgeteilt, die bestimmte Funktionsbereiche abbilden. Module können miteinander und mit anderen Modulen kombiniert werden, sodass darauf basierend sowohl erweiterte als auch abgespeckte Dokumenttypen entworfen werden können.

XHTML 1.1

XHTML 1.1 ***[REC-XHTML11 2001]*** wurde im Mai 2001 basierend auf der Modularisierung von XHTML als modulbasierte Sprache entworfen. War XHTML 1.0 lediglich eine Migration des aus HTML 4.01 bekannten Sprachumfangs in die Welt von XML, so stellte XHTML 1.1 nun einen vollkommen von Altlasten bereinigten Neuentwurf der Sprache dar, der nicht mehr kompatibel zu HTML ist. Während XHTML 1.0 nahezu dasselbe Vokabular umfasst wie HTML 4.01 und daher ebenfalls mit dem für HTML-Dokumente vorgesehenen MIME-Typ `text/html` ausgeliefert werden kann, sollten XHTML 1.1-Dokumente mit dem XHTML-MIME-Typ `application/xhtml+xml` ausgeliefert werden. Der Internet Explorer und zahlreiche andere Benutzerprogramme können diesen MIME-Typ jedoch nicht verarbeiten; Ihre Webseiten würden in diesen Browsern im Allgemeinen nicht angezeigt, sondern zum Download angeboten werden. Daher und auch aus anderen Gründen ist XHTML 1.1 noch nicht praxistauglich. Vielleicht wird es mir erlaubt sein, in einer dritten Auflage dieser Einführung über XHTML 1.1 zu berichten.

2.6.3 Warum sollten Webautoren heute auf XHTML setzen?

SGML

Sie haben bereits gelesen, dass SGML eine sehr komplexe Sprache ist, die es schwer, wenn nicht nahezu unmöglich macht, Benutzerprogramme zu schreiben, die mit allen Facetten von SGML umgehen können. HTML als SGML-Dokumenttyp hat viel dieser Komplexität übernommen; beispielsweise Shorttags: Konstrukte, die falsch zu sein scheinen (wie `<p/Absatz/` statt `<p>Absatz</p>`, um ein wenig vorzugreifen), aber syntaktisch richtig sind und dennoch von keinem modernen Browser richtig verstanden werden. Webautoren verwenden diese Konstrukte zumeist unabsichtlich und wundern sich dann über eine falsche Darstellung, die aus »Fehlern« resultiert, die sie auch mithilfe von Werkzeugen wie Validatoren nicht aufspüren können, weil es sich rein syntak-

tisch nicht um Fehler handelt. XHTML als XML-Dokumenttyp gehorcht viel strengeren Regeln als HTML; die Syntax ist eindeutiger. Fehler dieser Art können nicht passieren.

Kontextabhängigkeit von HTML

Christoph Schneegans zeigt in seinem Artikel »XHTML-Einmaleins« ***[Schneegans]*** viele weitere Beispiele, die den Umgang mit HTML erschweren. Zum einen führt er aus, dass HTML in hohem Maße kontextabhängig ist:

- `<a href=foo.html>` ist gültiges HTML, `<a href=../foo.html>` nicht.
- `<p>x < y</p>` dürfen Sie schreiben, `<p>x<y</p>` hingegen nicht.
- `<p>&uuml</p>` ist erlaubt, `<p>T&uumlr</p>` wiederum verboten.

Beliebigkeit von HTML

Darüber hinaus bemängelt Schneegans die Beliebigkeit von HTML.

- Sie dürfen sowohl `<a href=foo.html>` als auch `<a href="foo.html">` schreiben.
- `<div><p></div>` ist korrekt, `<div><p></p></div>` aber auch.
- Die Schreibweise `<p>&auml</p>` ist ebenso gültig wie `<p>ä</p>`.

Diese Mehrdeutigkeiten, die es schwierig machen können, HTML-Dokumente zu lesen und darin Fehler zu finden, vermeidet XHTML durch seine strengere und dadurch einfachere und intuitivere Syntax, die es Anfängern erleichtert, XHTML zu lernen, da sie nicht auf die zahlreichen kleinen Feinheiten und Ausnahmen achten müssen.

Ich möchte nicht verschweigen, dass viele XHTML-Kritiker noch immer mit großer Beharrlichkeit Argumente gegen XHTML vorbringen, die gerade in den Ohren von Anfängern durchaus schlüssig klingen. Ich kann Ihnen versichern, dass kaum eines dieser Argumente stichhaltig und keines praxisrelevant ist. Einer dieser Kritiker ist Ian Hickson, dessen sehr bekannter Artikel »Sending XHTML as text/html Considered Harmful« (*www.hixie.ch/advocacy/xhtml*) jedoch hauptsächlich dadurch besticht, dass er vor allem einseitig und irreführend ist.

Gründe für XHTML

Wie Sie bei der Lektüre dieser Einführung feststellen werden, brauchen Sie vor der Verwendung von XHTML 1.0 nicht zurückzuschrecken. Ganz im Gegenteil: Es gibt handfeste Gründe für den Wechsel von HTML zu XHTML[17]:

- XHTML ist der aktuelle Standard für Webauszeichnungssprachen.
- XHTML ist so entworfen, dass es gut mit anderen, auf XML basierenden Auszeichnungssprachen, Applikationen und Protokollen zusammenarbeitet.
- XHTML ist konsistenter als HTML und somit sowohl leichter zu erlernen als auch einfacher zu verarbeiten.

17 siehe ***[Zeldman 2007]***

- XML ist sehr leistungsfähig und mittlerweile praktisch überall anzutreffen, doch leider können die meisten aktuellen Browser mit XML-Technologien noch nicht vollständig umgehen, und es ist daher nicht möglich, Browsern einfach XML vorzusetzen in der Hoffnung, sie würden schon das anzeigen, was man von ihnen erwartet. XHTML 1.0 stellt eine Art Brücke zwischen HTML und XML dar. Es ist leichter, von XHTML 1.0 auf zukünftige XHTML-Versionen zu wechseln als von HTML.
- XHTML vereinfacht Webautoren die Arbeit, da es klarere und einfachere Regeln gibt. Darüber hinaus gibt es zahlreiche Validatoren (siehe Kapitel 3.2 »Validatoren«), die XHTML-Dokumente wesentlich genauer analysieren können als HTML-Dokumente. Dadurch können Webautoren Fehler besser finden und beseitigen.

Wenn Sie Ihre Webseiten in XHTML verfassen, stehen Ihnen viel effizientere Werkzeuge zur Verfügung als bei HTML, beispielsweise Schema-Validatoren oder die zahlreichen XML-Techniken und -Sprachen, die in den letzten Jahren wie Pilze aus dem Boden schießen. So ist es mithilfe von **XSL** verhältnismäßig einfach möglich, Inhalte in andere Ausgabeformate zu konvertieren, beispielsweise per **XSL-FO** in PDF-Dokumente für die Printausgabe.

Wichtig ist allerdings, dass Sie bei der Verwendung von XHTML die **HTML-Kompatibilitätsrichtlinien** beachten (siehe Kapitel 4.4).

2.6.4 Ausblick: XHTML 2.0, (X)HTML 5

Die letzten wesentlichen inhaltlichen Veränderungen der Sprachen HTML und XHTML sind mittlerweile etwa zehn Jahre alt. Seit dieser Zeit sind nahezu keine neuen Elemente und Attribute im Rahmen eines neuen Standards hinzugekommen. Allerdings geschieht seit einiger Zeit hinter den Kulissen eine ganze Menge.

XHTML 2.0

Seit 2002 arbeitet das W3C an XHTML 2.0 ***[WD-XHTML20 2006]***, das derzeit noch als Arbeitsentwurf vorliegt und diesen Status auch so bald nicht verlassen wird. Dabei handelt es sich um eine Neuentwicklung von XHTML, die grundlegende Veränderungen im Vergleich zur aktuellen Version vorsieht und mit dieser (und somit auch mit HTML) nicht mehr kompatibel ist. Wichtige Neuerungen sind

- die vereinfachte, uneingeschränkte Einbindung von Hyperlinks,
- die einfachere Integration anderer Medien wie Grafiken und Videos,
- die verbesserten Möglichkeiten, die Zugänglichkeit von Webseiten zu gewährleisten,
- das ausgereiftere Modell zur Notation von Metadaten sowie
- zahlreiche neue Elementtypen, darunter Navigationslisten, neue Strukturen für Überschriften und Verbesserung für Codebeispiele.

Bisherige Kernfunktionen von (X)HTML werden in XHTML 2.0 in andere XML-Dokumenttypen ausgelagert, beispielsweise Formulare (**XForms**) und Frames (XFrames).

XHTML 2.0 verspricht viele nützliche Neuerungen und kann einen echten Fortschritt gegenüber den aktuellen Technologien darstellen. Dieser Fortschritt wird hingegen mit hoher Komplexität erkauft. Zu den aufgeführten XML-Sprachen kommen noch einige weitere hinzu; so müssen Webentwickler, die mit XForms arbeiten wollen, auch **XPath** und XML Schema verstehen. Im Vergleich zu heute muss ein guter Webentwickler in Zukunft viel mehr Techniken beherrschen, was den einen oder anderen semiprofessionellen Entwickler überfordern könnte. Auch auf Hersteller von Autorenwerkzeugen (beispielsweise Editoren) oder Browsern kommt eine ganze Menge Arbeit zu – implementiert sind die meisten Sprachen derzeit noch nicht einmal ansatzweise.

WHATWG

Einen deutlich praxisnäheren Ansatz verfolgt die WHATWG mit der Entwicklung von (X)HTML 5.

DEFINITION

*Die **WHATWG** (*Web Hypertext Application Technology Working Group*) ist eine Arbeitsgruppe, die 2004 gegründet wurde und sich zum Ziel gesetzt hat, neue Webtechnologien durch die Erweiterung bereits bestehender Technologien zu entwickeln und auf die tatsächlichen Bedürfnisse der Webgemeinde auszurichten. Die Gründung der WHATWG erfolgte als Reaktion auf die schleppende Entwicklung von Webstandards durch das W3C. Beteiligte Unternehmen sind unter anderem die Mozilla Foundation, Opera Software ASA und Apple Inc.*

Die WHATWG hat *nicht* vor, das W3C zu ersetzen oder Standardisierungsprozesse am W3C vorbei zu etablieren. Das Gegenteil ist der Fall: Mitglieder des WHATWG arbeiten zum Teil in W3C-Arbeitsgruppen mit und umgekehrt. In seinem Selbstverständnis ebnet das WHATWG dem W3C sozusagen den Weg: Das letzte Wort hat stets das W3C, dem alle Standards zur Zustimmung oder Nachbesserung vorgelegt werden.

Die bekannteste Entwicklung der WHATWG ist (X)HTML 5 ***[WD-XHTML5 2008]***.

(X)HTML 5

(X)HTML 5, zunächst veröffentlicht als »Web Applications 1.0«, ist ein Arbeitsentwurf, der versucht, zahlreiche Standards wie HTML 4, XHTML 1.0 oder DOM unter einen Hut zu bringen und abwärtskompatibel zu erweitern. Ziel ist es, die Unterschiede zwischen den Sprachen zu verringern und eine bessere Zusammenarbeit der Komponenten zu erreichen.

In einer Abstimmung im Mai 2007 haben die Mitglieder der HTML-Arbeitsgruppe entschieden, die Spezifikation von (X)HTML 5 als Startpunkt zur Diskussion und Weiterentwicklung von (X)HTML zu verwenden. Am 22. Januar 2008 hat das W3C den ersten Working Draft (Arbeitsentwurf) von (X)HTML 5 veröffentlicht. Damit wird (X)HTML 5 unter dem Dach des W3C weitergeführt.

Die derzeitigen Entwicklungen sind spannend mitzuverfolgen, aber seien Sie gewiss, es wird doch mehrere Jahre dauern, bis XHTML 2.0 und/oder (X)HTML 5 wirklich praxisrelevant werden. Die WHATWG selbst geht davon aus, dass (X)HTML 5 erst 2022 oder noch später als W3C-Empfehlung fertiggestellt sein wird! Bis dahin werden Sie hoffentlich noch viele Websites mit XHTML 1.0 erstellt haben.

2.6.5 CSS

DEFINITION

Cascading Style Sheets *(CSS) ist eine Formatierungssprache, die es Webautoren gestattet, Formatierungen von strukturierten Dokumenten durchzuführen. CSS wird hauptsächlich verwendet, um HTML- und XHTML-Dokumente für die Ausgabe in unterschiedlichen Medien visuell aufzubereiten, beispielsweise für Computerbildschirme, Drucker, akustische Geräte oder Handhelds. In Kombination mit semantisch reichem (X)HTML erleichtert CSS die Erstellung und Wartung von Webseiten, die weniger Bandbreiten erfordern, zugänglicher sind und dadurch eine größere Klientel erreichen.*

Entwicklung

Bereits 1993, also noch vor der Veröffentlichung des ersten HTML-Standards 2.0 durch die IETF, gab es erste Vorschläge für Sprachen, die die zentrale Steuerung der Formatierung von Webdokumenten gestatten und dabei sowohl vom Webautor wie auch vom Nutzer selbst kontrolliert werden können. Dahinter liegt der Gedanke der Trennung des Inhalts von der Präsentation, also der Separation von Markup und Styling. Darauf gehe ich im CSS-Kapitel genauer ein.

Zu den ersten Entwicklern, die den Entwurf einer Stylesheetsprache zur Diskussion vorlegten, gehörte Robert Raisch, damals bei O'Reilly & Associates Inc. angestellt. Er beschrieb seinen Vorschlag *WWW HTML Style Guide Recommendations* (*www.w3.org/Style/History/www.eit.com/www.lists/www-talk.1993q2/0445*) mit folgenden Worten:

> *»There is a need within the WWW to be able to specify rendering specific information along with the content tagging in a WWW document. It is not appropriate to use HTML for this purpose, since one of the first principles of HTML is to encode objects within a document, not how they might be rendered in a particular environment.«*

Die Syntax hatte Raisch extra für seine Stylesheetsprache entworfen. Wie später CSS verfügte sie über Selektoren, Eigenschaften und Werte. Sie war zwar sehr kompakt, aber für Menschen zu schwierig zu lesen, um sich in dieser Form durchzusetzen.

Einen weiteren Vorschlag für eine Stylesheetsprache postete im Oktober 1993 Pei-Yuan Wei, Student an der Universität von Berkeley, in die www-talk-Mailingliste (*lists.w3.org/Archives/Public/www-talk*) des W3C. Wei hatte soeben ViolaWWW entwickelt, einen der ersten Webbrowser, und seine Stylesheetsprache für ViolaWWW bereits implementiert. Auch diese Syntax basierte darauf, bestimmte HTML-Elementtypen zu selektieren und einer Menge von Eigenschaften bestimmte Werte zuzuweisen. Anders als bei Raischs Entwurf konnte man bei Weis Syntax Zuweisungsblöcke ineinander verschachteln und so Wertzuweisungen für in HTML-Dokumenten verschachtelte Elemente vornehmen, also beispielsweise Zuweisungen für Elemente vom Typ `E` innerhalb von Elementen des Typs `F`. Es zeigte sich jedoch, dass diese Form der Syntax die Komplexität der Stylesheets erhöhte und somit die Lesbarkeit für Autoren erschwerte.

CHSS

1994 näherte sich die Entwicklung dem heutigen CSS durch die Veröffentlichung von *Cascading HTML Style Sheets* (CHSS) durch Håkon Wium Lie, damals wissenschaftlicher Angestellter am CERN und seit 1999 CTO beim Browserhersteller Opera. Viele heute bekannte Konzepte wurden mit dieser Sprache eingeführt, unter anderem die Spezifität von Selektoren, Regeln für unterschiedliche Medientypen und **Kaskade**, worüber Lie schrieb:

> *»The proposed scheme supplies the browser with an ordered list (cascade) of style sheets. The user supplies the initial sheet which may request total control of the presentation, but – more likely – hands most of the influence over to the style sheets referenced in the incoming document.«*

Andere Konzepte wie Umgebungsvariablen oder die Möglichkeit, Berechnungen durchzuführen, wurden nicht oder noch nicht übernommen, auch nicht die noch relativ unübersichtliche Syntax.

Stream-based Style Sheets

Zu dieser Zeit arbeitete der Mathematiker Bert Bos, heute Leiter der CSS-Arbeitsgruppe des W3C, an seinem eigenen Webbrowser »Argo« und verwendete in diesem Rahmen seine eigene Stylesheetsprache *Stream-based Style Sheets*. Lie wurde auf Bos' Arbeit aufmerksam, und aufgrund ihrer ähnlichen Ideen und Inspirationen entschlossen sie sich, zusammenzuarbeiten und gemeinsam, basierend auf den jeweils entwickelten Konzepten, eine neue Sprache zu entwickeln: CSS. Ihre Entwicklung präsentierten sie 1994 auf der Konferenz »Mosaic and the Web« in Chicago, wo das W3C auf sie aufmerksam wurde und sie einlud, in Rahmen des Konsortiums an CSS weiterzuarbeiten. Im Dezember 1996 wurde dann schließlich CSS 1 als W3C-Empfehlung veröffentlicht.

CSS 1 erlaubte es Webautoren erstmals, in einer übersichtlichen und intuitiven Syntax Farben und Schrifteigenschaften zu definieren, Breiten und Höhen von Elementen zu bestimmen, Text zu formatieren und auszurichten sowie Abstände und Rahmen zu definieren und vieles mehr. Im Januar 1999 folgte eine leicht überarbeitete und fehlerbereinigte Version der CSS 1-Empfehlung ***[REC-CSS10 1999]***. CSS 1

Nicht einmal zwei Jahre nach CSS 1, im Mai 1998, erschien die nächste Version. CSS 2 ***[REC-CSS20 1998]*** basierte auf seinem Vorgänger, sodass CSS 1-Dokumente mit wenigen Ausnahmen gültige CSS 2-Dokumente sind. CSS 2 hatte etwa den vierfachen Umfang von CSS 1 und erweiterte den Standard unter anderem um die Unterstützung medienspezifischer Stylesheets, Positionierung von Elementen, Tabellenlayout, Internationalisierung sowie automatische Zähler und Nummerierung. CSS 2

Derzeit arbeitet die CSS-Arbeitsgruppe parallel an zwei neuen Versionen: CSS 2.1 und CSS 3.

Der aktuelle Arbeitsentwurf für CSS 2.1 ***[CR-CSS21 2006]*** erschien im Juli 2007 und stellt einen Sonderfall in der mittlerweile beachtlichen Reihe an W3C-Empfehlungen dar. Anders als gewöhnlich wird kein neuer Standard definiert, sondern es werden jene Teile aus der CSS 2-Empfehlung herangezogen, die sich als stabil und gebrauchstauglich erwiesen haben. Wo Implementierungen in modernen Browsern ganzheitlich vom Standard abweichen, wird der Standard angepasst; Eigenschaften und Werte, die überhaupt noch nicht implementiert wurden, werden als nicht ausgereift definiert und daher entfernt; sie tauchen gegebenenfalls erst in CSS 3 wieder auf. CSS 2.1

In der ersten Auflage dieses Buchs habe ich voller Optimismus geschrieben, dass mit der Veröffentlichung von CSS 2.1 als offizielle Empfehlung noch im Jahr 2006 gerechnet wird. Ich schreibe diese Zeilen nun im November 2008, und CSS 2.1 ist zwar kein Working Draft mehr, sondern mittlerweile eine Candidate Recommendation, jedoch noch lange keine fertige Empfehlung (siehe den Recommendation Process in Abschnitt 2.4.2). Realistisch betrachtet, ist es derzeit nicht möglich, einen Veröffentlichungstermin vorauszusagen. Dennoch ist auch diese Auflage auf dem Stand von CSS 2.1, da bis zur Verabschiedung nicht mehr mit größeren Änderungen zu rechnen ist und sich die meisten Browser ohnehin inzwischen eher CSS 2.1 als CSS 2 nähern.

Parallel zu CSS 2.1 arbeitet das W3C an CSS 3. CSS 3 teilt CSS 2 in eine ganze Reihe von Modulen auf, ähnlich wie XHTML durch die Empfehlung »Modularisierung von XHTML« aufgeteilt worden ist. Einige Module werden unverändert übernommen, beispielsweise die Syntax, die meisten jedoch wurden stark CSS 3

erweitert. Das Modul *Backgrounds and Borders*[18] wird es ermöglichen, Rahmen aus beliebigen Grafiken zu erzeugen, das *Multi-column layout module*[19] ist ein Modul für Mehrspaltigkeit, und es wird voraussichtlich auch Module für die Formatierung von Hypertextverhalten (Pop-up-Boxen, auf- und zuklappbare Listen oder Fußnoten) und mathematische Formeln geben. Eine Übersicht aller Module bietet die CSS 3-Roadmap[20]. Wann die ersten Module als Empfehlung veröffentlicht werden, ist noch nicht festgelegt. Zum Teil können Sie bereits heute CSS 3 einsetzen; an den entsprechenden Stellen weise ich Sie darauf hin. Aber im Allgemeinen ist die Browserunterstützung derzeit noch ziemlich gering – was natürlich nicht verwundern kann.

2.6.6 Spezifikationen, Empfehlungen und Arbeitsentwürfe

Tabelle 2.7 gibt einen Überblick über die in diesem Kapitel angesprochenen Sprachen und deren aktuelle Versionen. Gibt es eine deutsche Übersetzung, so wird diese ebenfalls aufgeführt.

Sprache	Version	URL
HTML	2.0	http://www.faqs.org/rfcs/rfc1866.html
	3.2	http://www.w3.org/TR/REC-html32
	4.01	http://www.w3.org/TR/html4 http://www.edition-w3c.de/TR/html4
XHTML	1.0	http://www.w3.org/TR/xhtml1/ http://www.edition-w3c.de/TR/xhtml1
	1.1	http://www.w3.org/TR/xhtml11 http://www.8ung.at/w3c-trans-de/xhtml11
	2.0	http://www.w3.org/TR/xhtml2
Modulbasiertes XHTML	1.1	http://www.w3.org/TR/xhtml-modularization http://www.edition-w3c.de/TR/xhtml-modularization
(X)HTML	5	http://www.w3.org/TR/html5/
CSS	1	http://www.w3.org/TR/REC-CSS1
	2	http://www.w3.org/TR/REC-CSS2 http://www.edition-w3c.de/TR/REC-CSS2
	2.1	http://www.w3.org/TR/CSS21
	3	http://www.w3.org/Style/CSS/current-work

Tabelle 2.7: Links zu aktuellen Spezifikationen, Empfehlungen und Arbeitsentwürfen

18 *http://www.w3.org/TR/css3-background*
19 *http://www.w3.org/TR/css3-multicol*
20 *http://www.w3.org/TR/css3-roadmap/#modlist*

2.7 Browser

In diesem Buch ist häufig von Benutzerprogrammen die Rede.

DEFINITION

*Ein **Benutzerprogramm** ist ein beliebiges Programm, das eine Ressource ausliest und weiterverarbeitet. Ein Benutzerprogramm kann eine Ressource anzeigen, verarbeiten, sie vorlesen, ihren Ausdruck veranlassen oder sie in ein anderes Format umwandeln.*

Über spezielle Benutzerprogramme, die Webseiten in Blindenschrift darstellen oder vorlesen (**Screenreader**) und von körperlich eingeschränkten Menschen im Rahmen assistiver Technologien verwendet werden, lesen Sie in Kapitel 2.9 »Barrierefreiheit«. Hier geht es vor allem um Webbrowser.

DEFINITION

*Ein **Webbrowser** (auch: Browser) ist ein Benutzerprogramm, das hauptsächlich zum Anzeigen von Webseiten über HTTP verwendet wird. Ursprünglich bezeichnet der englische Begriff to browse (stöbern, sich umsehen, schmökern) lediglich das Benutzen von Navigationselementen (Vor, Zurück, Index, ...) zum Lesen von (Hyper-)Texten am Computer. Moderne Browser verfügen über eine ganze Palette weiterer Funktionen, zum Beispiel FTP oder E-Mail. Die meisten Browser laufen auf einer grafischen Benutzeroberfläche und können mit Maus und/oder Tastatur bedient werden.*

2.7.1 Die Entwicklung bis zum Browserkrieg

Als Tim Berners-Lee im Jahr 1990 damit begann, den Code für sein WWW zu schreiben, entwickelte er neben

- HTTP, dem Protokoll, über das Computer über das Internet kommunizieren,
- URI, dem Schema, nach dem Dokumentadressen erstellt werden können, und
- HTML, der Sprache zur Auszeichnung von Webseiten,

auch den ersten Webbrowser. Er sah im Wesentlichen wie ein Textverarbeitungsprogramm aus und war eine Kombination aus Browser und Editor – HTML-Dokumente konnten mit demselben Programm bearbeitet und angezeigt werden. Leider lief das Programm nur auf NeXT-Computern, einer zu dieser Zeit zwar sehr innovativen, aber gering verbreiteten Plattform.

Erwise

Um das World Wide Web weiter zu verbreiten, war es notwendig, auch mit anderen Betriebssystemen daran teilhaben zu können. Dazu waren Browser für den PC, den Macintosh und für Unix-Systeme nötig. Berners-Lee und andere Perso-

nen, die am CERN und darüber hinaus an der Entwicklung des World Wide Webs mitwirkten, versuchten daher, Studenten die Entwicklung von Webbrowsern schmackhaft zu machen. Schließlich handelte es sich dabei – zumindest zu dieser Zeit – um verhältnismäßig überschaubare Projekte, die Studenten eine hervorragende Möglichkeit boten, ihre Programmierkenntnisse zu demonstrieren. 1992 nahmen verschiedene Studenten der Technischen Universität von Helsinki diese Herausforderung an und schrieben als Diplomarbeit einen Webbrowser namens Erwise, der im April erschien und für den Einsatz auf Unix-Computern unter der grafischen Benutzeroberfläche X-Windows programmiert war.

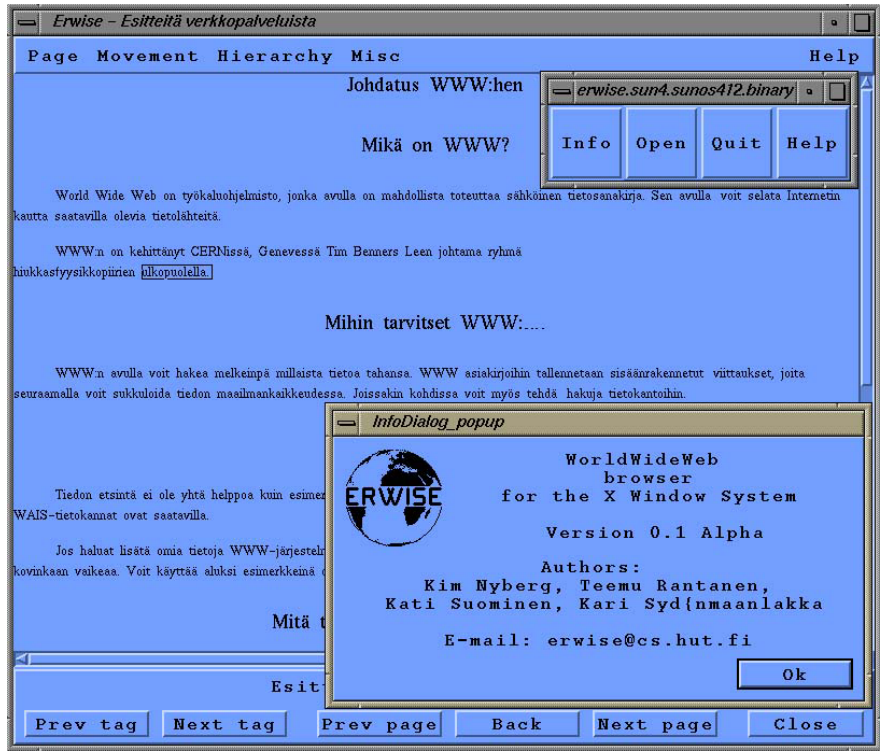


Abbildung 2.4: Erwise[21]

ViolaWWW

Nach und nach entstanden weitere Browser, unter anderem ViolaWWW, einer der ersten Browser, die HTML-Dokumente mit Grafiken darstellen, Animationen anzeigen und kleine, eingebettete Anwendungen aus dem Internet herunterladen und ausführen konnten. Allerdings beschränkten sich alle Projekte auf ein universitäres Umfeld; war die konkrete Aufgabe erfüllt, wurde der

21 Quelle: *http://www.nic.funet.fi/index/FUNET/history/internet/en/erwise.jpg*

Browser nicht weiterentwickelt. Darüber hinaus war die Installation aller Browser so schwierig, dass sie sich in der Breite nicht durchzusetzen vermochten und daher nicht viel mehr blieben als vielversprechende Studien.

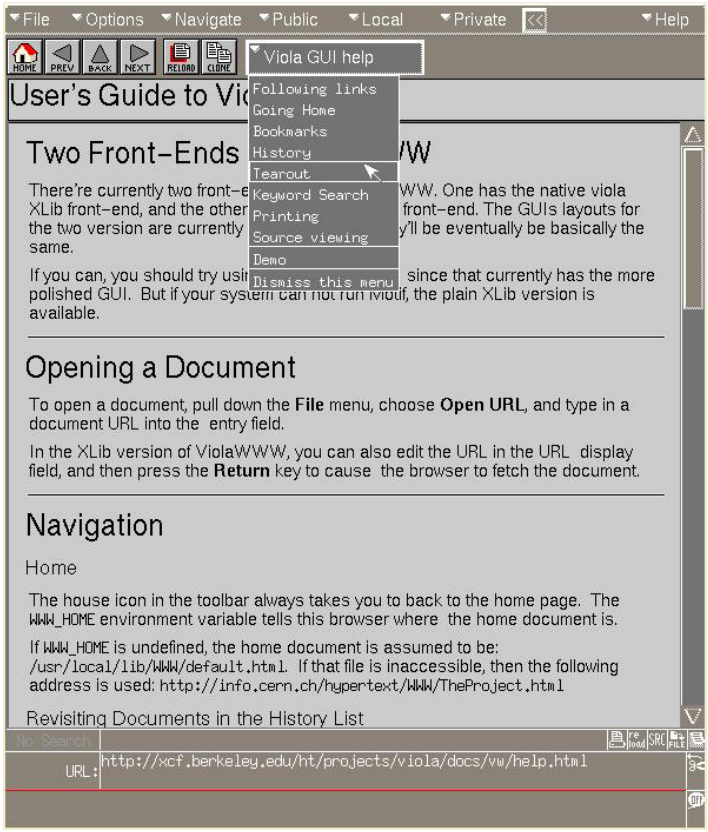


Abbildung 2.5: ViolaWWW[22]

Mosaic

1993 beschlossen Eric Bina, ein Mitarbeiter am *National Center for Supercomputing Applications* (**NCSA**) an der Universität von Illinois, und Marc Andreessen, ein Student, einen Browser für X-Windows zu erstellen. Andreessens Arbeit wurde durch zahlreiche lebhafte Diskussionen in den Newsgroups, in denen er fast ständig präsent war, angetrieben und beeinflusst. Schnell implementierte er gewünschte Funktionen und gab reihenweise Testversionen seines Browsers **Mosaic** heraus, damit andere diese ausprobieren konnten. Im

22 Quelle: *http://www.xcf.berkeley.edu/~wei/viola/violaHome.html*

Gegensatz zu anderen Browserentwicklern vor ihm ging es Andreessen nicht darum, ein akademisch hochwertiges Produkt zu entwickeln, sondern er wollte einen Browser schreiben, der von möglichst vielen Menschen verwendet werden kann. Dieser praktisch orientierte Ansatz unterschied sich stark von der akademisch beeinflussten Arbeit anderer Entwickler und war genau das, was das Web zu dieser Zeit benötigte. Das Endergebnis nannte er Mosaic – der erste Browser, der außer Text auch Grafiken auf einer Seite anzeigen konnte, ohne dass man diese extra nachladen musste.

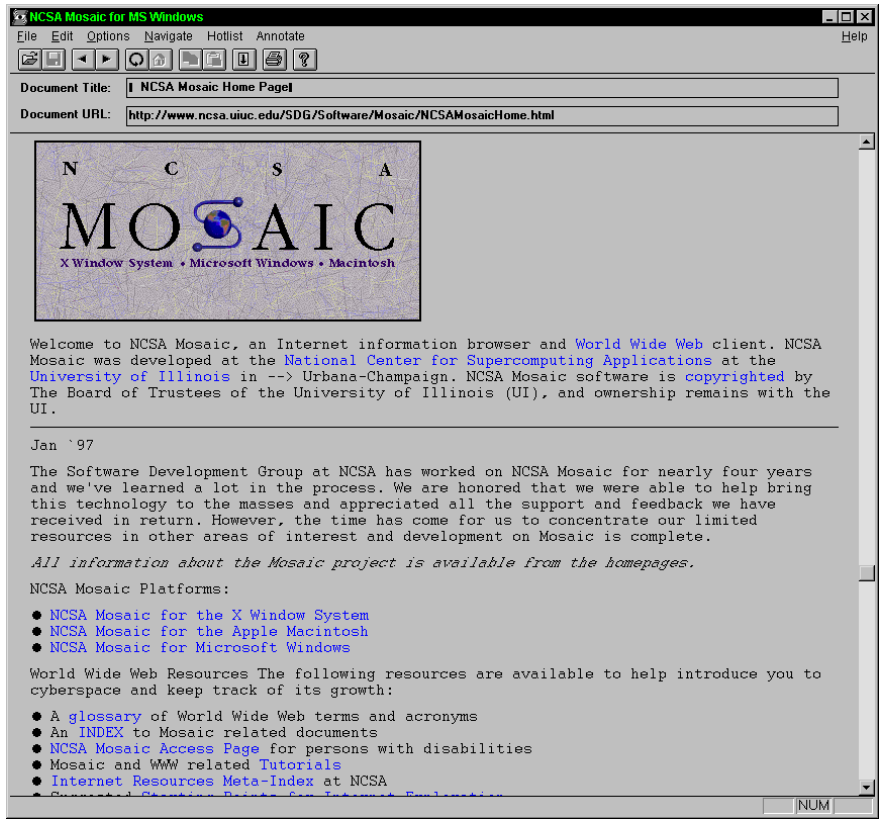


Abbildung 2.6: Mosaic[23]

Mosaic ließ sich leicht herunterladen und installieren, und es war keine lange Einarbeitungszeit erforderlich, um Zugriff auf das Web zu erhalten. Weitere Betaversionen erschienen ab September 1993 für diverse Unix-Betriebssysteme, für Apples Macintosh-System 7 sowie für Windows NT. Versionen für

23 Quelle: *http://en.wikipedia.org/wiki/Mosaic_(web_browser)*

Windows 3.1 und 95 folgten nach. Die Zahl der Mosaic-Nutzer vervielfachte sich von Tag zu Tag. Zu dieser Zeit war der Name »Mosaic« ein Synonym für »Webbrowser« oder das Web allgemein – Inhalte waren nicht »im Web«, sondern »in Mosaic«.

Netscape Navigator

Der kostenlose Mosaic verbreitete sich rasch über das Internet und machte das Web sehr schnell sehr populär. Im Jahr 1994 wurde Jim Clark auf den Browser und dessen Entwickler Marc Andreessen aufmerksam. Clark war damals Professor an der Stanford University und mit seiner Firma *Silicon Graphics Inc.* bereits ein erfolgreicher Unternehmer. Er hatte genügend Geschäftssinn, um die (kommerziellen) Möglichkeiten, die das Internet bot, zu erkennen, und gründete zusammen mit Andreessen in Mountain View (Kalifornien) die *Mosaic Communications*, die sich bald darauf in *Netscape Communications* umbenannte.

Die zu größten Teilen aus dem ehemaligen Mosaic-Team übernommenen Programmierer entwickelten den neuen Browser Netscape Navigator in ungewöhnlich kurzer Zeit von Grund auf neu. Die Version 1.0 des Navigators entsprach damals exakt der HTML 2.0-Empfehlung des W3C. Andreessen und Clark gingen aggressiv vor, um den gesamten Browsermarkt zu erobern. Sie gaben ihr Produkt kostenlos weiter und gewannen so bereits Anfang 1995, nur vier Monate nach Veröffentlichung des Navigators, über 75 Prozent des Browsermarkts.

2.7.2 Browserkrieg

1995 bemerkten Bill Gates und Microsoft, die dem Internet und vor allem dem Web bis dahin kaum Beachtung geschenkt hatten, dass sie wichtige Geschäftsfelder verpassen könnten. Gates erhob das Internet zu einem neuen und wichtigen Bestandteil der Firmenstrategie. Anstatt einen eigenen Browser zu entwickeln, lizenzierte Microsoft den Browsercode eines kleinen Abkömmlings des NCSA namens Spyglass für zwei Millionen Dollar. Nur eine Handvoll Programmierer entwickelten daraus in aller Eile einen Browser namens Internet Explorer, der im August 1995 veröffentlicht wurde. In den nächsten zwölf Monaten erschienen drei weitere Versionen.

Microsoft hatte gegenüber Netscape zwei entscheidende Marktvorteile:

1. Microsoft hatte wesentlich mehr finanzielle Mittel als die Konkurrenz, konnte mehr Entwickler beschäftigen und eine größere Marketingmaschinerie auffahren; im Jahr 1999 waren es mehr als 1.000 Mitarbeiter, die an der Entwicklung und Vermarktung des Browsers arbeiteten. Das ist eine erstaunlich hohe Anzahl für ein verhältnismäßig kleines Softwareprojekt. Denken Sie daran: Die ersten Browser wurden von einer Handvoll Studenten entwickelt.

2. Der Internet Explorer wurde in Microsofts Betriebssystem Windows integriert, das damals auf rund 95 Prozent aller PCs installiert wurde[24]. Damit eliminierte Microsoft alle Gründe für den Erwerb des Navigators effektiv.

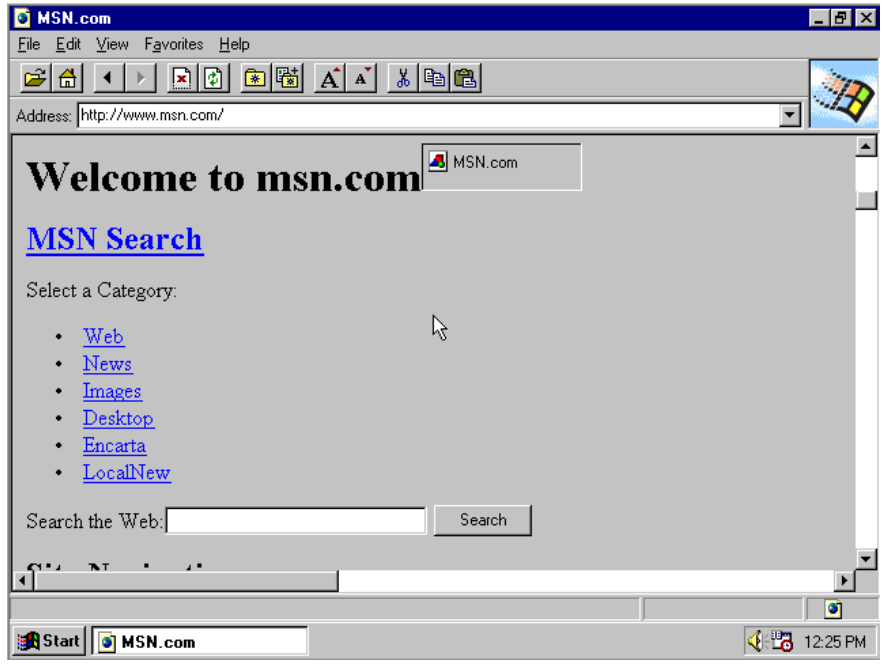


Abbildung 2.7: Internet Explorer 1.0[25]

Um die Nutzer von ihrem Browser zu überzeugen, überboten sich Netscape und Microsoft gegenseitig in der Implementierung neuer proprietärer Funktionen und Erweiterungen. Die damals aktuelle Empfehlung HTML 2.0 diente dabei eher als grobe Orientierung oder kleinster gemeinsamer Nenner, weniger als verbindlicher Standard. Der Konkurrenzkampf wandelte HTML von einer strukturorientierten Sprache zunehmend zu einem Universalwerkzeug für die Veröffentlichung und visuelle Gestaltung von Multimedia. Ein neuer Beruf entstand: der Webdesigner. Stefan Münz, technischer Autor und Webentwickler, beschreibt[26] den Geist dieser Zeit mit folgenden Worten:

> *»Träumend saßen [die Webdesigner] in den Jahren 1995 und 1996 vor ihren ersten Webseiten, die dank Netscape bunte Hintergrund- und Schriftfarben, Hintergrundtapeten, Tabellenlayouts, mehrgeteilte Bildschirmfenster (Frames)*

24 Noch heute läuft auf etwa 90 Prozent aller PC-Systeme ein Windows-Betriebssystem.
25 Quelle: *http://www.microsoft.com/windows/ie/community/columns/historyofie.mspx*
26 *http://de.selfhtml.org/intro/internet/www.htm*

und Multimedia-Plug-ins enthalten konnten. Zigtausende von Privatanwendern begannen, eigene Homepages zu erstellen. Nach und nach drängten alle Firmen, Organisationen, Regierungen und Behörden mit eigenen Angeboten ins Web – weltweit. Die Wachstumsraten glichen einer Explosion. Es war aber auch die Stunde jener Marketing-Strategen, die am liebsten aus der ganzen Welt eine Plakatwand machen würden. Mit Feuereifer und Finanzen stürzten sie sich auf das neue, aufstrebende Medium, ohne den geringsten Schimmer einer Ahnung von dessen wahrem Wesen zu haben. Kein Wunder, dass viele Versuche, die Sahnetorte Web aufzurollen, kläglich scheiterten und schließlich wieder für Katerstimmung am Werbemarkt und an der Börse sorgten.«

In den folgenden Jahren konnte Microsoft Netscape verstärkt Marktanteile abnehmen. Netscape hatte dem Aufwand Microsofts nichts entgegenzusetzen, was zur Folge hatte, dass der Marktanteil des Navigators von 1995 bis 2003 von über 80 Prozent auf unter 4 Prozent sank, während der Marktanteil des Internet Explorers im selben Zeitraum von unter 3 Prozent auf über 90 Prozent stieg. Mit der für damalige Verhältnisse hervorragenden Version 4 des Internet Explorers versetzt Microsoft Netscape 1997 den Todesstoß: Bereits im Januar 1998 kapitulierte Netscape und veröffentlichte den Quellcode seines Browsers als **Open Source**. Im Rahmen des daraus entstandenen Projekts Mozilla wurde das Programm vollständig neu geschrieben, dennoch konnte Netscape nie wieder an die unglaublichen Erfolge früherer Zeiten anknüpfen. Schließlich wurde Netscape Ende 1998 vom Onlinedienstanbieter AOL aufgekauft. Im Juli 2003 entließ AOL die letzten 50 Browserentwickler und stellte sein finanzielles Engagement am Mozilla-Projekt ein. Die Rechte an der kompletten Netscape-Technologie wurden auf die neu gegründete *Mozilla Foundation* übertragen. Microsoft hatte den Browserkrieg gewonnen.

2.7.3 Windows Internet Explorer

DEFINITION

*Der **Windows Internet Explorer** (früher Microsoft Internet Explorer, Abkürzung: IE) ist ein Webbrowser von Microsoft für das Betriebssystem Microsoft Windows. Frühere Versionen waren zum Teil auch für Mac OS und Unix-Derivate wie Solaris und HP-UX) verfügbar. Die derzeit aktuelle Version ist Windows Internet Explorer 7 (IE 7).*

Nach der Verdrängung des Netscape Navigators konnte sich der Internet Explorer viele Jahre lang einer Monopolstellung erfreuen. Vor allem der IE 6 war über mehrere Jahre der unangefochten meistgenutzte Browser im World Wide Web.

Vergleichbar mit Netscape, die sich aufgrund der damals herausragenden Marktanteile ihrer Sache zu sicher waren und daher wenig Engagement in die Weiterentwicklung des Navigators 4 steckten, ruhte sich Microsoft auf dem

Erfolg des Internet Explorers 6 aus. Nach dessen Erscheinen im Herbst 2001 wurde das Entwicklerteam nahezu aufgelöst. Dabei war der IE 6 im Grunde niemals wirklich fertig; neben einer fehlerhaften Rendering Engine, die die damals (und noch heute) aktuellen Webstandards, vor allem CSS, nur unzureichend unterstützte, waren zahlreiche Sicherheitsmängel die Folge. Letztere versuchte man Monat für Monat durch Updates zu beheben, an der Rendering Engine selbst passierte jedoch nahezu nichts mehr.

Am 19. Oktober 2006 erschien die fertige Version 7 für Windows XP und Windows Server 2003. Die deutschsprachige Version folgte am 31. Oktober.

Abbildung 2.8: Windows Internet Explorer 7

IE 8 Der IE 7 rief unter Webentwicklern zwiespältige Reaktionen hervor. Der Microsoft-Browser hatte zwar viele Änderungen an der Benutzeroberfläche erfahren, war aber in Bezug auf Webstandards nicht viel mehr als eine Art längst überfälliger **Bugfix** voller alter und neuer Probleme und Fehler. Mittlerweile steht der IE 8 in den Startlöchern, bei dessen Entwicklung Microsoft deutlich höhere Maßstäbe angelegt hat als noch beim IE 7. Die Entwicklung des IE 8 startete im August 2007. Im Dezember erschienen die ersten Einträge über den neuen Browser im Weblog der IE-Entwickler. Die erste öffentliche

Betaversion richtet sich primär an Webentwickler und steht seit dem 5. März 2008 zur Verfügung. Bereits als Beta ist der IE 8 deutlich besser als sein Vorgänger, bleibt aber noch weit hinter der Konkurrenz zurück[27].

2.7.4 Firefox

DEFINITION

Firefox *ist ein Webbrowser des Mozilla-Projekts. Der Browser ist Open Source und erfreut sich einer aktiven Entwicklergemeinde. Deshalb wird Firefox als Browser für besonders viele Plattformen angeboten, darunter Microsoft Windows, Linux und Mac OS X ab Version 10.4. Darüber hinaus gibt es Portierungen für Solaris, OS/2, FreeBSD, SkyOS und viele andere Systeme. Derzeit aktuell ist die Version 3.*

Die erste Version 0.1 erschien im September 2002, die erste Hauptversion 1.0 im November 2004. Nach nur einem Tag erreichte Firefox 1.0 eine Million Downloads. Am 31. Juli 2006 wurde der 200-millionste Firefox-Browser heruntergeladen. Die zweite Hauptversion des Browsers, Version 1.5, erschien im November 2005. Sie enthielt eine neue **Rendering Engine**, verbesserte die Einbindung in andere Systeme und optimierte Softwareupdate- sowie Erweiterungsmanager. Im Oktober 2006 erschien mit der Version 2.0 die dritte Hauptversion. Neuerungen waren ein integrierter Phishing-Filter, eine Rechtschreibprüfung, automatische Suchvorschläge, eine überarbeitete Updatefunktion sowie eine Feedreader-Integration.

FF 3

Die Hauptversion 3.0 erschien im Juni 2008. Begleitet wurde die Veröffentlichung von einer herausragenden Marketingkampagne. Die Firefox-Entwickler riefen zum Download-Day auf. Das erklärte Ziel war es, das innerhalb von 24 Stunden weltweit meistheruntergeladene Programm zu werden. Dass es eine solche Kategorie im Guinness-Buch der Rekorde bis zu diesem Zeitpunkt noch nicht gab, störte niemanden. Die Aktion erbrachte die gewünschte Aufmerksamkeit, und tatsächlich luden Nutzer das Programm mehr als 8 Millionen Mal herunter, darunter über 1,2 Millionen Downloads aus Deutschland. Zu den Neuerungen in Version 3 gehören animierte PNGs, eine neue Zoomfunktion für Text- und Grafikinhalte, eine beschleunigte JavaScript-Verarbeitung sowie die Unterstützung von Farbprofilen.

Gecko Rendering Engine

Als Rendering Engine verwendet Firefox die **Gecko** Rendering Engine, ein programmübergreifendes Softwaremodul, das eingesetzt wird, um Webseiten darzustellen. Gecko liest (X)HTML- und CSS-Dateien sowie Mediadaten ein

27 Ausführliche Betrachtungen über IE 7 und IE 8 finden Sie in meinen iX-Artikeln, siehe ***[Jendryschik 2006]*** und ***[Jendryschik 2008]***.

und stellt das Ergebnis auf dem Bildschirm als formatierten Text mit Grafiken und anderen Elementen dar. Ein besonderes Augenmerk bei der Entwicklung von Gecko wird auf die vollständige Unterstützung von offenen Internetstandards wie (X)HTML, CSS, DOM, XML, RDF und JavaScript gelegt. Die Gecko Rendering Engine ist der Grund für die weitreichende Unterstützung von Webstandards im Firefox.

Abbildung 2.9: Firefox 3.0

2.7.5 Opera

DEFINITION

Opera *ist eine kostenlos erhältliche Software und vereint Webbrowser, E-Mail-Client und weitere Werkzeuge in einem Programm. Hersteller ist das norwegische Unternehmen Opera Software ASA. Opera gibt es für zahlreiche Betriebssysteme, darunter Microsoft Windows, Mac OS X, Linus, FreeBSD und Solaris, zudem gibt es Versionen für mobile Geräte. Derzeit aktuell ist die Version 9.5.*

Im Dezember 1996 veröffentlichte die norwegische *Opera Software ASA* einen Browser namens Opera in der Version 2.10. Anders als seine Konkurrenz war er nicht kostenfrei verfügbar, konnte sich aber aufgrund seiner vielfältig anpassbaren Oberfläche und der damals herausragenden Geschwindigkeit beim Laden und Aufbauen von Seiten sowie seiner hohen Kompatibilität zu aktuellen Webstandards zur Überraschung vieler am Markt bewähren. 2000 kam für Opera der kommerzielle Durchbruch mit Version 4.0 für Windows, die innerhalb des ersten Monats über eine Million Mal heruntergeladen wurde. Im Dezember 2000 erschien Version 5.0, auf die innerhalb des ersten Folgemonats bereits über zwei Millionen Mal zugegriffen wurde. Ab dieser Version finanzierte sich Opera durch einen Werbebanner rechts neben der Menüleiste, der für 39 US-Dollar ausgeblendet werden konnte.

Seit der Version 8.5 vom September 2005 können Sie Opera kostenlos verwenden. Die aktuelle Version 9.5 wurde innerhalb der ersten fünf Tage 5 Millionen Mal heruntergeladen. Sie beinhaltet unter anderem

- den Browser selbst mit Lesezeichenverwaltung, Passwortmanager, Werbe- und Pop-up-Blocker,
- ein eingebautes E-Mail-, News- und RSS-Reader-Programm,
- einen IRC-Client zur Teilnahme am Internet Relay Chat,
- einen Downloadmanager,
- ein Adressbuch,
- die Möglichkeit schneller Notizen,
- einen integrierten BitTorrent-Client, eingebettet in den Downloadmanager, sowie
- einen Phishing-Filter, der auf die Datenbanken von GeoTrust und Phishtank zurückgreift.

Opera ist ein hervorragender Browser mit einer großen Fangemeinde, die nicht müde wird, die Vorteile des Browsers gegenüber seinen Konkurrenten hervorzuheben. Und in der Tat ist Opera anderen Browsern, was Innovationen angeht, häufig ein Stück voraus.

Abbildung 2.10: Opera 9.5

2.7.6 Konqueror

DEFINITION

Konqueror *ist der Dateimanager und Webbrowser der grafischen Betriebssystemoberfläche KDE, die auf Unix-Betriebssystemen wie Linux, BSD oder Solaris eingesetzt werden kann. Die Komponente* ***KHTML*** *sorgt dafür, dass Konqueror ein vollwertiger Browser ist. Aktuell ist die Version 4.x.*

Der Name Konqueror setzt augenzwinkernd die Tradition vorheriger Browsergenerationen fort: Zunächst gab es den Seefahrer (navigator), anschließend den Erforscher (explorer) und nun eben den Eroberer (conqueror).

Zu den fortgeschrittenen Funktionen des Browsers gehören die automatische Vervollständigung von URIs, das automatische Ausfüllen von Formularen, die Verwendung einer Rechtschreibkontrolle mit freier Sprachwahl sowie die Fähigkeit, Lesezeichen anderer Browser zu importieren, und das Browsen in Karteikartenfenstern. Die Unterstützung von Webstandards im Konqueror ist seit KHTML 3.5 hervorragend.

Konqueror gehört auf Linux-Systemen zu den wichtigsten Browseralternativen.

Abbildung 2.11: Konqueror 3.4.0

2.7.7 Safari

DEFINITION

Safari *ist ein Webbrowser der Firma Apple für das Betriebssystem Mac OS X und seit Version 3.1 auch für Microsoft Windows. Safari gehört zum Lieferumfang von Mac OS X ab der Version 10.3 (Panther) und ersetzte den vorher mitgelieferten Microsoft Internet Explorer für Mac als Standardbrowser. Aktuell ist die Version 3.1.1.*

Die Unterstützung aktueller Webstandards ist hervorragend. Safari verwendet zum Darstellen von HTML-Seiten Apples WebKit, eine Anwendungsarchitektur (Framework), die auf der KHTML-Bibliothek des KDE-Projekts basiert. Apple hat jedoch einige Änderungen vorgenommen – einerseits, um eine verbesserte Anbindung an andere Mac OS X-Bibliotheken zu ermöglichen, und andererseits, um die bestmögliche Darstellung aller Webseiten zu garantieren.

Safari verfügt über einen Pop-up-Blocker und ermöglicht das Lesen und Archivieren von RSS-Nachrichten. Der Browser erhält eine Funktion zum Surfen ohne Cache, Cookies und andere Datenspuren.

Abbildung 2.12: Safari 3.1.1 auf Windows XP

Safari 3.0 erschien im Juni 2007 und wurde erstmals auch auf Windows XP und Vista portiert. Die Windows-Version wurde laut Apple innerhalb von 48 Stunden mehr als eine Million Mal heruntergeladen, war allerdings noch sehr fehlerhaft. Im März 2008 erschien die Version 3.1 für Mac OS sowie auch für Windows, die zahlreiche Verbesserungen mit sich brachte, darunter eine stark verbesserte CSS 3-Unterstützung.

2.7.8 Lynx

DEFINITION

Lynx *ist ein textbasierter Browser und unterscheidet sich damit wesentlich von den bisher vorgestellten Browservarianten. Er kann lediglich Text darstellen, keine Bilder oder andere Multimedia-Elemente. Lynx wurde bereits 1992 an der Universität von Kansas als Hypertextbrowser entwickelt und im März 1993 als 2.0 für das Web freigegeben. Lynx wird als GPL-Software (GNU General Public License) vertrieben, das heißt, er darf frei kopiert und benutzt werden. Die derzeit aktuelle Version läuft auf Unix, Windows ab Version 95, DOS und weiteren Plattformen.*

Textbrowser sind im Vergleich zu ihren grafikbasierten Konkurrenten sehr schnell und gelten als absturzsicher. Sie werden von vielen Entwicklern nicht ernst genommen, dabei sind sie ein hervorragendes Mittel, um zu testen, ob Webseiten ohne CSS, JavaScript und Grafiken bedienbar sind. Darüber hinaus vermitteln sie ein Gefühl dafür, wie **Suchmaschinen** und Nutzer assistiver Technologien, beispielsweise Screenreader, eine Webseite interpretieren beziehungsweise lesen. Textbrowser sind eine hervorragende Hilfe bei der Strukturierung von Inhalten. Lynx findet auch Verwendung als Bestandteil von **Crawlern**.

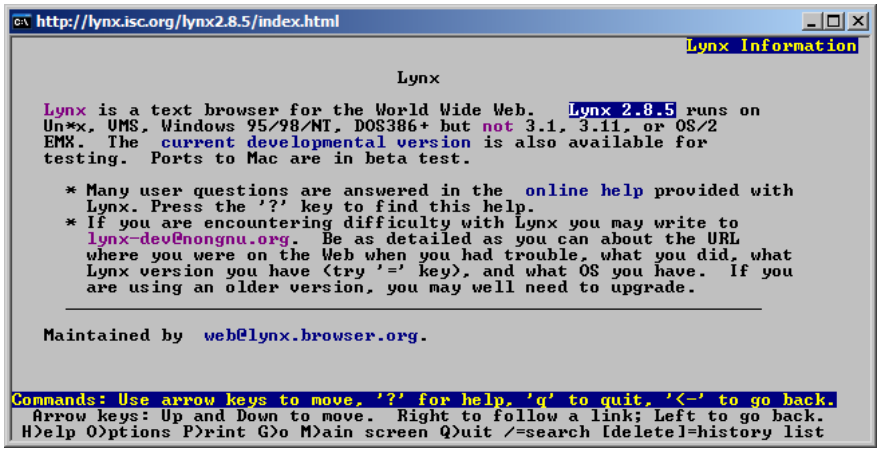


Abbildung 2.13: Lynx 2.8.5

2.7.9 Browsertests

Lebten wir in einer perfekten Welt, bräuchte ein Webautor im Wesentlichen nur drei Dinge für seine tägliche Arbeit:

- einen Editor zum Schreiben gültiger und semantisch reichhaltiger (X)HTML-Dokumente und dazugehöriger Stylesheets,
- Zugriff auf relevante Spezifikationen, Empfehlungen und Standards sowie
- einen Browser seiner Wahl, um die Ergebnisse seiner Arbeit online zu betrachten.

Leider ist die Welt nicht perfekt; daher gibt es keinen einzigen Browser, der die Empfehlungen vollständig und fehlerfrei unterstützt. Webautoren bleibt damit nichts anderes übrig, als ausgiebig zu testen. Da niemand alle Browser dieser Welt installieren und seine Arbeit darauf überprüfen kann, beschränken sich Webautoren auf eine kleine Auswahl. Dabei ist es empfehlenswert, die ungefähren Marktanteile von Browsern und Betriebssystemen zu beachten.

Der mit großem Abstand meistverwendete Browser ist – noch immer – der Windows Internet Explorer von Microsoft. Der geschätzte Marktanteil liegt bei 70 bis 80 Prozent, wobei mindestens ein Drittel davon weiterhin auf den IE 6 fallen dürfte. Die 5er-Versionen des IE sind heute nicht mehr von Bedeutung, weder auf Windows- noch auf Macintosh-Systemen. Diese können Sie beim Testen folglich vernachlässigen. Dafür sollten Sie ein Auge auf aktuelle Betaversionen des IE 8 werfen.

Mehrere Versionen des Internet Explorers lassen sich nicht ohne Weiteres zusammen auf einem System betreiben. Installiert man den einen, verliert man den anderen. Für Webautoren ist es jedoch zwingend notwendig, auf beiden Browsern testen zu können. Wer folglich nicht gerade zwei Rechner zu Hause stehen hat, muss sich mit einer der drei folgenden Möglichkeiten zufriedengeben.

- Eine sehr gute Alternative zu einem vollwertigen Zweitsystem ist eine virtuelle Umgebung (siehe unten). Auf einer virtuellen Maschine lässt sich der IE 7 unter realen Bedingungen testen und installieren.
- Solange der IE 8 noch nicht fertig ist, sollten Sie – ein Windows-Betriebssystem vorausgesetzt – die Version 7 auf Ihrem System installiert haben. Mithilfe der Software »Multiple IE« können Sie andere IE-Versionen als Standalone-Version parallel installieren. Das Programm bietet IE 3, IE 4, IE 5, IE 5.5 und IE 6 in einem Paket. Wie das funktioniert, können Sie bei Tredosoft nachlesen und »Multiple IE« dort herunterladen: *http://tredosoft.com/Multiple_IE*.
- Noch einfacher ist der IETester. Dabei handelt es sich um einen Browser, der die Rendering- und JavaScript-Engines des Internet Explorers in den Versionen 5.5, 6, 7 und 8 beta 2 in einer Oberfläche zusammenfasst. IETester läuft unter Windows Vista und XP mit installiertem IE 7 oder 8. Das Werkzeug befindet sich noch im Teststadium, läuft aber bereits einigermaßen stabil. Sie können den IETester hier herunterladen: *http://www.my-debugbar.com/wiki/IETester/HomePage*.

* * * TIPP

Testen Sie im IE 7 und in aktuellen Betaversionen des IE 8. Testen Sie auch im IE 6, allerdings können Sie in diesem veralteten Browser den einen oder anderen Darstellungsfehler hinnehmen. Alle IE 5-Varianten auf Windows- und Macintosh-Betriebssystemen können Sie vernachlässigen.

Browser, die auf der Gecko Rendering Engine basieren (Firefox, **Camino**, Galeon, Epiphany, ...), ähneln sich in ihrer Interpretation von Webstandards so stark, dass Sie nicht alle zu testen brauchen. In der Regel reicht es aus, auf einer aktuellen Firefox-Variante zu testen und hin und wieder auch mal einen Blick auf ältere Versionen zu werfen.

* * * TIPP

Testen Sie im aktuellen Firefox und dazu hin und wieder die ältere Hauptversion 2.0.0.17. Ein Blick auf den auf Macintosh-Plattformen beliebten Camino kann nicht schaden.

Opera hat zwar zu keiner Zeit große Marktanteile erobern können, wird aber vor allem von Nutzern verwendet, die sich bewusst und überdurchschnittlich häufig im Web bewegen. Daher sollten Sie diesen Browser nicht völlig außer Acht lassen. Gerade ab der ersten kostenlosen Version 8.5 sollte intensiv getestet werden, aber auch ältere Versionen werden noch von einigen Leuten eingesetzt, die auf leistungsschwachen Rechnern unterwegs sind. Je älter die Versionen sind, desto weniger sollten Sie auf eine perfekte Darstellung achten; es reicht aus, dass alle Inhalte zugänglich sind.

* * * TIPP

Testen Sie im jeweils aktuellen Opera, dazu hin und wieder ältere Final-Versionen wie 8.54 oder 7.54.

Auf Unix- und Linux-Systemen wird neben Mozilla-Varianten vor allem Konqueror im *K Desktop Environment* eingesetzt. Tests auf diesem Browser sind durchaus sinnvoll.

* * * TIPP

Testen Sie gelegentlich aktuelle Konqueror-Versionen.

Für ein Mac OS X-System haben Nutzer eine hervorragende Browserauswahl: Neben Mozilla-Varianten und Opera stehen mit Safari und iCab zwei weitere Browser bereit, die die aktuellen Webstandards gut bis sehr gut unterstützen. iCab ist ein bei vielen Mac-Nutzern sehr beliebter Browser für Mac OS 7.1 bis zur aktuellsten OS X-Version. War dessen CSS-Unterstützung in Vorversionen noch eher mangelhaft, ist iCab in der Version 3 erwachsen geworden und wartet mit einer hervorragenden HTML- und erstaunlich weitreichender CSS 2.1-Unterstützung auf – in einigen Bereichen besser als die Konkurrenz Safari oder Mozilla. Aktuell ist die Version 4.x.

* * * TIPP

Testen Sie die neuesten Safari- und iCab-Versionen auf einer aktuellen Macintosh-Plattform. Testen Sie darüber hinaus den neuesten Safari-Browser unter Windows.

Lynx

Textbrowser eignen sich hervorragend dazu, Semantik und Struktur von Webseiten sowie deren allgemeine Zugänglichkeit zu überprüfen. Vor allem Lynx sollte jeder Webautor installieren und die Darstellung seiner Website darauf intensiv überprüfen.

* * * TIPP

Testen Sie in einem aktuellen Textbrowser, vorzugsweise Lynx.

2.7.10 Bezugsquellen und Testumgebungen

Wenn Sie einen bestimmten Browser installieren möchten, besuchen Sie die Website des Herstellers und laden sich den gewünschten Browser herunter, entweder direkt von der Startseite oder aus dem Archiv. Tabelle 2.8 führt Sie direkt zu den Archiven der Herstellerseiten. Werden Sie dort nicht fündig, ist die Website *http://browsers.evolt.org/* die beste Anlaufstelle im Netz, um jeden beliebigen Browser in jeder beliebigen Version zu finden und herunterzuladen. Dort treffen Sie auch auf »antike« Browser wie Mosaic oder den Netscape Navigator 4.

Browser	Version	Bezugsquelle
Internet Explorer	8	www.microsoft.com/windows/products/winfamily/ie/ie8/
	7	www.microsoft.com/windows/products/winfamily/ie/
Firefox	3.0	www.mozilla-europe.org/de/firefox/
Camino	1.x	www.caminobrowser.org
Opera	9.x	www.opera.com/download/index.dml?custom=yes
	Archiv	arc.opera.com/pub/opera
Konqueror	4.x	www.konqueror.org
Safari	3.x	www.apple.com/de/macosx/features/safari.html
iCab	4.x	www.icab.de
Lynx	2.8.x	lynx.browser.org

Tabelle 2.8: Bezugsquellen aktueller Browser

Wer einen Windows-Rechner zu Hause stehen hat, besitzt in den seltensten Fällen zusätzlich einen Macintosh-Rechner und umgekehrt. Das Gleiche gilt für Unix-Systeme. Darüber hinaus ist es oftmals nicht möglich, unterschiedli-

che Versionen eines Browsers parallel auf seinem System zu installieren. Man könnte Freunde oder Kollegen in entsprechenden Newsgroups oder Foren höflich um Ansicht einer Website und vielleicht einigen Screenshots bitten, aber auf Dauer ist dies keine Lösung. Hilfestellung bieten eingeschränkte Standalone-Versionen, Bildschirmfotodienste und virtuelle Umgebungen.

Bildschirmfotodienste

Bildschirmfotodienste sind teils kommerzielle, teils kostenfreie Dienste, die Bildschirmabzüge (*screenshots*) von der Darstellung von Webseiten in bestimmten Browsern anfertigen und dem Nutzer zur Anzeige im Browser oder zum Download anbieten.

- **ieCapture**

  ieCapture von Daniel Vine fertigt einen Bildschirmabzug der Darstellung in der aktuellsten Vorschauversion des IE 8 an: *http://www.danvine.com/iecapture*.

- **Browsershots**

  Browsershots fertigt Screenshots auf verschiedenen Browsern in erstaunlich vielen Versionen an, sowohl unter Windows als auch unter Linux, Mac OS und BSD. Der Dienst ist kostenlos: *http://browsershots.org*.

- **BrowserCam**

  Die BrowserCam ist ein kommerzieller Dienst und dabei vergleichsweise teuer. Allerdings besteht die Möglichkeit, einen Testzugang anzulegen. Das Angebot an Plattformen und Browsern ist beeindruckend. Auf den Systemen Linux Fedora Core 6, Mac OS X 10.3, 10.4 und 10.5, Windows 2000 Professional, 98, XP und Vista sind jeweils rund ein halbes bis ein Dutzend Browser installiert, teilweise in unterschiedlichen Versionen: *http://www.browsercam.com*.

Bildschirmabzüge zeigen sehr gut, wie eine Webseite in dem einen oder anderen Browser aussieht, sie sind jedoch kein Werkzeug für die Fehleranalyse. Im Allgemeinen muss man die Stelle, die einen Fehler verursacht, erst suchen und eine Weile herumprobieren, bis man das Problem beseitigt hat. Jedes Mal einen Bildschirmabzug anzufertigen, wenn man eine Kleinigkeit geändert hat, ist nicht nur nervig, sondern vor allem zeitaufwendig. Bei allen vorgestellten Diensten ist mitunter mit mehreren Stunden bis Tagen Wartezeit zu rechnen. Virtuelle Umgebungen sind oftmals die bessere Alternative.

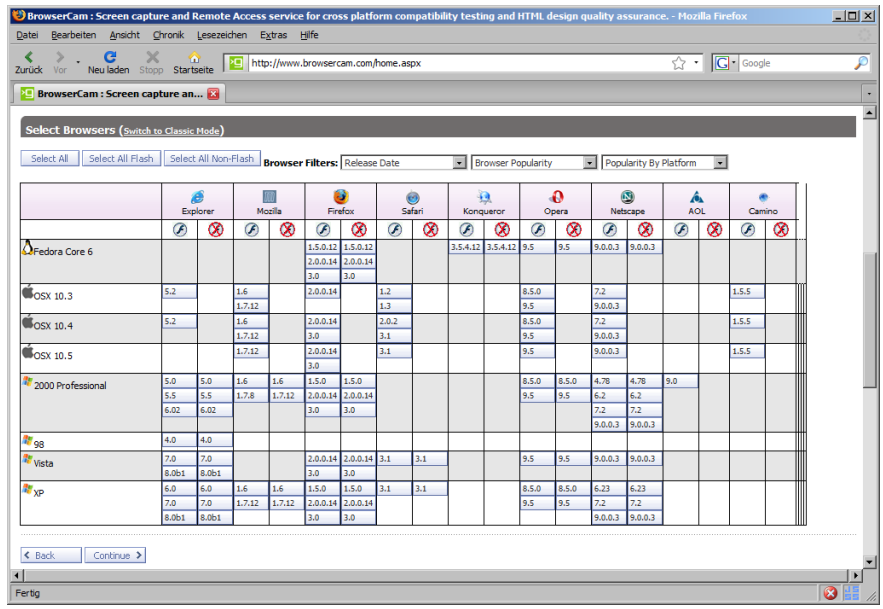


Abbildung 2.14: BrowserCam ermöglicht Ihnen einen Screenshot Ihrer Websites in zahlreichen Browsern auf verschiedenen Systemen

Virtuelle Umgebungen

Virtuelle Umgebungen sind Browsersimulationen oder Programme, die auf einem ausreichend mit Speicher ausgestatteten Rechner gleichzeitig verschiedene Betriebssysteme laufen lassen; ein Mausklick, und schon steht ein neuer Test-PC zur Verfügung.

VMware[28] bietet mit dem VMware Player eine kostenlose Software, mit der Sie bereits fertig eingerichtete virtuelle Maschinen starten können. Im *Virtual Technology Network* unter der Adresse *http://www.vmware.com/appliances/* können Sie zudem komplette VMware-Images herunterladen. So ist es möglich, sich ohne Kostenaufwand beispielsweise eine Linux-Umgebung unter Windows einzurichten, um damit ausgiebig zu testen.

28 *http://www.vmware.com*

Wem das zu viel Aufwand ist, der hat auch noch andere Möglichkeiten:

- **Knoppix**

  Knoppix ist ein komplett von CD oder DVD lauffähiges Linux-System für den Desktop. Einfach CD/DVD einlegen, den Rechner hochfahren und fertig. Es ist keinerlei Installation auf der Festplatte notwendig. Auf diesem Weg ist es möglich, Linux-Browser zu installieren und zu testen, beispielsweise Konqueror: *http://www.knopper.net/knoppix*.

- **BrowserCam**

  Die BrowserCam bietet nicht nur die Möglichkeit, Screenshots anzufertigen, sondern auch einen vollständigen Fernzugriff auf unterschiedliche Plattformen und darauf installierte Browser. Eine Webseite lässt sich dadurch so testen, als säßen Sie direkt an dem entsprechenden Rechner: *http://www.browsercam.com*.

- **Lynx Viewer**

  Es ist zwar relativ einfach, Lynx selbst zu installieren und seine Webseite damit direkt zu betrachten, zur Not tut es aber auch ein Online-Viewer, der die Darstellung in Lynx simuliert: *http://www.kk-software.de/Lynxview.html*.

Weitere Simulationen stehen in Form von Browsererweiterungen (siehe Kapitel 3.5) zur Verfügung.

2.8 Webstandards

Egal welche komplizierten oder tollen Techniken wir zur Erschaffung unserer Website oder Applikation nutzen, am Ende müssen wir immer mindestens (X)HTML ausgeben, um etwas in einem Browser anzuzeigen. Soll das Ganze auch noch hübsch aussehen, nutzen wir CSS. Und soll auch noch Dynamik rein, kommen wir um JavaScript nicht herum. All diese Techniken werden Webstandards genannt.

2.8.1 Was ist ein Standard?

Bisher habe ich immer wie selbstverständlich von »Webstandards« gesprochen. Aber was verstehen wir eigentlich unter einem Standard? Wikipedia meint hierzu[29]:

29 *http://de.wikipedia.org/wiki/Standard*

DEFINITION

*Ein **Standard** ist eine vergleichsweise einheitliche oder vereinheitlichte, weithin anerkannte und meist auch angewandte (oder zumindest angestrebte) Art und Weise, etwas herzustellen oder durchzuführen, die sich gegenüber anderen Arten und Weisen durchgesetzt hat. [...] Ein Standard kann in einem formalisierten oder nicht formalisierten Regelwerk bzw. in einem sich ungeplant ergebenden Regelfall bestehen, beispielsweise in einer einzelnen Regel bzw. mehreren Regeln oder einer Norm.*

Standards sind dynamisch

XHTML und CSS sind genau wie JavaScript formalisierte Regelwerke, die beschreiben, wie Inhalte einer Seite organisiert und mit Layout und Dynamik versehen werden sollen. An diesen Regeln haben unter anderem alle Browserhersteller mitgearbeitet. Diese Regeln sind nicht endgültig, sondern werden Überarbeitungen unterzogen. Auch die Standards tragen der Dynamik des Internets Rechnung, obgleich in zeitlich großem Abstand.

2.8.2 Der Sinn von Standards

Der Sinn von Standards erschließt sich, wenn man in die Anfangszeit des Webs zurückblickt. Die Browserhersteller versuchten, sich mit Eigenerfindungen zu übertreffen und ihre Browser mit neuen, coolen Features attraktiver zu machen. Es war notwendig, Webseiten für einen Browser zu »optimieren«. Wollte man zwei Browser bedienen, begann das Abenteuer. Die Alternative war die Ausgrenzung großer Nutzerscharen. Wenn Sie für das Internet entwickeln, können Sie niemals voraussehen, welchen Browser in welcher Konfiguration ein Nutzer hat. Eine Fokussierung auf Spezialfähigkeiten eines Browsers ist also eine Sackgasse, die Ihnen deutlichen Mehraufwand und Ihren Kunden dementsprechend höhere Kosten aufnötigt.

Eine Standardisierung ist wirtschaftlich vernünftig. Sie schafft auch eine gewisse Zukunftssicherheit, denn es ist höchst wahrscheinlich, dass künftige Browser alten Code interpretieren können, wenn er sich an Standards hält. Dies können wir beispielsweise von Elementen nicht sagen, die nur in einem bestimmten Browser funktionieren – beispielsweise `layer` bei Netscape.

2.8.3 Die Realität der Browserhersteller

So schön die Standardisierung in der Theorie ist, so traurig ist oft die Realität. Jahrelang konnten wir eine Trennlinie zwischen Microsoft, das sich wenig für Standards interessierte, und den anderen Browserherstellern ziehen. Diese Trennung scheint sich aufzulösen. Microsoft hat verstanden, dass es im Interesse aller ist, wenn ihre Produkte den Standards vermehrt folgen. Eine Besserung ist mit dem IE 8 auf alle Fälle zu konstatieren, trotz einiger Kritikpunkte. CSS 2 wurde 1998 als

W3C-Empfehlung veröffentlicht, also als Standard bekannt gegeben. Der IE 8 steht zehn Jahre später vor der Tür und beherrscht noch immer nicht alle damals festgelegten Attributselektoren. Und obwohl Safari, Opera und Firefox jeweils gute Unterstützung der Webstandards (X)HTML und CSS bieten, so können sie doch nicht von sich behaupten, diese komplett und fehlerlos anzubieten[30]. Die Unterstützung des Media-Attributs für Handybrowser ist nahezu gleich null, was eine Entwicklung alternativer Oberflächen für Handys erschwert.

Der beste Standard ist also nur so gut, wie es die Browserhersteller zulassen. Auch alternative Zugangsgeräte wie Screenreader (siehe Kapitel 2.9.2) haben manchmal ihre standardfernen Eigenheiten. Findet beispielsweise **JAWS** keine Legende in einem Fieldset (siehe Kapitel 5.10.4), nimmt es die nächstgelegene Überschrift stattdessen als Ersatz.

2.8.4 Die Realität der Standards

Natürlich ist es einfach, den Browsern schlechte oder mangelnde Implementierung der Standards vorzuwerfen. In manchen Situationen sind aber auch die Standards sehr unbestimmt, sodass den Browserprogrammierern nur eigene Interpretationen bleiben. Sehr auffällig ist dies bei Formularen. Eric Meyer hat in einem Artikel ***[Meyer 2007]*** sehr schön seziert, warum wir beim Styling von Formularen mittels CSS so großen Differenzen und »Unmöglichkeiten« begegnen.

Wir dürfen nie außer Acht lassen, dass HTML und CSS noch von Menschen entwickelt wurden, denen der reine Informationsaustausch über alles ging. Zudem hatten wir Ende der 90er-Jahre des letzten Jahrtausends noch keine Webapplikationen wie heute. Das Internet hatte einen ganz anderen Charakter. Die heutigen Ansprüche sind mit Auszeichnungssprachen, die für den Austausch wissenschaftlicher Dokumente geschaffen waren, nicht adäquat zu bewältigen. Die Standards selbst müssen sich weiterentwickeln. Und nach der fälligen Weiterentwicklung müssen nicht nur die dann aktuellen Browser diese neuen Standards implementieren, sondern wir müssen auch eine Lösung für den Umgang mit den alten Browsern finden.

IE 6

Wir sehen es derzeit am Anteil des IE 6. Dieser Browser kam 2001 auf den Markt, und obwohl Microsoft mittlerweile schon die zweite Nachfolgeversion vorlegt, ist der Anteil des IE 6 noch hoch. Um nicht ganz in der Entwicklung stehen zu bleiben und uns am schwächsten Glied – dem IE 6 – zu orientieren, müssen wir also Strategien dazu entwickeln, wie wir gleichzeitig moderne Webseiten erstellen und den veralteten Browser nicht komplett aussperren. Diese Strategien existieren (beispielsweise das Progressive Enhancement). Das

30 siehe *http://www.webdevout.net/browser-support-summary*

letzte Problem sind die Auftraggeber, die selbst den IE 6 noch nutzen (müssen) und dementsprechend eine Ungleichbehandlung der Browser nicht goutieren.

2.8.5 Standards sind mehr als nur pure Regeln

Ich möchte dem Missverständnis vorbeugen, es ginge bei der Anwendung der Webstandards nur um ein paar wenige Elemente und Regeln, die Sie lernen und anwenden müssen. Die sogenannte Webstandards-Bewegung – in Deutschland vor allem durch die Webkrauts[31] präsent – wendet viel Energie auf, um Webautoren und -entwicklern zu erklären, dass korrekte Webseiten mehr sind als eine Ansammlung korrekt geschriebener Tags.

Semantik und Gültigkeit

Die Basis einer Webseite ist die sinnvolle Anwendung der zur Verfügung stehenden Elementtypen. Allgemein wird dies als Semantik bezeichnet. Listen sollten genau so mit dem dafür passenden Element ausgezeichnet werden wie Tabellen oder Zitate. Elemente sollten nicht »missbraucht« werden, wie viele Webautoren es beim Tabellenlayout tun. Tabellen sind für tabellarische Daten gedacht. Eine valide Webseite ist noch nicht automatisch eine semantisch sinnvolle Webseite. Validität bedeutet nur, dass alle Elemente korrekt geschrieben wurden. Das bedeutet nicht zwangsläufig, dass sie auch sinnvoll eingesetzt wurden. Es ist wie bei unserer Sprache. Ein grammatikalisch korrekter Text muss nicht automatisch inhaltlichen Sinn ergeben (siehe auch die Kapitel 3.2.6 und 5.9.1).

2.8.6 Was also sind Webstandards?

DEFINITION

Webstandards *sind Regeln, auf die sich die wichtigsten handelnden Unternehmen im Web als Basis der Interpretation von Webseiten geeinigt haben. Webstandards sind Regeln, an die sich Webautoren halten sollten. Im allgemeinen Sprachgebrauch fokussieren sich diese Standards auf (X)HTML, CSS, das DOM und JavaScript. Implizit wird mit dem Begriff »Webstandards« aber auch die sinnvolle Anwendung der Regeln gemeint.*

Webstandards sind derzeit nur ein Versprechen, das von den Browserherstellern mehr oder weniger gut, niemals aber komplett eingelöst wird. Es ist nicht hilfreich, entschuldigend auf die Browserhersteller zu verweisen. Wir alle sollten nicht die Fehler der Vergangenheit wiederholen und für einen bestimmten Browser optimieren. Erst sollten wir alles korrekt schreiben, den Webstandards entsprechend. Danach kann man immer noch schauen, wie man den Fehlinterpretationen einiger Browser begegnet.

31 *http://www.webkrauts.de*

2.9 Barrierefreiheit

»Stell dir vor, du kaufst eine Zeitung. Beim Lesen in der U-Bahn ist es eng, da hat sie A5-Format. Auf dem Fußweg zur Arbeit lässt du dir einen Artikel vom Walkman vorlesen. Im Büro liest du sie auf dem Schreibtisch, da hat sie A2. Abends gibst du sie deiner Oma zu lesen und machst dafür die Schrift größer. Und das alles mit ein und derselben Zeitung! Geht leider nicht mit Papier; mit einer Website geht so etwas.« (Michael Nahrath)

2.9.1 Warum dies auch eine Einführung in barrierefreies Webdesign ist

Ergonomische Websites, zugängliche Websites, behindertengerechte Websites, barrierefreie Websites, Benutzbarkeit, **Usability**, **Accessibility** … die Anzahl an Schlagwörtern, die uns seit Jahren ständig um die Ohren fliegen, nimmt stetig zu. Dass es dermaßen viele Begriffe für und Diskussionen über einen im Grunde auf der Hand liegenden und selbstverständlichen Sachverhalt gibt, nämlich für den Wunsch nach einem für alle Menschen offenen World Wide Web, mag verwundern. Vor allem deshalb, da das Web von jeher als universell zugängliche Informationsplattform entworfen worden ist. Das heißt auch, dass Webautoren Websites auf eine Art und Weise gestalten sollten, die Menschen mit Sinnes- und Körperbehinderungen das Surfen im Netz erleichtert, anstatt sie daran zu hindern. Ein Blick auf das World Wide Web der Gegenwart ist jedoch noch immer nicht zufriedenstellend.

Perfekte und behinderte Nutzer

Wie Sie in Ihrer alltäglichen Arbeit und in zahlreichen Diskussionen in Newsgroups, Mailinglisten und Webforen feststellen können, existiert in den Köpfen vieler Webentwickler eine künstliche Einteilung der potenziellen Besucher einer Website in »normale« Nutzer auf der einen und eingeschränkte Nutzer auf der anderen Seite.[32] Einerseits wird der Typus eines perfekten Benutzers konstruiert, dessen physische Konstitution sowie Systemausstattung keinerlei Probleme erwarten lasse, da für ihn per se keine Barrieren existierten. Auf der anderen Seite gäbe es die Minderheit der behinderten Benutzer, auf die entweder nicht eingegangen werden müsse, da sie nicht der Zielgruppe entspreche, oder aber die enormen Aufwand nötig mache, um die Website für sie zugänglich zu gestalten. Sowohl bei vielen Auftraggebern als auch bei Webentwicklern wird Barrierefreiheit aufgrund dieser diskriminierenden und fragwürdigen Einteilung als vermeidbarer Kostenfaktor beziehungsweise lästige, zusätzliche Arbeit missverstanden.

32 vgl. ***[Schäfer]***

Dabei ist Barrierefreiheit keine Eigenschaft, die Sie einer Website nachträglich überstülpen können, sondern ein wichtiger Teil des Gesamtkonzepts, von dem letztendlich alle Benutzer profitieren. Schließlich bedeutet Barrierefreiheit nichts anderes, als dass eine Website **für alle Menschen zugänglich** ist.

DEFINITION

Barrierefreies Webdesign *ist die Gestaltung einer Website derart, dass sie Benutzern in so hohem Maße wie möglich zur Verfügung steht, auch wenn diese Einschränkungen irgendeiner Art (zum Beispiel Sinnes- und/oder Körperbehinderungen) unterliegen.*

Zielgruppenmaximierung

Barrierefreiheit ist Zielgruppenmaximierung! Webentwickler, die sich um Barrierefreiheit bemühen, legen in den meisten Fällen auch Wert auf **Ergonomie**, einfache Dialog- und Menüführung, Kompatibilität und Standardkonformität. All dies ist dann wichtig, wenn Sie über Ihre Website Gewinne erzielen oder neue Kunden gewinnen wollen. Ein Nutzer, der Ihnen durch die Lappen geht, weil ihm Ihre Website nicht zugänglich ist, wird niemals zu Ihrem Kundenkreis gehören, niemals bei Ihnen einkaufen oder auch niemals Dienstleistungen, die Sie online anbieten, in Anspruch nehmen können (und dies sicherlich auch nicht wollen).

Diese Einführung in XHTML, CSS und Webdesign kann Themen wie Web-Ergonomie und Barrierefreiheit nur am Rande behandeln. Die durchgehende Trennung von semantischer Struktur und Präsentation gewährt jedoch bereits ein gewisses Maß an Zugänglichkeit, und die meisten Beispiele wurden mit Blick auf maximale Barrierefreiheit entwickelt. Dadurch werden Sie hoffentlich ermutigt, diesem Aspekt Ihrer Arbeit ebenfalls große Bedeutung beizumessen – dieser Abschnitt soll Ihnen dabei helfen. Als weiterführende Lektüre empfehle ich Ihnen folgende Bücher:

Buchtipps

- »Barrierefreies Webdesign« von Jan Eric Hellbusch und anderen ausgewiesenen Fachleuten ***[Hellbusch 2004]***,
- »Webdesign & Web-Ergonomie« von Heide Balzert ***[Balzert 2004]***,
- »Barrierefreies Webdesign – Attraktive Websites zugänglich gestalten« von Angie Radtke und Dr. Michael Charlier ***[Radtke, Charlier 2006]*** sowie
- »Building Accessible Websites« von Joe Clark ***[Clark 2002]***.

2.9.2 Welche Arten von Behinderungen müssen Sie berücksichtigen?

In der EU gibt es 38 Millionen Menschen mit Behinderungen verschiedenster Art, von leichten Behinderungen (Sehschwächen) bis hin zu sehr schweren Behinderungen (wie Blindheit oder schweren Mehrfachbehinderungen). Nach vorsichtigen Schätzungen gelten rund 39 Millionen US-Amerikaner (15 Prozent der Gesamtbevölkerung) als behindert. Die W3C-Entwurfsanmerkung »Web Content Accessibility Guidelines Impact Matrix« ***[DN-WCAGIM 1999]*** stellt umfangreiche Informationen darüber zusammen, auf welche Arten von Einschränkungen Webautoren bei der Entwicklung von Webseiten achten müssen. Dabei wird grundsätzlich zwischen folgenden Einschränkungen unterschieden:

- Sehbehinderungen,
- Hörbehinderungen,
- kognitiven, Lern- und Sprachbehinderungen,
- körperlichen Einschränkungen und Epilepsie.

Sehbehinderungen

Es gibt eine große Vielzahl von Sehbehinderungen, angefangen bei Farbfehlsichtigkeit über Sehschwäche bis hin zur vollkommenen Blindheit.

Farbfehlsichtigkeit

Personen mit einer Farbfehlsichtigkeit, beispielsweise einer Rotgrünblindheit, brauchen starke Kontraste und klare Schriften sowie Kontrolle über die Farbe von Schrift und Hintergrund.

Blindheit

Blinde Menschen oder Menschen mit Sehschwäche verwenden unter anderem Bildschirmauslesegeräte beziehungsweise -software (Screenreader), Bildschirmlupen, **Braillezeilen** oder gewöhnliche Browser mit zum Teil extrem vergrößerter Schrift. Screenreader geben Texte entweder in synthetischer Sprache oder in Blindenpunktschrift (Braille) über eine Braillezeile aus. Bei Letzterem handelt es sich um ein spezielles Ausgabegerät, das über ein taktiles Display mit in den meisten Fällen acht beweglichen Stiften Punktemuster entsprechend dem Braille-Alphabet ausgibt. In beiden Fällen handelt es sich um eine eindimensionale Ausgabe, die gegenüber der zweidimensionalen Ausgabe auf dem Bildschirm enorme Einschränkungen mit sich bringt.

33 Quelle: *http://upload.wikimedia.org/wikipedia/de/4/48/Biene2004-29.jpg*

Abbildung 2.15: Verschiedene Hilfsmittel erleichtern oder ermöglichen behinderten Menschen die Nutzung des World Wide Webs[33]

Alternative Inhalte

Das einzige Medium, das im Web jedem Benutzerprogramm zugänglich ist, ist einfacher Text. Für alle Nicht-Text-Elemente ist daher ein Text-Äquivalent bereitzustellen, das das Element ersetzt, wenn es nicht dargestellt werden kann. Das gilt für alle Arten von Bildern, Applets, programmierte Objekte, Flash, **ASCII**-Zeichnungen, Scripts sowie Audio- und Videodateien. Wichtig darüber hinaus sind

- die Kennzeichnung von Sprachwechseln,
- korrekte Auszeichnungen für Überschriften, Tabellen, Listen und andere inhaltliche Strukturen,
- Informationen zum Kontext und zur Orientierung, beispielsweise über eine klare Gliederung mittels Zwischenüberschriften und Zusammenfassungen, sowie
- übersichtliche und schlüssige Navigationsmechanismen.

Hörbehinderungen

Menschen mit Hörbehinderungen wird im Allgemeinen unterstellt, dass sie keine Probleme bei der Verwendung des Internets und dem Surfen im World Wide Web hätten, zumindest keine, die über die Probleme von nicht eingeschränkten Menschen hinausgehen; denn schließlich sei das Internet ein schriftsprachbasiertes Medium, und Gehörlose sollten Texte doch ohne Schwierigkeiten lesen können. Diese Einschätzung ist allerdings ein großer und weit verbreiteter Irrtum.

Abbildung 2.16: Hörbehinderten wird die Benutzung des Portals »Leben mit Behinderungen in NRW« durch Gebärdensprachvideos erleichtert

Gebärdensprache

Miteinander kommunizieren Gehörlose hauptsächlich über die sogenannte Gebärdensprache, eine Zeichensprache bestehend aus Bewegung mit den Händen, Körperhaltung, Mimik und Gestik. Die Gebärdensprache ist für Gehörlose die Muttersprache. Wie bei Lautsprachen gibt es auch bei der Gebärdensprache mehrere Ausprägungen. Gebärdensprache und Lautsprachen unterscheiden sich jedoch wesentlich: Die eine verfährt visuell-motorisch, die andere akustisch-auditiv. Je gravierender der Hörschaden und je früher er eingetreten ist, desto schwieriger wird auch der Spracherwerb. Die meisten Gehörlosen sind nicht in der Lage, Sprache in dem Umfang zu lernen

wie hörende Menschen. Das betrifft auch die schriftliche Kommunikation, also das Lesen und Schreiben. Ein Vortrag von Bernd Rehling ***[Rehling 2002]*** macht das Problem deutlich:

> *»Die Sprachbarriere ist die allerhöchste Barriere für Hörgeschädigte. Und ihre Beseitigung ist meines Erachtens die personell und finanziell aufwändigste Maßnahme, wenn es um Barrierefreiheit für Behinderte im Internet geht.«*

Der Artikel »Gehörlose können doch lesen...?« ***[EfA 2003]*** formuliert es drastischer:

> *»Statistisch gesehen verlassen bis heute ca. 80 Prozent der Gehörlosen mit einer enormen sprachlichen Rückständigkeit trotz größter Mühe ihre Gehörlosenschule, praktisch als Analphabeten mit einem Schreibpotenzial von hörenden Dritt- oder Viertklässlern. [...] Angesichts der Kommunikationsbarrieren (Lautsprache/komplizierte Schriftsprache vs. Gebärdensprache/vereinfachte Schriftsprache) sind gehörlose, hör- und lernbehinderte Interessenten mit ihrer rudimentären schrift- sowie lautsprachlichen Kompetenz und blinde sowie sehbehinderte Menschen aufgrund der schriftsprachlichen Kommunikationsbarrieren vom Teilhaben an der Informationsgesellschaft ganz ausgeschlossen.«*

Leichte Sprache

Wollen Sie gehörlose Menschen nicht vollkommen von Ihren Websites ausschließen, müssen Sie Texte klar strukturieren, sehr einfach, am besten in sogenannter leichter Sprache, formulieren und durch visuell wahrnehmbare Inhalte ergänzen. Der Begriff **leichte Sprache** beschreibt eine sprachliche Ausdrucksweise, die von fast allen Menschen, die diese Sprache beherrschen, ohne Probleme verstanden wird. Besondere Merkmale sind kurze Sätze, die nur eine Aussage enthalten, Vermeidung von abstrakten Begriffen, Fach- und Fremdwörtern sowie übersichtliche Textgestaltung, illustriert mit Bildern oder Filmen.

Am besten wird leichte Sprache durch ein Beispiel verdeutlicht. Auf der Website des Rathauses Bremen[34] gibt es Beschreibungen des Rathauses in normaler und in leichter Sprache. Es ist interessant, beide Versionen miteinander zu vergleichen. Lesen Sie zunächst den normalen Text:

> *Das viel besuchte Bremer Rathaus wurde in den Jahren 1405–1408 erbaut. Es ist das einzige europäische Rathaus des Spätmittelalters, das nie zerstört wurde. Seit seiner Errichtung wurde das Rathaus kontinuierlich instand gesetzt und gewartet. (...) Dem Bremer Rathaus wird bescheinigt, dass es vom ersten gotischen Bau – einem so genannten Saalgeschossbau – Anfang des 15. Jahrhunderts über die umfangreiche Restaurierung zwei Jahrhunderte später bis hin zum Anbau des neuen Rathauses zu Beginn des 20. Jahrhunderts seine Authentizität bewahrt hat. Zahlreiche deutsche Rathäuser wurden während des 2. Weltkrieges zerstört, an vielen wurden Änderungen vorgenommen. Das Bremer Gebäude jedoch ist in seinem ursprünglichen Zustand vollständig erhalten geblieben.*

34 *http://rathaus.bremen.de*

Bis heute bewahrt blieb auch die Funktion der beiden übereinanderliegenden Rathaushallen. Die obere Etage für repräsentative Zwecke, die untere für die Nutzung durch das "Marktvolk", heute für alle Bürger. So kam die Bevölkerung in Kontakt mit den Regierenden – was bis heute so ist.

Nun die Übersetzung in leichte Sprache:

Das Bremer Rathaus ist mehr als 600 Jahre alt.
Besonders am Rathaus ist, dass es noch fast so aussieht wie früher.
Es wurde zum Beispiel im Krieg nie zerstört.
Und wenn etwas repariert wurde, wurde darauf geachtet,
dass das Rathaus trotzdem so schön bleibt wie früher.
Besonders am Rathaus ist auch, dass man dort auch noch dasselbe macht wie früher.
In der oberen Rathaushalle treffen sich wichtige Menschen. Zum Beispiel Politiker.
In der unteren Rathaushalle treffen sich die Bewohner und Besucher von Bremen.

Im Vergleich zum Originaltext fehlen in der Übersetzung zwar einige Details. Die wesentlichen Aussagen sind jedoch erhalten und kommen dadurch, dass sie klar und prägnant dargeboten werden, wesentlich besser zur Geltung. Ich bin mir sicher, dass Sie sich noch in einigen Wochen daran erinnern werden, dass das Bremer Rathaus mehr als 600 Jahre alt ist und nie zerstört wurde, wahrscheinlich aber vergessen werden, dass es einen gotischen Saalgeschossbau gibt.

Verständliche Texte schreiben

Es wäre eine unrealistische und nicht zweckmäßige Forderung, alle Websites in leichte Sprache zu übersetzen. Aber es ist sinnvoll, Websites sprachlich zu vereinfachen. Folgende Maßnahmen sind einfach umzusetzen und helfen dabei, die Verständlichkeit eines Texts zu erhöhen:

- Schreiben Sie in einfacher, klarer Sprache und formulieren Sie sachlich, direkt und faktenorientiert. Nutzer überfliegen Webseiten eher, als dass sie sie Zeile für Zeile lesen, und zudem ist die Lesegeschwindigkeit höher als bei herkömmlichen Medien.
- Verzichten Sie auf Ironie und Sprachwitz; im Web bleiben eher Aussagen hängen als die feinen Nuancen zwischen den Zeilen.
- Verzichten Sie auf Superlative und Übertreibungen.
- Erläutern Sie unbekannte Fremdwörter und Abkürzungen.
- Verzichten Sie auf Füllwörter wie eigentlich, halt, nun, an sich, ziemlich, einfach oder irgendwie. Sie blähen einen Text auf und verwässern die Aussagen.
- Formulieren Sie aktiv. Schreiben Sie nicht »daher muss dafür gesorgt werden«, sondern besser »wir sorgen dafür«. Das hat mehrere Vorteile: Sie vereinfachen den Satzbau dadurch, dass das Verb weiter vorn steht. Und Sie erhöhen die Verständlichkeit Ihrer Sätze, da Sie nicht nur schreiben, was zu tun ist, sondern auch, wer dies tun muss.

- Das Wichtigste muss bereits am Anfang des Texts stehen, gegebenenfalls in einer Zusammenfassung. Direkt zu Beginn muss klar werden: Was handelt? Was passiert? Wo und wann wird es geschehen?
- Lockern Sie Texte durch Absätze, Zwischenüberschriften und Aufzählungen auf.
- Heben Sie das Wichtigste durch Fettdruck hervor.

Kognitive, Lern- und Sprachbehinderungen

Zu dem Bereich der kognitiven sowie der Lern- und Sprachbehinderungen zählen

- Beeinträchtigungen der Intelligenz oder des Denkens einschließlich Demenz,
- Beeinträchtigungen des Gedächtnisses einschließlich Vergesslichkeit, Amnesie, Erinnerungsillusionen,
- intellektuelle Beeinträchtigungen,
- die Unfähigkeit, Sprachsymbole zu deuten und/oder zu formulieren, sowie
- Lernbehinderungen in den Bereichen gesprochene und geschriebene Sprache, Rechenschwäche, logisches Denken.

Menschen mit kognitiven Behinderungen haben meist Probleme, lange und umständlich formulierte Texte mit schwierigen Schachtelsätzen und Fremdwörtern sowie komplexe Navigationen zu verstehen. Deswegen ist es sinnvoll, Webseiten in einfacher Sprache zu verfassen oder – falls das möglich ist – Übersetzungen in leichter Sprache anzubieten.

Körperliche Einschränkungen

Zum Bereich der körperlichen Einschränkungen zählen

- Lähmung,
- Schwäche,
- Störungen der Steuerung,
- Störungen in der Koordination von Bewegungsabläufen,
- eingeschränkte Gelenkbewegungen,
- kleine, fehlgebildete oder fehlende Glieder sowie
- Minderwuchs.

Körperlich eingeschränkte Menschen können Standardeingabegeräte wie Tastatur oder Maus je nach Grad ihrer Behinderung nur eingeschränkt oder überhaupt nicht nutzen. Mit Geräten wie Mausersatz- oder Augensteuerungssystemen können sie ihre Einschränkungen zum Teil kompensieren. Jedoch ist die Bedienung einer Website mit derartigen Geräten nicht gerade einfach. Stellen Sie sich vor, Sie müssten Maus und Tastatur mit dem Fuß anstatt mit der Hand bedienen, dann können Sie sicher nachvollziehen, weshalb kleine klickbare Bereiche für körperlich eingeschränkte Nutzer besonders große Hürden darstellen.

2.9.3 Verordnungen und Richtlinien

In Ihrem Bestreben, zugängliche und barrierefreie Websites zu entwerfen, sind Sie nicht vollkommen auf sich allein gestellt. Sie können sich an einer überschaubaren Anzahl von Verordnungen und Richtlinien orientieren, die zum Teil aufeinander aufbauen und viele Empfehlungen und Anforderungen dazu an die Hand geben, wie Sie Websites gestalten, die zum einen ergonomisch und zum anderen für alle Menschen ohne besondere Erschwernis und grundsätzlich ohne fremde Hilfe zugänglich und nutzbar sind. Folgende Verordnungen und Richtlinien stelle ich kurz vor:

- EN ISO 9241-110: Grundsätze der Dialoggestaltung,
- die Zugänglichkeitsrichtlinien für Webinhalte (*Web Content Accessibility Guidelines*),
- das Behindertengleichstellungsgesetz sowie die Verordnung zur Schaffung barrierefreier Informationstechnik nach dem Behindertengleichstellungsgesetz.

Grundsätze der Dialoggestaltung

DEFINITION

*Die Norm **EN ISO 9241** (»Ergonomie der Mensch-System-Interaktion«) ist ein internationaler Standard und unterstützt Sie bei der Leistungsbeschreibung, Konzeption, Gestaltung und Bewertung von Bildschirmarbeitsplätzen. Ziel der Richtlinie ist es, gesundheitliche Schäden beim Arbeiten am Bildschirm zu vermeiden und dem Benutzer die Ausführung seiner Aufgaben zu erleichtern. Sie definiert Mindestanforderungen für die ergonomische Gestaltung von Benutzerschnittstellen und Dialogsystemen – also auch von Webseiten.*

Teil 110 (»Grundsätze der Dialoggestaltung«) ***[EN ISO 9241-110 2006]*** beschreibt Grundsätze für die Gestaltung und Bewertung einer Schnittstelle zwischen Benutzer und System. Von diesen Grundsätzen profitiert der Nutzer,

der – in Ihrem Fall – die Website bedient, da deren Anwendung zu Benutzerschnittstellen führen sollte, die gebrauchstauglich und konsistent sind und somit eine hohe Produktivität ermöglichen.

Im Einzelnen stellt die Norm folgende Grundsätze auf:

- **Aufgabenangemessenheit**: Ein Dialog ist aufgabenangemessen, wenn er den Benutzer unterstützt, seine Arbeitsaufgabe effektiv und effizient zu erledigen.
- **Selbstbeschreibungsfähigkeit**: Ein Dialog ist in dem Maße selbstbeschreibungsfähig, in dem für den Benutzer zu jeder Zeit offensichtlich ist, in welchem Dialog und an welcher Stelle im Dialog sie sich befinden, welche Handlungen unternommen werden können und wie diese ausgeführt werden können.
- **Steuerbarkeit**: Ein Dialog ist steuerbar, wenn der Benutzer in der Lage ist, den Dialogablauf zu starten sowie seine Richtung und Geschwindigkeit zu beeinflussen, bis das Ziel erreicht ist.
- **Erwartungskonformität**: Ein Dialog ist erwartungskonform, wenn er konsistent ist und den Merkmalen des Benutzers entspricht, z. B. seinen Kenntnissen aus dem Arbeitsgebiet, seiner Ausbildung und seiner Erfahrung sowie allgemein anerkannten Konventionen.
- **Fehlertoleranz**: Ein Dialog ist fehlertolerant, wenn das beabsichtigte Arbeitsergebnis trotz erkennbar fehlerhafter Eingaben entweder mit keinem oder mit minimalem Korrekturaufwand seitens des Benutzers erreicht werden kann.
- **Individualisierbarkeit**: Ein Dialog ist individualisierbar, wenn das Dialogsystem Anpassungen an die Erfordernisse der Arbeitsaufgabe sowie an die individuellen Fähigkeiten und Vorlieben des Benutzers zulässt.
- **Lernförderlichkeit**: Ein Dialog ist lernförderlich, wenn er den Benutzer beim Erlernen des Dialogsystems unterstützt und anleitet.

Zugänglichkeitsrichtlinien für Webinhalte

DEFINITION

*Die **Web Accessibility Initiative (WAI)** ist der Bereich des W3C, der für die Berücksichtigung von Zugänglichkeitsaspekten in den W3C-Empfehlungen zuständig ist. Die WAI bringt Unternehmen, Behindertenorganisationen, Wissenschaft und Regierungen in einem Prozess zusammen, um Anforderungen an die Barrierefreiheit zu erarbeiten.*

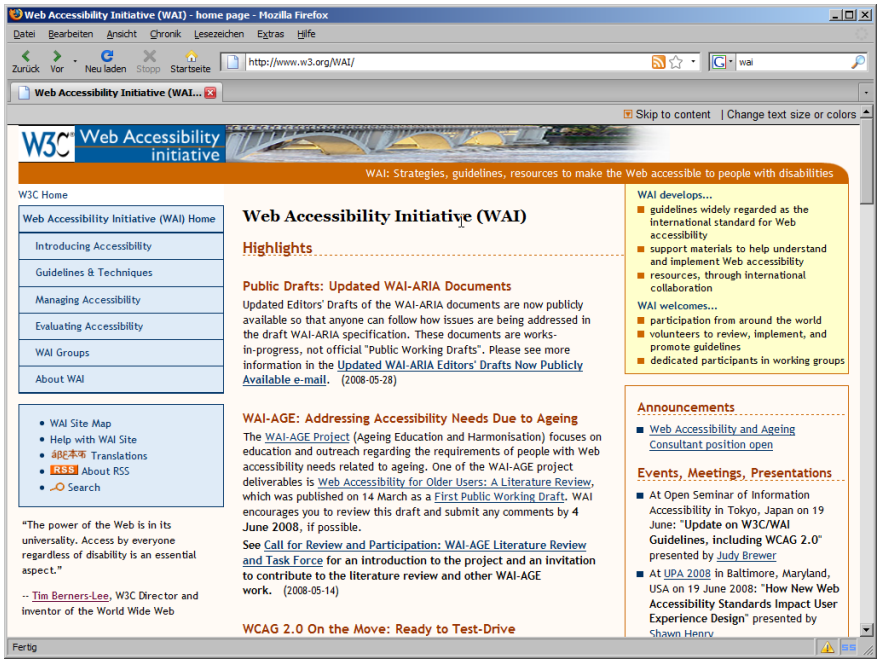


Abbildung 2.17: Screenshot der WAI-Homepage

Motto der WAI

Während EN ISO 9241-10 Grundsätze aufstellt, die die Ergonomie und Benutzbarkeit von Dialogsystemen erhöhen, hat die WAI sich die Förderung der Zugänglichkeit von Webseiten zum Ziel gesetzt. Auf ihrer Homepage steht ein Zitat von Tim Berners-Lee, das das Motto der WAI umschreibt:

> *»The power of the Web is in its universality. Access by everyone regardless of disability is an essential aspect.«*

Die WAI hat eine Vielzahl von Empfehlungen erarbeitet, die die Zugänglichkeit des Webs insbesondere für Menschen mit Behinderungen gewährleisten sollen, darunter unter anderem:

Empfehlungen

- Die *Web Content Accessibility Guidelines* (**WCAG**) erklären detailliert, wie Websites barrierefrei gestaltet werden.
- Die *Authoring Tool Accessibility Guidelines* (ATAG) erläutern, was Software leisten muss, um barrierefreie Webseiten zu produzieren.
- Die User Agent *Accessibility Guidelines* (UAAG) beschreiben, welche Anforderungen durch Browser, Plug-ins und Computerhilfsmittel erfüllt sein müssen, um barrierefrei gestaltete Inhalte auch für alle zugänglich wiedergeben zu können.

- Die *XML Accessibility Guidelines* (XAG) erklären, wie Entwickler XML-basierter Anwendungen Barrierefreiheit unterstützen können.

WCAG 1

Die für Webautoren wichtigste Empfehlung sind die im Mai 1999 veröffentlichten »Zugänglichkeitsrichtlinien für Webinhalte« ***[REC-WCAG10 1999]***. Das Dokument enthält im Wesentlichen 14 Richtlinien mit dazugehörigen Checkpunkten, die näher ausführen, wie diese Richtlinien einzuhalten sind. Jedem Checkpunkt wurde eine Prioritätsstufe zugeordnet, entweder 1 (besonders wichtig), 2 (wichtig) oder 3 (weniger wichtig). Checkpunkte der Priorität 1 sind grundlegende Anforderungen, damit die Website allen Menschen zugänglich ist, Checkpunkte der Priorität 2 beseitigen signifikante Hindernisse, und Checkpunkte der Priorität 3 schließlich erleichtern den Zugriff.

Ein separates Dokument, genannt »Techniques for Web Content Accessibility Guidelines 1.0« ***[Note-WCAG10-TECHS 2000]*** erläutert, wie die Checkpunkte der Zugänglichkeitsrichtlinien zu implementieren sind. Es enthält unter anderem Beispiele in HTML, CSS, **SMIL** und **MathML** sowie Techniken zur Validierung und zum Testen.

Die einzelnen Richtlinien und Checkpunkte können Sie bei Interesse in den Empfehlungen nachlesen. Sie bilden die Grundlage für die Anforderungen und Bedingungen der Anlage 1 der Barrierefreien Informationstechnik-Verordnung (**BITV**) und finden sich dort in nahezu identischer Formulierung wieder. Mehr dazu im nächsten Abschnitt.

WCAG 2

Die WCAG 1 sprechen zwar viele Aspekte der Barrierefreiheit an, beschreiben sie aber nicht ausreichend ausführlich, darüber hinaus sind viele Kriterien nicht objektiv prüfbar. Daher sind die WCAG 1 seit ihrer ursprünglichen Veröffentlichung längst überarbeitet worden. In der zweiten Fassung, die derzeit noch als Candidate Recommendation vorliegt ***[CR-WCAG20 2008]***, wird ein grundlegend anderer Ansatz verfolgt. Nicht die korrekte Verwendung von XHTML und CSS steht im Vordergrund, sondern es werden Prinzipien und Richtlinien aufgestellt, die unabhängig von heutigen und zukünftigen Technologien gelten. Die Checkpunkte sind allgemeiner formuliert, sodass sie sich auf beliebige Technologien anwenden lassen. Es werden vier grundlegende Prinzipien aufgestellt, in die die einzelnen Richtlinien unterteilt werden:

Richtlinien

1. **Inhalte müssen wahrnehmbar sein.**

   Alle Funktionen und Informationen auf einer Website müssen in einer Form präsentiert werden, die von jedem Benutzer mit dem von ihm gewählten Benutzerprogramm zugänglich und benutzbar ist.

2. **Schnittstellenelemente in den Inhalten müssen bedienbar sein.**

   Personen mit Behinderungen, die Schwierigkeiten mit der Augen-Hand-Koordination haben oder aus anderen Gründen kein Zeigegerät wie eine Maus bedienen können, zum Beispiel weil sie den Mauszeiger nicht erkennen können, sind auf Geräteunabhängigkeit angewiesen. So müssen Webseiten auch mit der Tastatur oder anderen Eingabegeräten wie zum Beispiel einer Spracheingabe gesteuert werden können.

3. **Inhalte und Kontrollelemente müssen verständlich sein.**

   Auch Menschen mit kognitiven Behinderungen oder Kommunikationsschwächen müssen sich auf Webseiten zurechtfinden und deren Navigationsmechanismen und Inhalte verstehen können.

4. **Inhalte müssen ausreichend robust sein, damit sie in derzeitigen und zukünftigen Technologien funktionieren.**

   Damit wird vor allem der Einsatz verfügbarer und zugänglicher Standards angesprochen, auf die sich die verarbeitende Software verlassen muss, sodass systemübergreifend identische Zugänglichkeit sichergestellt werden kann.

Mitarbeit

Es lohnt sich, die Arbeit der WAI in Zukunft weiter zu verfolgen und kritisch zu begleiten. Über die beiden Mailinglisten *w3c-wai-ig@w3.org* und *w3c-wai-gl@w3.org* können Sie sich aktiv beteiligen.

Barrierefreie Informationstechnik-Verordnung (BITV)

DEFINITION

*Die **Barrierefreie Informationstechnik-Verordnung [BITV 2002]** stellt Anforderungen und Bedingungen auf, die behinderten Menschen, denen die Nutzung der Informationstechnik üblicherweise nur eingeschränkt möglich ist, den Zugang dazu eröffnen. Die BITV ist die Grundlage des barrierefreien Webdesigns in Deutschland und die verbindliche Rechtsverordnung zu § 11 Behindertengleichstellungsgesetz **[BGG 2002]**.*

Ziel des BGG

§ 1 BGG definiert das Ziel des Behindertengleichstellungsgesetzes:

> *»Ziel dieses Gesetzes ist es, die Benachteiligung von behinderten Menschen zu beseitigen und zu verhindern sowie die gleichberechtigte Teilhabe von behinderten Menschen am Leben in der Gesellschaft zu gewährleisten und ihnen eine selbstbestimmte Lebensführung zu ermöglichen. Dabei wird besonderen Bedürfnissen Rechnung getragen.«*

§ 4 BGG definiert den Begriff Barrierefreiheit:

DEFINITION

Barrierefrei *sind bauliche und sonstige Anlagen, Verkehrsmittel, technische Gebrauchsgegenstände, Systeme der Informationsverarbeitung, akustische und visuelle Informationsquellen und Kommunikationseinrichtungen sowie andere gestaltete Lebensbereiche, wenn sie für behinderte Menschen in der allgemein üblichen Weise, ohne besondere Erschwernis und grundsätzlich ohne fremde Hilfe, zugänglich und nutzbar sind.*

Der für uns Webentwickler relevante Paragraf des Behindertengleichstellungsgesetzes ist § 11 Barrierefreie Informationstechnik:

Barrierefreie Informationstechnik

»(1) Träger öffentlicher Gewalt [...] gestalten ihre Internetauftritte und -angebote [...] nach Maßgabe der nach Satz 2 zu erlassenden Verordnung schrittweise technisch so, dass sie von behinderten Menschen grundsätzlich uneingeschränkt genutzt werden können. Das Bundesministerium des Innern bestimmt im Einvernehmen mit dem Bundesministerium für Arbeit und Sozialordnung durch Rechtsverordnung, die nicht der Zustimmung des Bundesrates bedarf, nach Maßgabe der technischen, finanziellen und verwaltungsorganisatorischen Möglichkeiten

1. *die in den Geltungsbereich der Verordnung einzubeziehenden Gruppen behinderter Menschen,*
2. *die anzuwendenden technischen Standards sowie den Zeitpunkt ihrer verbindlichen Anwendung,*
3. *die zu gestaltenden Bereiche und Arten amtlicher Informationen.*

(2) Die Bundesregierung wirkt darauf hin, dass auch gewerbsmäßige Anbieter von Internetseiten [...] ihre Produkte entsprechend den technischen Standards nach Absatz 1 gestalten.«

Entwicklung

Um die technischen Standards, von denen hier die Rede ist, sowie die einzuhaltenden Voraussetzungen für Angebote im Internet und den Zeitpunkt der Umsetzung genauer zu definieren, wurde im Juli 2002 die Verordnung zur Schaffung barrierefreier Informationstechnik nach dem Behindertengleichstellungsgesetz, kurz »Barrierefreie Informationstechnik-Verordnung«, erlassen. Sie entspricht einem Eckpunktepapier, das von Experten aus den Behindertenverbänden auf Zugänglichkeitsrichtlinien für Webinhalte 1.0 des W3C formuliert wurde. Vorausgegangen war eine umfangreiche Prüfung durch die Bundesarbeitsgemeinschaft Hilfe für Behinderte e. V., inwiefern die Anwendung der Zugänglichkeitsrichtlinien behinderten Menschen bei alltäglichen Problemen wirklich weiterhilft. Das Ergebnis der Prüfung bestätigte dies jedoch auf ganzer Linie. Die Verordnung wendet sich aufgrund ihrer Gesetzesgrundlage an die Bundesverwaltung. Das Gesetz sieht aber ausdrücklich vor, dass die Schaffung eines barrierefreien Internets auch darüber hinaus gelten sollte. Das bedeutet, dass Sie nicht verpflichtet sind, Websites nach den Anforderungen der BITV zu gestalten, solange Sie keine Websites für Träger öffentlicher Gewalt entwickeln, beispielsweise Bundes- und Landesbehörden. Dennoch sollten Sie sich an den Anforderungen der BITV orientieren.

Anforderungen und Bedingungen

Der für Webentwickler relevante Teil der BITV versteckt sich in der »Anlage (Teil 1) zu den §§ 3 und 4, Absatz 1 BITV«, die die Anforderungen und Bedingungen aufführt. Insgesamt gibt es 14 Anforderungen. Jede Anforderung wird durch eine Handvoll Bedingungen spezifiziert und näher erläutert. Die Bedingungen sind in Priorität 1 und Priorität 2 aufgeteilt. Angebote der Informationstechnik, also natürlich auch Webseiten, sind gemäß der Anlage so zu gestalten, dass alle Angebote die unter Priorität 1 aufgeführten Bedingungen zwingend erfüllen und zentrale Navigations- und Einstiegsangebote zusätzlich die unter Priorität 2 aufgeführten Bedingungen berücksichtigen.

Literatur

Einfach für Alle: Serie: BITV – Reloaded
Mittlerweile sind sämtliche in der Bundes-BITV gesetzten Fristen abgelaufen; das Web hat sich in dieser Zeit dramatisch weiterentwickelt. Grund genug, die Verordnung genau anzusehen und sich zu überlegen, was sich bewährt hat und was nicht. Tomas Caspers wirft in dieser Artikelserie einen genauen Blick auf die BITV, bewertet ihre Umsetzbarkeit und stellt leicht nachvollziehbare Beispiele von typischen Barrieren und Techniken zu deren Vermeidung vor.
URL: *http://www.einfach-fuer-alle.de/artikel/bitv-reloaded/*

Jan Eric Hellbusch: Web Content Accessibility Guidelines 1.0
Übersetzung der WCAG 1 in Form zweier Listen: Quick-Tipps und Die 66 Gebote (und zum Teil auch Verbote).
URL: *http://www.barrierefreies-webdesign.de/richtlinien/index.php#wcag1*

Jan Eric Hellbusch: Die vier Prinzipien der Web Content Accessibility Guidelines 2.0
Auf dieser Seite stellt der Autor die vier Prinzipien der WCAG 2 dar und ordnet sie den Bedingungen aus der BITV zu. Die Ausführungen sind angelehnt an den WCAG 2-Arbeitsentwurf vom 19. November 2004.
URL: *http://www.barrierefreies-webdesign.de/wcag2/*

Thomas Caspers: Letzte Runde für WCAG 2
Ein Überblick über Struktur und Inhalte der WCAG 2.
URL: *http://www.einfach-fuer-alle.de/artikel/wcag2-last-call/*

Derek Featherstone: Kinder und Barrierefreiheit: Es ist wichtig
Ein halb ernsthafter, halb scherzhafter Blick darauf, warum Kinder teilweise wie eine Art vorübergehende Behinderung sind. Übersetzt von Thomas Caspers.
URL: *http://www.einfach-fuer-alle.de/artikel/kinder-und-barrierefreiheit/*

Tomas Caspers: Barrierefrei zum Mitnehmen
Hier finden Sie eine ganze Reihe von Vorlagen, die Sie für Ihre Arbeit einsetzen und damit einen wichtigen ersten Schritt in Richtung Barrierefreiheit machen können.
URL: *http://www.einfach-fuer-alle.de/artikel/barrierefrei-zum-mitnehmen/*

Tomas Caspers: Links zur Überprüfung Ihrer Website
Im Netz gibt es viele Angebote, die Ihnen bei der Überprüfung von Webseiten weiterhelfen. Hier eine aktuelle Übersicht.
URL: *http://www.einfach-fuer-alle.de/artikel/tools/*

2.10 Zeichenkodierung

Die Kodierung und Darstellung von Schriftzeichen mit und auf dem Computer ist ein äußerst umfangreiches, kompliziertes, aber auch spannendes Thema. Gerade Webautoren, die täglich mit Texten der weltweiten elektronischen Kommunikation umgehen, müssen die Konzepte der Zeichenkodierung zumindest in den Grundzügen verstanden haben, um funktionierende Webseiten bauen zu können. Dieser Abschnitt gibt eine zum Teil vereinfachte, aber dennoch genügend tief gehende Einführung.

2.10.1 00101010 – Am Anfang waren null und eins

Bits

Computer arbeiten nur mit Nullen und Einsen, sogenannten *Binary Digits* (**Bits**). Für den Menschen ist deren Ein- und Ausgabe mühsam. Er ist seine vertrauten Zeichen gewohnt: Buchstaben, Ziffern und Satzzeichen, aus denen dann Wörter, Sätze und Absätze entstehen.

Der Computer muss die eingegebenen Zeichen aus den unterschiedlichsten Alphabeten und Sprachen in Nullen und Einsen umwandeln. Dafür gibt es Tabellen, in denen Zeichenkodierungen beschrieben werden. Beispielsweise wird der Bit-Kombination 01000001 der Großbuchstabe »A« zugeordnet, 01000010 steht für »B«, 01000011 für »C« und so weiter. Damit können Mensch und Maschine kommunizieren.

2.10.2 Von Bits und Bytes, dezimal und hexadezimal

Bytes

Für gewöhnlich werden acht Bits zu einem Oktett (*octet*) zusammengefasst, besser bekannt als »Byte«. Ein Oktett kann 256 Zustände beschreiben, in diesem Fall 256 unterschiedliche Zeichen.

Hexadezimalsystem

Neben dem Dezimalsystem, unserem Zahlensystem zur Basis 10 mit den Ziffern 0 bis 9, müssen Webautoren das **Hexadezimalsystem** beherrschen. Dabei

handelt es sich um ein Zahlensystem zur Basis 16. Es verwendet die Ziffern 0 bis 9 und für die dezimalen Zahlen 10 bis 15 die »hexadezimalen Zahlen« A bis F. Die binäre Zahl 01000001 hat dezimal den Wert 65 und hexadezimal 41, die binäre Zahl 01011010 hat dezimal den Wert 90 und hexadezimal 5A.

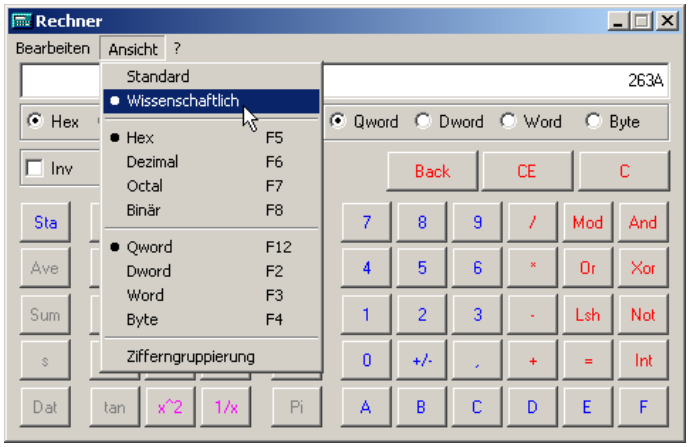


Abbildung 2.18: Windows-Rechner in der wissenschaftlichen Ansicht

Bei der Umrechnung von binär, dezimal und hexadezimal hilft der in Windows mitgelieferte Rechner in der wissenschaftlichen Ansicht oder der Rechner von Mac OS X unter Verwendung des Menüeintrags DARSTELLUNGSFORMAT.

Oftmals wird die Hexadezimalschreibweise durch das Voranstellen von `x` (in XHTML) oder `0x` (in Programmiersprachen wie C++ oder Java) kenntlich gemacht, also etwa `x41`, damit eine Verwechslung mit der Dezimal- oder anderen Schreibweisen ausgeschlossen wird.

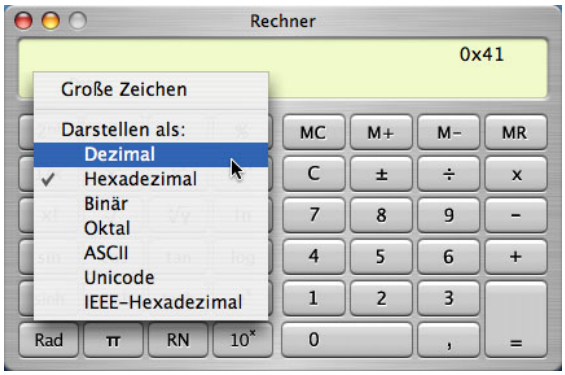


Abbildung 2.19: Rechner von Mac OS X

2.10.3 Begrifflichkeiten

DEFINITION

*Der Begriff **Zeichenvorrat** (*character repertoire*) beschreibt eine endliche, nicht leere, ungeordnete Menge aller Zeichen, die man zur Verfügung hat oder verwenden möchte.*

DEFINITION

*Der Zeichenvorrat lässt sich anordnen, die Zeichen also in eine Reihenfolge bringen und durchnummerieren. Dann spricht man von einem **Zeichensatz** (*character set*).*

DEFINITION

*Um mit Zeichen arbeiten zu können, muss der Computer die Darstellung in Bits kennen. Dafür ist die **Zeichenkodierung** (*character encoding*), oder kurz einfach Kodierung (*encoding, coding*), zuständig. Da im Zeichensatz bereits Reihenfolge und Nummerierung festgelegt wurden, braucht man also nur noch die Bit-Muster zuzuordnen.*

Einige Zeichenkodierungen können nur wenige Zeichen kodieren, andere ein paar mehr und wieder andere schließlich alle. Im Allgemeinen kodieren zwei verschiedene Zeichenkodierungen dasselbe Zeichen in unterschiedliche Byte-Sequenzen und dieselbe Byte-Sequenz in unterschiedliche Zeichen.

2.10.4 Zeichenkodierungen

US-ASCII

DEFINITION

US-ASCII ist die grundlegende und daher wohl auch am weitesten verbreitete Zeichenkodierung, jedoch auch eine der am wenigsten umfangreichen. US-ASCII wird seit Ende der 1960er verwendet und ist eine 7-Bit-Kodierung, umfasst also 128 Zeichen.

Die ersten 32 Zeichen (von Position 0 bis 31) sind **Steuerzeichen**, ebenso das letzte auf Position 127. Steuerzeichen sind zum Beispiel für die Kontrolle von Bildschirmdarstellung und Druckern zuständig oder können bestimmte Editierfunktionen auslösen.

	0	1	2	3	4	5	6	7	8	9
30				!	"	#	$	%	&	'
40	(	)	*	+	,	-	.	/	0	1
50	2	3	4	5	6	7	8	9	:	;
60	<	=	>	?	@	A	B	C	D	E
70	F	G	H	I	J	K	L	M	N	O
80	P	Q	R	S	T	U	V	W	X	Y
90	Z	[	\	]	^	_	`	a	b	c
100	d	e	f	g	h	i	j	k	l	m
110	n	o	p	q	r	s	t	u	v	w
120	x	y	z	{	|	}	~			

Tabelle 2.9: Druckbare Zeichen in US-ASCII: Position 32 (Leerzeichen) bis 126 (Tilde)

Die druckbaren US-ASCII-Zeichen beginnen an Position 32 mit dem Leerzeichen, dann folgen einige Satz- und Sonderzeichen, ab Position 48 die Ziffern beginnend mit »0«, das große »A« (65) bis »Z« (90) und das kleine »a« (97) bis »z« (122). Zwischen diesen Blöcken befinden sich immer wieder einige Satz- und Sonderzeichen. Das letzte druckbare Zeichen ist die Tilde (~) an Position 126.

Mit US-ASCII können Sie schon viele Texte elektronisch niederschreiben, vor allem englischsprachige, wenngleich hier auch nicht alle wünschenswerten Zeichen, wie typografische **Anführungszeichen** oder Gedankenstriche, zur Verfügung stehen. Für nicht englische Sprachen ist US-ASCII wenig geeignet, da zum Beispiel nicht einmal die in der deutschen Sprache wichtigen Umlaute vorhanden sind.

Die ISO-8859-Familie

Da US-ASCII für nicht englische Sprachen nur einen Teil der benötigten Zeichen enthält, wurde (unter vielen anderen) die ISO-8859-Familie standardisiert.

DEFINITION

Die ISO-8859-Familie besteht aus 15 8-Bit-Zeichenkodierungen, in denen besonders europäische Sprachen gut dargestellt werden können: ISO-8859-1 bis ISO-8859-16, wobei es ISO-8859-12 nicht gibt. Die ersten 128 Zeichen (0 bis 127) entsprechen dabei US-ASCII.

- ISO-8859-1 (Latin 1): Westlich/Westeuropäisch
- ISO-8859-2 (Latin 2): Zentraleuropäisch, Osteuropäisch

- ISO-8859-3 (Latin 3): Südeuropäisch, Maltesisch, Esperanto
- ISO-8859-4 (Latin 4): Nordeuropäisch
- ISO-8859-5: Kyrillisch, Slawisch
- ISO-8859-6: Arabisch
- ISO-8859-7: (modernes) Griechisch
- ISO-8859-8: Hebräisch
- ISO-8859-9 (Latin 5): Türkisch
- ISO-8859-10 (Latin 6): Nordisch (Sámi, Inuit, Isländisch)
- ISO-8859-11: Thai
- ISO-8859-13 (Latin 7): Baltisch (»baltischer Rand«)
- ISO-8859-14 (Latin 8): Keltisch
- ISO-8859-15 (Latin 9): ähnlich ISO-8859-1, mit acht Änderungen (beispielsweise +E+U+R+O+-Symbol)
- ISO-8859-16 (Latin 10): Albanisch, Kroatisch, Englisch, Finnisch, Französisch, Deutsch, Ungarisch, irisches Gälisch, Italienisch, Lateinisch, Polnisch, Rumänisch, Slowenisch

Alle diese Zeichensätze enthalten an Position 160 (xA0) ein geschütztes Leerzeichen, an dem im Textfluss nicht umbrochen wird. Die nachfolgenden Positionen unterscheiden sich zum Teil sehr stark. An Position 173 findet sich in der Regel der Trennungsstrich. Allgemein wird ein Zeichensatz als passend für eine Sprache benannt, wenn alle für die Sprache benötigten Buchstaben enthalten sind. Jedoch fehlen dann oftmals immer noch typografisch korrekte Satzzeichen.

Einsatz Viele Zeichenkodierungen dieser Familie eignen sich für deutschsprachige Texte, da in ihnen Umlaute und das »ß« vorhanden sind, nämlich ISO-8859-1, -2, -3, -4, -9, -10, -13, -14, -15 und -16. Benötigen Sie auch noch die in deutschen Texten seltener vorkommenden Buchstaben »À«, »à«, »É« und »é«, verkürzt sich die Liste auf -1, -3, -9, -14, -15 und -16. Durch diese Vielseitigkeit können Sie auf einer Website in der entsprechenden Kodierung zum Beispiel deutschen zusammen mit polnischem Text (ISO-8859-2, -13 oder -16) schreiben oder Deutsch und Türkisch gemeinsam (ISO-8859-9). Allerdings stoßen Sie auch schnell an die Grenzen: ISO-8859-5 enthält neben US-ASCII nur kyrillische Zeichen, -6 arabische, -7 modernes Griechisch und -8 Hebräisch. Diese können Sie weder untereinander noch mit beispielsweise Deutsch und dessen Umlauten kombinieren.

ISO-8859-1 ist die für Deutsch am weitesten verbreitete Zeichenkodierung. Sie bietet die nötigen Umlaute samt Eszett (ß) sowie die in der deutschen Sprache verwendbaren französischen Anführungszeichen (», Zeichennummer 187, und «, Zeichennummer 171). Das später eingeführte ISO-8859-15 wurde gegenüber ISO-8859-1 auf acht Positionen verändert, um unter anderem das Euro-Symbol zur Verfügung zu stellen.

Einschränkungen

Schöne **Typografie** ist mit ISO-8859-1 (oder anderen Zeichenkodierungen der Familie) jedoch nicht oder nur eingeschränkt möglich, da zwar die nötigen Buchstaben vorhanden sind, aber andere häufig genutzte (Satz-)Zeichen wie doppelte und normale Anführungszeichen (»Gänsefüßchen«, einfach und doppelt), Apostroph, Gedankenstrich und Auslassungszeichen (Ellipse) nicht existieren. Müssen oder wollen Sie in einer ISO-8859-Zeichenkodierung bleiben, müssen Sie für (schöne) Typografie auf **Zeichenreferenzen** ausweichen.

Unicode

Das **Unicode** Consortium[35] setzte sich im Jahr 1991 das Ziel, als Zeichenvorrat möglichst jedes Zeichen, das in irgendeiner Sprache verwendet wird oder wurde (und dazu noch jede Menge zusätzlicher Symbole wie zum Beispiel mathematische Zeichen, Lautschrift, geometrische Formen, Pfeile sowie Musiknotation), aufzunehmen, zu katalogisieren und in geeigneten Kodierungen zur Verfügung zu stellen, die nicht mehr 128 oder 255 mögliche Positionen und damit Zeichen enthalten, sondern anfänglich 65.536 und später dann über eine Million.

Die ersten 256 Zeichen aus Unicode sind die Zeichen aus ISO-8859-1 in der entsprechenden Reihenfolge und damit die ersten 128 Zeichen aus US-ASCII. Allerdings werden sie je nach gewählter Kodierung anders als Nullen und Einsen abgespeichert. Den gesamten Zeichenvorrat können Sie in den Code Charts[36] auf der Unicode-Website unter anderem als PDF anschauen.

Irgendwie müssen nun diese Tausende von Zeichen in Unicode dem Computer in Nullen und Einsen verständlich gemacht werden. Dafür gibt es mehrere Kodierungen, die jeweils kleine Vor- und Nachteile bieten. Für Webautoren ist vor allem UTF-8 interessant.

35 *http://www.unicode.org*
36 *http://www.unicode.org/charts*

DEFINITION

UTF-8 *(*8-bit Unicode Transformation Format*) ist eine 8-Bit-Kodierung. Besondere Bit-Kombinationen am Anfang eines Oktetts sagen dem Computer, dass eine bestimmte Anzahl nachfolgender Oktette zusammen betrachtet werden muss, um ein Zeichen ergeben zu können. Die Zeichen aus US-ASCII werden deshalb in UTF-8 in genau einem Oktett kodiert, die Tausende von nachfolgenden Zeichen in zwei, drei oder vier Oktetten.*

Deshalb ist Unicode nicht mit dem 8-bittigen ISO-8859-1 kompatibel, obwohl auf den ersten 256 Positionen die gleichen Zeichen stehen – die Positionen werden anders in Nullen und Einsen dargestellt. So bringt man alle Unicode-Zeichen in einer 8-Bit-Kodierung, eben UTF-8, unter.

Noch mehr Zeichenkodierungen

Die hier vorgestellten Zeichenkodierungen sind die in unseren Breiten am häufigsten genutzten und die für Webseiten am besten geeigneten. Es gibt Dutzende weiterer, die mehr oder weniger oft verwendet werden. Dazu gehören ältere und »unschöne« Kodierungen von Computerherstellern wie IBMs EBCDIC oder Apples MacRoman – oder auch die weit verbreiteten Zeichenkodierungen der Windows-Familie, die in Microsofts Betriebssystemen ihre Heimat haben und quasi Obermengen der ISO-8859-Zeichenkodierungen darstellen.

So enthält die bei der IANA registrierte Zeichenkodierung Windows-1252 alle Zeichen von ISO-8859-1 auf den entsprechenden Positionen, jedoch zusätzlich im Bereich 128 bis 159, der in den ISO-8859-Standards für Steuerzeichen reserviert ist, weitere Zeichen. Darunter befinden sich unter anderem auch typografische Anführungszeichen, Gedankenstriche, das Auslassungszeichen und das Euro-Symbol.

Selbst das oberflächliche Betrachten von in bestimmten Teilen der Welt weit verbreiteten Zeichenkodierungen, wie KOI8-R für Russisch beziehungsweise Shift-JIS, ISO-2022-JP oder EUC-JP für Japanisch, würde den Umfang dieser Einführung und vermutlich auch Ihr Interesse an diesem Thema sprengen. Es ist aber wohl schon durch diese kurze Einführung klar geworden, dass es zahlreiche Zeichenkodierungen gibt und die Existenz der »universellen« Zeichenkodierung UTF-8 einiges zu vereinfachen vermag.

2.10.5 Das Problem mit Zeichenkodierungen

Warum muss sich ausgerechnet ein Webautor mit diesen Dingen beschäftigen? Das Problem besteht darin, dass ein Computer nur die Nullen und Einsen, die Bits, sieht. Er kann nicht entscheiden, ob eine Bit-Kombination ein Zeichen aus ISO-8859-n, Windows-125n, MacRoman, KOI8-R oder UTF-8 beschreibt. Benutzerprogramme wie Browser oder Texteditoren müssen mitgeteilt bekommen, nach welcher Kodierung die Bits zu Zeichen aufgelöst werden sollen.

Geschieht das nicht, etwa weil der Webserver dem Browser keine oder eine falsche Kodierungsinformation mitliefert, erscheint auf dem Monitor Zeichenmüll. Ein Dokument, das ISO-8859-5-Zeichen enthält, zum Beispiel russischen Text, aber als ISO-8859-1 geöffnet wird, zeigt statt des kyrillischen großen F ein »Ä« an, da diese Zeichen auf der gleichen Position in beiden Zeichensätzen stehen. Wird ein in UTF-8 gespeicherter deutscher Text mit Umlauten als ISO-8859-1 geöffnet, werden statt eines Umlauts gleich zwei wirre Zeichen erscheinen, da in UTF-8 Umlaute mit zwei Oktetten kodiert werden, in ISO-8859-1 aber ein Zeichen regelmäßig mit genau einem.

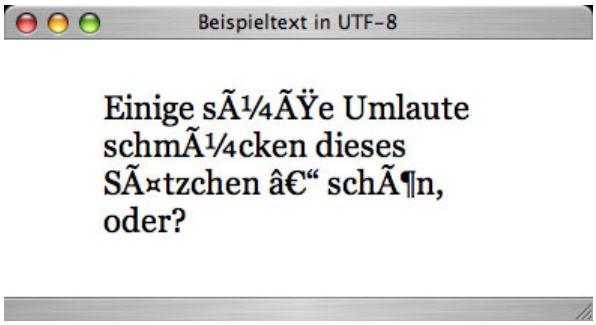


Abbildung 2.20: »Einige süße Umlaute schmücken dieses Sätzchen – schön, oder?«, in UTF-8 gespeichert, als ISO-8859-1 aufgerufen

Angabe der Zeichenkodierung

Sie müssen immer explizit angeben, vor allem beim erstmaligen Speichern einer Datei, mit welcher Kodierung Sie arbeiten. Verwenden Sie daher einen Text- oder HTML-Editor, der mit unterschiedlichen Kodierungen umzugehen versteht, aber wenigstens US-ASCII, die ISO-8859- und Windows-125n-Familien sowie UTF-8 unterstützt, da diese in unseren Gefilden die am weitesten verbreiteten Kodierungen sind. Abbildung 2.21 zeigt, wie Sie Dokumente im Editor UltraEdit korrekt als UTF-8 (ohne BOM) abspeichern.

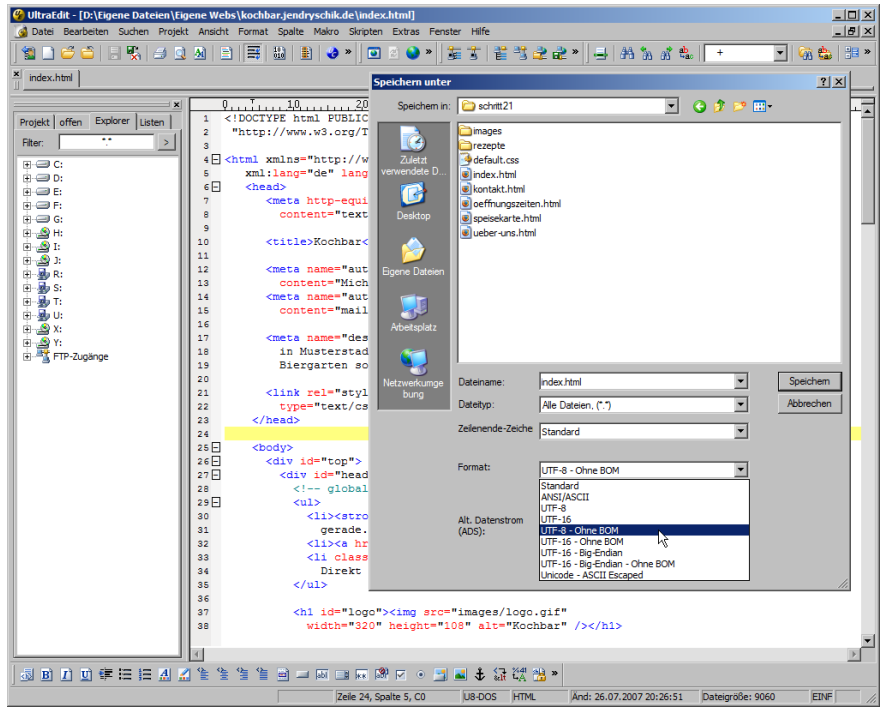


Abbildung 2.21: Dokumente mit UltraEdit als UTF-8 abspeichern

Auch bei der Auslieferung von Dokumenten durch den Webserver muss die richtige Kodierung bekannt sein. Wie Sie die Zeichenkodierung Ihres Webservers konfigurieren, können Sie in dessen Dokumentation nachlesen. Weitere Informationen bietet auch der Artikel *[Dürst 2007]*. Beim **Apache HTTP Server** beispielsweise geht dies sehr einfach über eine Zugriffskontrolldatei (`.htaccess`-Datei) und dem Eintrag der verwendeten Zeichenkodierung bei der Direktive `AddCharset`:

```
AddCharset UTF-8 .html .css
AddCharset ISO-8859-1 .txt
```

.htaccess

Zugriffskontrolldateien werden auch **.htaccess-Dateien** genannt, denn es handelt sich dabei standardmäßig um Textdateien ohne Dateinamen mit der Dateiendung `htaccess`. Solche Dateien können Sie unter Windows nur über Umwege anlegen. Öffnen Sie einen Editor Ihrer Wahl (Notepad tut es auch) und speichern Sie ein leeres Textdokument einfach als `.htaccess` ab. Achten Sie darauf, dass der Editor nicht automatisch eine Dateiendung (z. B. `txt`) anhängt. Im unwahrscheinlichen Fall, dass sowohl die Anzeige der Endungen bei bekannten Dateitypen ausgeschaltet ist als auch Dateien vom Typ `.htaccess` mit einem Programm verknüpft sind, blicken Sie im Windows Explorer ins Leere, aber keine Sorge, die Datei ist natürlich vorhanden und kann problemlos per FTP auf den Webserver hochgeladen werden.

Bei `.htaccess`-Dateien ist es wichtig, dass Sie auf den korrekten Zeilentrenner achten. Da der Apache zumeist auf einem Unix-System läuft, ist der korrekte Zeilentrenner das Zeichen Zeilenvorschub (U+000A). Wenn Sie unter Windows oder Mac arbeiten, sollten Sie einen Editor verwenden, der die Wahl des Zeilentrenners erlaubt, oder Sie übertragen die Datei im FTP-Programm im ASCII-Modus, wodurch die Zeilentrenner automatisch richtig konvertiert werden. Übertragen Sie die Datei jedoch mit einem falschen Zeilentrenner, kann das unter Umständen die ganze Website außer Funktion setzen.

- Apache HTTP Server Version 1.3 – *.htaccess files*

  Englischsprachige Einführung in die Zugriffskontrolldatei `.htaccess` für Apache 1.3.

  URL: *http://httpd.apache.org/docs/1.3/howto/htaccess.html*

- Apache HTTP Server Version 2.0 – Apache Tutorial: *.htaccess files*

  Englischsprachige Einführung in die Zugriffskontrolldatei `.htaccess` für Apache 2.0.

  URL: *http://httpd.apache.org/docs/2.0/howto/htaccess.html*

2.10.6 Zeichendarstellung

Ob gewisse Zeichen überhaupt angezeigt werden können, hängt vom Betriebssystem, der Anwendungssoftware (Webbrowser, Texteditor) und den installierten Schriften ab. Die auf dem lateinischen Alphabet basierenden Zeichenkodierungen wie US-ASCII oder ISO-8859-1 lassen sich im Allgemeinen problemlos auf jedem Rechner verwenden. Auf einigen Systemen mag es Schwierigkeiten mit komplexeren Alphabeten geben; die umfangreichen japa-

nischen oder chinesischen Schriftzeichen sind dafür Kandidaten. Aber in der Regel sind bei der Zielgruppe dieser Schriftzeichen – eben Personen, die dieser Sprachen mächtig sind – Betriebssysteme entsprechend ausgestattet und eingestellt. Sollte ein Zeichen nicht angezeigt werden können, wird stattdessen zumeist ein Fragezeichen oder ein Rechteck verwendet.

Intelligente, halbwegs aktuelle Webbrowser holen sich ein Schriftzeichen, das in der gerade verwendeten Schriftdatei nicht vorhanden ist, aus irgendeiner anderen Schriftdatei auf dem Rechner, sofern das Zeichen dort vorkommt. Sie müssen also keine Angst haben, wenn in der von Ihnen vorgegebenen Schriftfamilie ein Zeichen nicht verfügbar sein sollte, solange es in einer anderen auf allen Rechnern standardmäßig installierten Schriftdatei aufzufinden ist.

Leider finden sich unter den vielen Tausenden von Zeichen aus Unicode immer welche, besonders Nicht-Schriftzeichen, die nicht durch die diversen mitgelieferten Schriften eines Betriebssystems abgedeckt sind. Diese Symbole sind aber meistens recht selten und für das Verständnis eines normalen Texts nicht notwendig (bei zum Beispiel mathematischen Texten mag das anders aussehen, aber dafür ist XHTML auch keine geeignete Darstellungsform). Manche Browser oder Betriebssysteme fordern den Benutzer in solchen Fällen auf, entsprechende Schriften nachzuinstallieren. Hier hilft nur, wie überall in der Webentwicklung, ausführliches Testen auf unterschiedlichen Standardinstallationen – Betriebssystemen wie auch Browsern. Sollte der Test auf den verbreiteten (Ziel-)Systemen negativ ausfallen, sollten Sie Alternativen für das benötigte Zeichen ins Auge fassen: ein anderes passendes Zeichen aus Unicode oder eine Grafik, was jedoch andere Probleme mit sich bringen könnte (bei der Grafik zum Beispiel fehlende Skalierbarkeit oder möglicherweise andere Schrift).

2.10.7 Zeicheneingabe

Wie bekommen Sie nun die Zeichen in den Computer? Für die gewöhnlichen auf der Tastatur befindlichen Zeichen ist das kein Problem: einfach die entsprechende Taste drücken. Eventuell noch die ⇧-Taste dazu. Für normale Texte reicht das erst einmal. Wollen Sie aber eine Selbstverständlichkeit wie ein Anführungszeichen oder ein Apostroph eingeben, dann gibt es schon ein Problem. Zwar existieren die generischen US-ASCII-Zeichen (" und '), aber schön sind die nicht[37]. Textverarbeitungsprogramme wie Microsoft Word oder OpenOffice.org Writer wandeln diese US-ASCII-Zeichen nach Eingabe in ansehnliche sogenannte *Smart Quotes* um, ein Texteditor in der Regel aber

37 Siehe auch das folgende Kapitel über Typografie ab Seite 119.

nicht. Dennoch können Sie auch diese Zeichen mehr oder weniger einfach eingeben. Ein Mac OS X-Rechner macht es dabei leichter als ein Windows-System. Tabelle 2.10 zeigt, welche Tasten Sie unter Windows und Mac OS X drücken müssen, um ein bestimmtes Zeichen zu erhalten.

Zeichenname	**Zeichen**	**Tastenkombination**	
		Windows	**Mac OS X**
Apostroph	'	Alt + 0 1 4 6	alt shift #
doppeltes Anführungszeichen unten	„	Alt + 0 1 3 2	alt ^
doppeltes Anführungszeichen oben	“	Alt + 0 1 4 7	alt 2
einfaches Anführungszeichen unten	‚	Alt + 0 1 3 0	alt s
einfaches Anführungszeichen oben	‘	Alt + 0 1 4 5	alt #
französisches Anführungszeichen Anfang	»	Alt + 0 1 8 7	alt shift q
französisches Anführungszeichen Ende	«	Alt + 0 1 7 1	alt q
einfaches französisches Anführungszeichen nach rechts	›	Alt + 0 1 5 5	alt shift n
einfaches französisches Anführungszeichen nach links	‹	Alt + 0 1 3 9	alt shift b
Gedankenstrich	–	Alt + 0 1 5 0	alt -
geschütztes Leerzeichen		Alt + 0 1 6 0	alt Leertaste

Tabelle 2.10: Tastenkombinationen zur Eingabe typografischer Zeichen

Die Tastenkombinationen von Mac OS X sind nach kurzer Eingewöhnungszeit gut zu merken. Es sind weniger Tasten zu drücken, und oft deutet auch die eigentliche Tastenbeschriftung schon die möglichen Zeichen an. Unter Windows ist es nicht ganz so einfach; dort müssen Sie die Alt-Taste drücken und währenddessen Ziffernkombinationen auf dem Nummernblock eingeben. Aber es gibt auch Software, mit der Sie unter Windows eigene Tastenkombinationen definieren können – zum Beispiel AllChars[38] von Jeroen Laarhoven. Damit müssen Sie sich keine Zahlen mehr merken, um diese dann umständlich einzugeben.

38 *http://allchars.zwolnet.com*

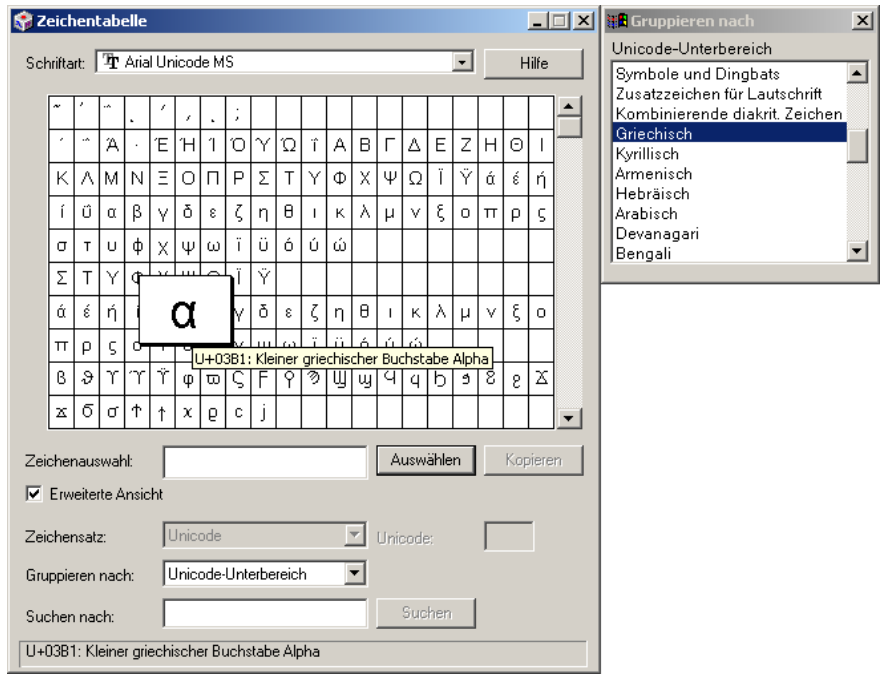


Abbildung 2.22: Zeichentabelle von Windows 2000, hier die Gruppe für Griechisch

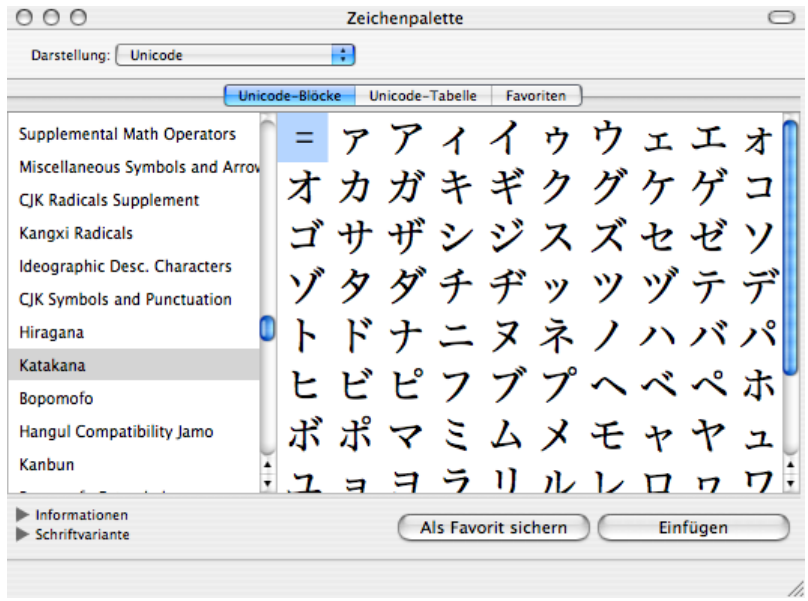


Abbildung 2.23: Zeichenpalette von Mac OS X, hier der Katakana-Block

Das betraf jetzt allerdings nur Satzzeichen für deutschsprachige Texte. Es gibt Situationen, in denen Sie mehr Zeichen benötigen. Dazu können Sie auf die Zeichentabelle (Windows) beziehungsweise Zeichenpalette (Mac OS X) zurückgreifen.

Zeichentabelle, Zeichenpalette

Die Windows-Zeichentabelle finden Sie unter START > PROGRAMME > ZUBEHÖR > SYSTEMPROGRAMME (sofern sie installiert wurde), die Mac OS X-Zeichenpalette unter PROGRAMME > SCHRIFTSAMMLUNG > BEARBEITEN > SONDERZEICHEN (seit 10.4 Tiger in fast allen Programmen unter BEARBEITEN > SONDERZEICHEN) beziehungsweise unter dem Flaggensymbol in der Menüleiste, wenn Sie die folgenden Einstellungen vornehmen:

Tastaturübersicht

- Unter Mac OS X 10.3 Panther können Sie unter SYSTEMEINSTELLUNGEN > LANDESEINSTELLUNGEN > TASTATURMENÜ eine Tastaturübersicht mit der Möglichkeit zum Wechseln der Tastenbelegungen in die Menüleiste einblenden. (In früheren Mac OS X-Versionen fand sich eine Tastaturübersicht im Ordner DIENSTPROGRAMME des Programmordners.) In der Tastaturübersicht sehen Sie die gegenwärtige Tastenbelegung – drücken Sie auch die Tasten ⇧ und alt, um weitere Belegungen zu sehen. Stellen Sie einfach mal andere Sprachen wie US-Amerikanisch, Französisch, Russisch, Türkisch oder Thailändisch ein, um andere Tastenbelegungen zu sehen. Sie können dann über Ihre Tastatur etwas mit der angezeigten Belegung eingeben. Es wird ein wenig ungewohnt sein, für die Eingabe kurzer Texte ist es jedoch durchaus geeignet. Komfortabler wird es nur noch, wenn Sie sich eine Tastatur mit der entsprechenden Belegung kaufen.
- In Windows können Sie Einstellungen zur Tastaturbelegung über START > SYSTEMEINSTELLUNGEN > LÄNDEREINSTELLUNGEN im Register unter EINGABE vornehmen. Sie können mehrere Tastaturlayouts einrichten, die auf Wunsch in der Taskleiste oder mit bestimmten Tastenkombinationen aktiviert werden können.

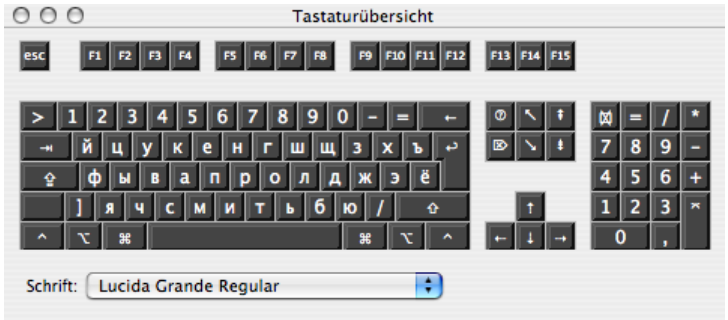


Abbildung 2.24: Tastaturübersicht von Mac OS X für eine kyrillische Tastenbelegung

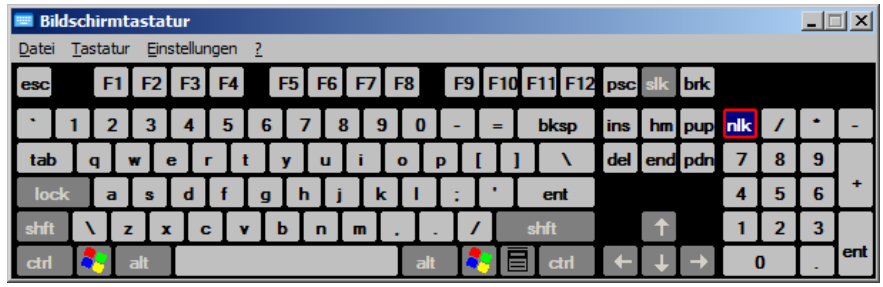


Abbildung 2.25: Windows-Bildschirmtastatur für eine englische Tastenbelegung

2.10.8 Die Wahl der »richtigen« Zeichenkodierung

Welche Zeichenkodierung sollen Sie nun verwenden, wenn Sie ein XHTML-Dokument schreiben? Die einfache Antwort: die passende.

Schreiben Sie einen englischen Text und brauchen dabei nur ab und zu typografisch passende Anführungszeichen oder ein Apostroph, können Sie US-ASCII als Zeichenkodierung wählen und diese Sonderzeichen als Zeichenreferenzen einfügen. Oder Sie verwenden UTF-8 und geben auch die Sonderzeichen direkt ein.

Wenn Sie einen Text auf Deutsch schreiben, kommen Sie um Umlaute und damit zum Beispiel ISO-8859-1 nicht herum. Auch hier können Sie für typografisch schöne Satzzeichen auf Zeichenreferenzen zurückgreifen. Oder Sie verwenden hier ebenfalls wieder direkt UTF-8.

Sollten Sie auf einer Seite verschiedene Sprachen und Schriftfamilien mischen wollen, werden Sie meistens um UTF-8 nicht herumkommen. Nur in wenigen Fällen, wenn Sie beispielsweise Deutsch und Türkisch (ISO-8859-9) oder Deutsch und Polnisch (ISO-8859-2, -13 oder -16) mischen wollen, kommen Sie ohne UTF-8 aus, sofern Sie eine Zeichenkodierung aus der ISO-8859-Familie nehmen und eventuell zusätzlich benötigte Zeichen als Zeichenreferenzen eingeben.

2.11 Typografie

DEFINITION

*Der Begriff **Typografie** entstammt den griechischen Wörtern »typos«, was so viel wie Schlag, Abdruck oder Figur bedeutet, sowie »graphein«, was sich mit malen, schreiben oder ritzen übersetzen lässt. Typografie steht für*

- *Gestalten mit Schrift,*
- *Gestalten von Schrift,*
- *Gestalten mittels Schrift, Bildern, Linien, Flächen und dem sogenannten typografischen Raum (optisch wirksamen Abstand)*

in einem visuellen Medium, also auch auf dem Bildschirm.

Ziel der Typografie

Das Ziel der Typografie ist es, die Intention eines Texts durch optimale Lesbarkeit der dargestellten Information bestmöglich zu übermitteln. Zur Intention eines Texts kann es aber auch gehören, dass er eben *nicht* gut lesbar ist; denken Sie an das Kleingedruckte bei Verträgen. Das gezielte Hervorheben oder Verbergen von Information gehört folglich ebenfalls zu den Aufgaben der Typografie. Möglichkeiten der typografischen Gestaltung sind der Einsatz unterschiedlicher Schriftarten, Schriftgrößen und Schriftschnitte sowie die Wahl der Zeilenlänge, des Zeilenabstands, des Satzspiegels innerhalb des Papierformats und vieles, vieles mehr.

HINWEIS

Regeln für gute Typografie sind immer an den historischen Kontext und die technischen Möglichkeiten gebunden. Da Typografen sich stets bemühen, Normen für die gute Gestaltung von Schriftwerken aufzustellen und bestehende Normen verändern, unterliegt die Typografie einem steten historischen Wandel. Im Laufe der Zeit haben sich viele Konventionen gebildet und wieder verändert. Aus diesem Grund hat jedes Land seine eigenen Regeln, die sich zum Teil deutlich voneinander unterscheiden. Im Folgenden orientiere ich mich an den durch den Duden definierten Regeln.

Typografie ist seit Gutenberg Teil des Fachwissens der Drucker und Setzer. Heute gehört sie auch zur Ausbildung von Grafikern und Mediengestaltern – und Webautoren sollten ebenfalls über ein gewisses typografisches Basiswissen verfügen. Verstehen Sie es als Teil Ihrer persönlichen fachlichen Qualifikation, Texte richtig setzen und typografisch korrekt auf dem Bildschirm umsetzen zu können.

Die folgenden Abschnitte sind alle analog aufgebaut. Sie beginnen mit einer Tabelle, die die einzelnen Zeichen mit Namen, Dezimal- und Hexadezimalwert sowie benannter Zeichenreferenz aufführt. Ein wenig weiter unten im

Buch in Kapitel 4.1.4 erkläre ich die Bedeutung dieser Angaben. Anschließend folgen Erläuterungen und Empfehlungen zur typografisch korrekten Anwendung. Bitte haben Sie Verständnis dafür, dass ich nur eine Auswahl von Zeichen präsentieren kann. Alle Unicode-Zeichen finden Sie auf der *Unicode Home Page*[39]. Die *HTML 4 character entity references*[40] listen alle benannten Zeichenreferenzen auf. Sollten Sie ein Zeichen in meiner Auflistung vermissen, schauen Sie am besten dort nach.

Als weiterführende Lektüre lege ich Ihnen Christoph Päpers sehr lesenswerten Artikel »Typographie für Webautoren« ***[Päper 2005]*** nahe, von dem ich einige Gedanken übernommen habe. Er ergänzt folgende Ausführungen hervorragend.

2.11.1 Leerzeichen und Festabstände

Zeichenname	**Zeichenreferenz**		
	dezimal	**hexadezimal**	**benannt**
Leerzeichen			
kurzes Leerzeichen			
langes Leerzeichen			
geschütztes Leerzeichen			
schmales geschütztes Leerzeichen			

Tabelle 2.11: Leerzeichen und Festabstände

DEFINITION

*Das **Leerzeichen** (auch: Leerschritt, Leerschlag, space, blank) dient der Abgrenzung von Wörtern innerhalb eines fortlaufenden Texts.*

Sie erhalten ein Leerzeichen durch Betätigen der `Leertaste`, der längsten Taste unten auf der normalen Computertastatur.

DEFINITION

***Festabstände** sind – auch bei **Blocksatz** – konstant breite, meist kleinere Zwischenräume zwischen Zeichen. Sie verbinden Zusammengehöriges oder gliedern Unübersichtliches und dienen so der besseren Ästhetik und Lesbarkeit von Texten.*

39 *http://www.unicode.org/*
40 *http://www.w3.org/TR/REC-html40/sgml/entities.html*

Verwenden Sie Festabstände beispielsweise bei Abkürzungen, beim Datum, bei der Gliederung von Nummern, bei Paragrafenzeichen, Rechenzeichen und Zahlen. Es gibt mehrere Arten von Festabständen, die sich in der Breite zum Teil deutlich voneinander unterscheiden. Die wichtigsten sind

- das kurze Leerzeichen (*en space*), das so breit ist wie der kurze Gedankenstrich (siehe unten),
- das lange Leerzeichen (*em space*), das der Breite des langen Gedankenstrichs (siehe unten) entspricht,
- das einfache geschützte Leerzeichen (*non-breaking space*), das in der Breite dem einfachen Leerzeichen entspricht, sowie
- das schmale geschützte Leerzeichen (*thin space*), etwa ein Fünftel so breit wie das lange geschützte Leerzeichen.

Bei den geschützten Leerzeichen wird nicht umbrochen – daher der Name.

Browser-kompatibilität

Das Leerzeichen sowie das (normale) geschützte Leerzeichen werden von jedem Browser und in jedem Textverarbeitungsprogramm interpretiert. Bei den anderen drei Zeichen können Sie sich leider nicht darauf verlassen. Ältere Browser wie Opera 6, Internet Explorer 6 oder Netscape Navigator 4 können mit diesen Zeichen in keiner Schreibweise und in keiner Kodierung etwas anfangen, weder als Dezimal- oder Hexadezimalwert noch als benannte Zeichenreferenz. Wenn im Folgenden also von Festabständen die Rede ist, sollten Sie – zumindest auf Webseiten – entweder überhaupt keinen Abstand setzen oder das geschützte Leerzeichen verwenden. Darauf werde ich später im Einzelfall aber noch genauer eingehen.

2.11.2 Waagerechte Striche

Zeichenname	Zeichen	Zeichenreferenz		
		dezimal	hexadezimal	benannt
Bindestrich	-	-	-	
geschützter Bindestrich	-	‑	‑	
Trennungsstrich	-			
kurzer Gedankenstrich	–	–	–	–
langer Gedankenstrich	—	—	—	—
Minus	–	−	−	−
Unterstrich	_	_	_	

Tabelle 2.12: Waagerechte Striche

Bindestrich

DEFINITION

*Der **Bindestrich** (auch: Bindestrich-Minus, hyphen-minus) verbindet Wörter und andere Zeichen zu Komposita.*

Der Duden benennt eine ganze Reihe von Anwendungsfällen, unter anderem

- Zusammensetzungen mit Einzelbuchstaben (E-Mail, x-Achse), Abkürzungen (Kfz-Mechaniker, Fußball-WM) oder Ziffern (100-prozentig, 17-jährig),
- substantivisch gebrauchte Aneinanderreihungen (das Entweder-Oder, das An-den-Haaren-Herbeiziehen),
- Hervorhebung einzelner Wortbestandteile (Ich-Erzählung),
- Gliederung unübersichtlicher Zusammensetzungen (Arbeiter-Unfallversicherungsgesetz) sowie
- Vermeidung von Missverständnissen (Drucker-Zeugnis, Druck-Erzeugnis).

Den Bindestrich finden Sie zwei Mal auf der Tastatur, nämlich zwischen rechter ⇧-Taste und dem Punkt sowie oben rechts im Num-Block. Verwechseln Sie den Bindestrich nicht mit dem deutlich längeren Minuszeichen!

Durchwahlnummer

Bei Telefonnummern schließen Sie Durchwahlnummern mit einem Bindestrich an die Amtsnummer an (06131 345765-224).

ISBN

Eine ISBN besteht aus Landes-, Verlags- und Artikelnummer sowie Reihenschlüssel. Diese vier Angaben trennen Sie durch Zwischenräume oder Bindestriche voneinander ab.

Auslassungsstrich

DEFINITION

*Mit dem **Auslassungsstrich** (auch: Ergänzungsstrich) zeigen Sie an, dass in Zusammensetzungen oder Ableitungen einer Aufzählung ein gleicher Bestandteil ausgelassen wurde, der sinngemäß zu ergänzen ist (Groß- und Einzelhandel, Schulbücher und -hefte).*

Richtig wäre es, bei Auslassungen den geschützten Bindestrich zu verwenden, der verhindert, dass der führende Bindestrich abgetrennt wird und einsam am Ende der Zeile steht; allerdings ist er nicht in jeder Schriftart vorhanden. Daher ist es üblich, den Bindestrich als Auslassungsstrich zu verwenden.

Trennungsstrich

DEFINITION

Wörter mit mehr als einer Silbe können Sie am Ende einer Zeile trennen. Steht am Zeilenende ein Bindestrich, so gilt er zugleich als ***Trennungsstrich*** *(auch: weicher Bindestrich, soft hyphen). Aus diesem Grund sind Trennungsstrich und Bindestrich optisch identisch.*

Der Trennungsstrich wird in den meisten Fällen, vor allem in Textverarbeitungsprogrammen wie Microsoft Word, nicht per Hand, sondern automatisch eingefügt, wenn eine Worttrennung am Zeilenende erfolgt. Browser beherrschen die automatische Wortrennung bis heute leider noch nicht. Sie können Sollbruchstellen zwar durch ein explizites Einfügen des Trennungsstrichs selbst erzeugen, üblicherweise durch die benannte Zeichenreferenz ``, allerdings handeln Sie sich damit eine ganze Reihe von Problemen ein. Ältere Browser zeigen den Strich immer an, andere interessieren sich überhaupt nicht dafür. Darüber hinaus ist die Verwendung auch in Bezug auf Suchmaschinen problematisch. So behandelt Google Trennstriche wie Leerzeichen; `Typografie` (in Googles Augen: Ty po gra fie) würde somit nicht gefunden werden, wenn nach »Typografie« gesucht würde.

Gedankenstrich

DEFINITION

Den Gedankenstrich (dash) verwenden Sie

- *zur Ankündigung, dass etwas Weiterführendes folgt oder dass Sie das Folgende als etwas Unerwartetes verstanden wissen wollen (»Sie sah kurz auf – und da stand er vor ihr.«),*
- *um einen Wechsel zwischen zwei Sätzen deutlich zu machen (»Damit haben wir diesen Punkt der Tagesordnung abgeschlossen. – Besprechen wir nun den nächsten.«)*
- *oder um Zusätze oder Nachträge abzugrenzen (»Johannes Gutenberg – der Erfinder der Buchdruckerkunst – wurde in Mainz geboren.«).*

Im Englischen gibt es zwei Arten von Gedankenstrichen: den kurzen Gedankenstrich (auch: Halbgeviertstrich, *en dash*), dessen Länge der Breite der Minuskel »n« entspricht, und den langen Gedankenstrich (auch: Geviertstrich, *em dash*), dessen Länge der Breite der Minuskel »m« entspricht. Im Deutschen wird ausschließlich der kurze Gedankenstrich verwendet, der mit vorausgehendem und folgendem Leerzeichen gesetzt wird. Der lange Gedankenstrich findet höchstens als Aufzählungszeichen Verwendung, kommt als solcher in (X)HTML allerdings praktisch gar nicht vor, da es entsprechende Elemente zur Auszeichnung von Aufzählungen (mit den Worten eines Webautors: von ungeordneten Listen) gibt.

»gegen« und »bis«

Als Zeichen für »gegen« und »bis« findet ebenfalls der kurze Gedankenstrich Verwendung. Für »gegen« ist er besonders im Sportbereich üblich (Borussia Dortmund – FC Bayern München 3 : 1) und wird mit umgebenden Leerzeichen gesetzt, für »bis« (90–100 Minuten) ohne.

Streckenangaben

Setzen Sie bei Streckenangaben den kurzen Gedankenstrich als Streckenstrich (Dortmund–München). Zwischen dem Strich und den Ortsbezeichnungen stehen keine Leerzeichen.

»00«-Strich

Den Gedankenstrich verwenden Sie auch bei glatten Währungsbeträgen als »00«-Strich hinter dem Komma (100,– EUR). In Tabellen und (Preis-)Listen ist der besseren Übersicht halber der lange Gedankenstrich gebräuchlich.

Minus

DEFINITION

Das Minus wird als Operator (Rechenzeichen) bei einer Subtraktion und als Vorzeichen negativer Zahlen eingesetzt.

Wie bereits erwähnt, ist das Zeichen oben rechts im Num-Block kein richtiges Minuszeichen, sondern ein Bindestrich. Das Minuszeichen steht nicht direkt auf der Tastatur zur Verfügung. In (X)HTML schreiben Sie es am einfachsten über die benannte Zeichenreferenz `−`. Setzen Sie um den Operator einen kleinen Zwischenraum (schmales geschütztes Leerzeichen), während Sie das Vorzeichen kompress vor die Zahl schreiben.

Unterstrich

DEFINITION

*Den Unterstrich (*low line*) verwendet man oft anstelle eines Leerzeichens in den Fällen, in denen Leerzeichen nicht erlaubt oder nicht erwünscht sind, zum Beispiel bei Dateinamen oder in URIs.*

Sie finden das Zeichen direkt auf der Tastatur, in den meisten Fällen über dem Bindestrich zwischen der rechten ⇧-Taste und dem Punkt.

2.11.3 Anführungszeichen, Apostroph und falsche Freunde

Zeichenname	Zeichen	Zeichenreferenz		
		dezimal	hexadezimal	benannt
Anführungszeichen	"	"	"	"
einfaches linkes Anführungszeichen	‘	‘	‘	‘
einfaches rechtes Anführungszeichen	’	’	’	’
einfaches unteres Anführungszeichen	‚	‚	‚	‚
doppeltes linkes Anführungszeichen	“	“	“	“
doppeltes rechtes Anführungszeichen	”	”	”	”
doppeltes unteres Anführungszeichen	„	„	„	„
einfaches gerades Anführungszeichen	'	′	′	′
doppeltes gerades Anführungszeichen	"	″	″	″
einfaches spitzes Anführungszeichen nach links	‹	‹	‹	‹
einfaches spitzes Anführungszeichen nach rechts	›	›	›	›
doppeltes spitzes Anführungszeichen nach links	«	«	«	«
doppeltes spitzes Anführungszeichen nach rechts	»	»	»	»
Apostroph	'	'	'	'
Gravis	`	`	`	
Akut	´	´	´	´
Komma	,	,	,	
Kleiner als	<	<	<	<
Größer als	>	>	>	>

Tabelle 2.13: Anführungszeichen, Apostroph und falsche Freunde

DEFINITION

Anführungszeichen *stammen aus der Antiqua-Tradition der Spätrenaissance. Sie dienten ursprünglich dazu, Textpassagen aus anderen europäischen Sprachen zu kennzeichnen. Heute verwenden wir Anführungszeichen zur visuellen Auszeichnung von direkter Rede und Zitaten sowie zur Hervorhebung von Wortteilen, Wörtern oder Satzteilen. Laut Duden werden im deutschen Schriftsatz im Allgemeinen zwei Varianten von Anführungszeichen verwendet:*

- Doppelte normale Anführungszeichen *(auch: Gänsefüßchen, Hasenöhrchen,* double quotation marks*), bestehend aus doppeltem unterem Anführungszeichen und doppeltem linkem Anführungszeichen („foo“)*
- Doppelte spitze Anführungszeichen *(auch: Guillemets, französische Anführungszeichen,* double angle quotation marks*),bestehend aus doppeltem spitzem Anführungszeichen nach rechts und doppeltem spitzem Anführungszeichen nach links (»foo«)*

Dazu kommen jeweils die sogenannten halben Anführungszeichen (‚foo‘ sowie ›foo‹), die bei Eigennamen und Begriffsdefinitionen sowie in verschachtelten Zitaten eingesetzt werden.

Anführungszeichen stehen ohne Zwischenraum vor und nach dem eingeschlossenen Textabschnitt. Bei den normalen Anführungszeichen steht das einleitende Anführungszeichen stets unten und das abschließende oben. Die spitzen Anführungszeichen zeigen nach innen zum Wort.

> > > HINWEIS

Welche der beiden Varianten Sie verwenden, ist zunächst reine Geschmackssache. Im deutschen Sprachraum sind beide Varianten gleichermaßen verbreitet. In der Handschrift werden fast ausschließlich die einfachen Anführungszeichen verwendet, da sich diese besser schreiben lassen. Für die Textverarbeitung auf dem Computer empfehle ich jedoch spitze Anführungszeichen. Die normalen Anführungszeichen befinden sich in einem Bereich der Zeichentabelle, der nicht von jeder Schriftfamilie aus ISO-8859-x unterstützt wird, während spitze Anführungszeichen in den meisten 8-Bit-Zeichensätzen wie ISO-8859-x, Windows 125x und Mac-Latin vorkommen und allgemein richtig angezeigt werden. Die spitzen Anführungszeichen fügen sich besser in den Text ein, was zu flüssigerem Lesen führt. Darüber hinaus ist die Zeichennummer um eine Ziffer kürzer. Ich finde zudem, dass spitze Anführungszeichen besser aussehen, aber nur mit diesem Argument hätte ich Sie sicherlich nicht überzeugt.

Einfache Anführungszeichen

Von den einfachen Anführungszeichen gibt es vier Varianten:

- das einfache linke Anführungszeichen (‘),
- das einfache rechte Anführungszeichen (’),
- das einfache untere Anführungszeichen (‚) und
- das einfache obere umgekehrte Anführungszeichen (‘).

Doppelte Anführungszeichen

Die doppelten Anführungszeichen kommen in drei Ausprägungen vor:

- das doppelte linke Anführungszeichen (“),
- das doppelte rechte Anführungszeichen (”) und
- das doppelte untere Anführungszeichen („).

99-66-Regel

Da das einleitende Anführungszeichen im Deutschen stets unten steht, ist jeweils offensichtlich, welches Zeichen Sie verwenden müssen. Verwechslungsgefahr besteht aber beim schließenden Anführungszeichen! Als Eselsbrücke prägen Sie sich am besten die sogenannte 99-66-Regel ein: Das einleitende Anführungszeichen (das doppelte untere Anführungszeichen) sieht aus wie die Zahl 99, das schließende Anführungszeichen (das doppelte linke Anführungszeichen) wie die Zahl 66. Bei einfachen Anführungszeichen gibt es entsprechend die »9-6-Regel«.

In anderen Ländern

Im angelsächsischen Raum gelten andere Regeln als im Deutschen. Das einleitende Anführungszeichen steht nicht unten, sondern oben, und ist, wie das schließende Anführungszeichen, zum Wort hin gewandt (“Foo”). Man kann von einer »66-99-Regel« sprechen. Die richtigen Zeichen sind das doppelte linke Anführungszeichen (“) und das doppelte rechte Anführungszeichen (”). Entsprechend gilt die »6-9-Regel« bei einfachen Anführungszeichen (‘Foo’): Verwenden Sie das einfache linke Anführungszeichen (‘) und das einfache rechte Anführungszeichen (’). Hier stimmen die Namen und das »l« sowie das »r« in den benannten Zeichenreferenzen auch mit der tatsächlichen Position des Zeichens überein; im Deutschen ist das nicht immer der Fall. Unter anderem Spanier, Italiener, Franzosen und Schweizer verwenden üblicherweise die spitzen Anführungszeichen, allerdings genau anders herum als wir Deutschen; die Anführungszeichen zeigen nach außen und »umklammern« die eingeschlossenen Textabschnitte («Foo»). Im Deutschen ist diese Ausrichtung *nicht* erlaubt.

Zitate

Wenn Sie nur einzelne Wörter und kurze Wendungen aus anderen Sprachen anführen, umschließen Sie diese mit deutschen Anführungszeichen. Zitieren Sie ganze Absätze aus einer anderen Sprache, verwenden Sie die in dieser Sprache üblichen Anführungszeichen[41].

! ! ! ACHTUNG

Das gemeine Anführungszeichen (”), der Apostroph (’), der Gravis (`), der Akut (´), das Komma (,) sowie das Kleiner-als- (<) und Größer-als-Zeichen (>) sind keine *Anführungszeichen! Diese »falschen Freunde« werden aus Bequemlichkeit gern für solche Zwecke verwendet, weil sie im Gegensatz zu den korrekten Zeichen direkt auf der Tastatur verfügbar sind und nicht über eine Tastenkombination oder ein Menü eingegeben werden müssen. Trotzdem sind Konstrukte wie `Foo´, >>Foo<<, ”Foo” oder ``Foo’’ eine typografische Katastrophe und schlichtweg falsch.*

41 *http://de.wikipedia.org/wiki/Anf%C3%BChrungszeichen#Andere_Sprachen*

Apostroph

DEFINITION

Den Apostroph verwendet man als Auslassungszeichen für einen oder mehrere Buchstaben (»Wie geht's?«) und zur Markierung des Genitivs von Namen, die auf s, ss, ß, tz, z oder x enden und kein Artikelwort bei sich haben (»Das ist Lars' Aufgabe.«).

Das Zeichen U+0027 ('), das auf den meisten Tastaturen rechts neben dem Zeichen »ä« zu finden ist, heißt zwar »Apostroph«, typografisch gesehen ist es jedoch keines, zumindest im Deutschen. Der Apostroph sollte kommaförmig gebogen sein. Am besten verwenden Sie das einfache rechte Anführungszeichen (').

2.11.4 Punkte

Zeichenname	Zeichen	Zeichenreferenz		
		dezimal	hexadezimal	benannt
Punkt	.	.	.	
Doppelpunkt	:	:	:	
Auslassungszeichen	…	…	…	…

Tabelle 2.14: Punkte

DEFINITION

*Der **Punkt** dient der Kennzeichnung des Schlusses von Ganzsätzen sowie von Abkürzungen (i. d. R.). Man verwendet ihn des Weiteren zum Gruppieren von Zahlen (10.000), in Datumsangaben (24.12.2000) und in Uhrzeitangaben (11.55 Uhr).*

Dabei nutzt man immer dasselbe Zeichen, allerdings unterscheidet sich die Setzung von Leerzeichen. Kennzeichnet der Punkt ein Satzende, steht dahinter immer ein Leerzeichen, es sei denn, er steht am Ende eines Absatzes. Setzen Sie bei der Gruppierung von Zahlen, in Datums- und Uhrzeitangaben kein Leerzeichen.

DEFINITION

*Mit dem **Doppelpunkt** kündigt man an, dass etwas Weiterführendes folgt (Familienstand: verheiratet). Es existiert ebenfalls eine Uhrzeitnotation mit Doppelpunkt (11:55 Uhr). Darüber hinaus kennzeichnet er Verhältnisangaben (Maßstab: 1 : 100, Spielstand: 1 : 1).*

Verwenden Sie auch hier stets dasselbe Zeichen, aber variieren Sie die Setzung von Leerzeichen. Platzieren Sie im normalen Textfluss kein Leerzeichen davor, aber eines danach. Notieren Sie Uhrzeiten ohne Leerzeichen; bei Verhältnisangaben setzen Sie eines davor und eines dahinter. Dabei empfehle ich das geschützte Leerzeichen, um einen Zeilenumbruch an ungünstigen Stellen zu vermeiden.

DEFINITION

*Das **Auslassungszeichen** (*horizontal ellipsis*) ist ein eigenes typografisches Zeichen, das im korrekten typografischen Satz nicht einfach durch die Aneinanderreihung von Punkten entsteht, wenngleich es in den meisten Schriftfamilien so aussieht. Setzen Sie vor und nach den Auslassungszeichen jeweils einen Wortzwischenraum, wenn sie für ein oder mehrere Wörter stehen (»Sie glaubten, auf dem richtigen Weg zu sein, doch dann …«). Schließen Sie sie bei Auslassung eines Wortteils (»Mit ›…keit‹ endet das gesuchte Wort.«) unmittelbar an den Rest des Worts an.*

Treten Auslassungszeichen am Satzende auf, setzen Sie keinen zusätzlichen Punkt zur Kennzeichnung des Schlusses. Fügen Sie Frage- und Ausrufezeichen direkt nach den Auslassungszeichen an.

Bei einigen Textverarbeitungsprogrammen ist die Verwendung der Auslassungszeichen ein wenig »fummelig«. Sicherlich wird es Ihnen niemand vorwerfen, wenn Sie einfach drei Punkte hintereinandersetzen – falls der Unterschied überhaupt jemandem auffällt.

2.11.5 Zahlen, Einheiten und Währungszeichen

Zeichenname	Zeichen	Zeichenreferenz		
		dezimal	hexadezimal	benannt
Punkt	.	.	.	
Komma	,	,	,	
Prozent	%	%	%	
Promille	‰	‰	‰	‰
Grad	°	°	°	°
Bogenminuten, Fuß	'	′	′	′
Bogensekunden, Zoll	"	″	″	″
Euro	€	€	€	€
Cent	¢	¢	¢	¢
Dollar	$	$	$	

Tabelle 2.15: Zahlen, Einheiten und Währungszeichen

Zeichenname	Zeichen	Zeichenreferenz		
		dezimal	hexadezimal	benannt
Pfund	£	£	£	£
Yen	¥	¥	¤	¥
Franc	₣	₣	₣	
Gulden	ƒ	ƒ	ƒ	
generisches Währungssymbol	¤	¤	¤	¤

Tabelle 2.15: Zahlen, Einheiten und Währungszeichen (Fortsetzung)

Bei Zahlen gibt es Dezimaltrennzeichen sowie Gruppierungszeichen. In Deutschland verwenden wir das Komma als Dezimal- und den Punkt oder das schmale geschützte Leerzeichen als Gruppierungszeichen. Dabei gliedern wir links und rechts des Kommas vom diesem ausgehend in 3-stellige Gruppen (3,141.592). Bei 4-stelligen Zahlen hat sich neben der Schreibung mit Gruppierungszeichen auch die ohne eingebürgert; 1.500, 1 500 sowie 1500 sind allesamt erlaubt. Gliedern Sie *nicht* Jahreszahlen, Seiten- und Paragrafenangaben.

Setzen Sie bei Zahlen mit dazugehöriger Einheit zwischen Wert und Einheit ein geschütztes Leerzeichen (1.273,15 K). Das gilt auch für das Währungszeichen (1,22 $), das entweder vor oder nach dem Betrag steht. Bei Nichtbuchstabeneinheiten wie Zoll, Prozent oder Winkel entfällt das Leerzeichen (7", 100 %).

Datum und Uhrzeit

Setzen Sie bei traditionellen Datumsangaben mit der Reihenfolge Tag, Monat, Jahr einen Punkt nach den Zahlen für Tag und Monat. Der Duden legt fest, dass zwischen Tag und Monat ein kleiner Zwischenraum gesetzt wird. Am besten eignet sich das schmale geschützte Leerzeichen. Setzen Sie zwischen Monat und Jahr ein Leerzeichen, es sei denn, das Jahr wird zweistellig angeben; in diesem Fall sollten Sie ebenfalls das schmale geschützte Leerzeichen verwenden. Da dieses, wie bereits erwähnt, jedoch nicht übergreifend unterstützt wird, empfehle ich, keinen Zwischenraum zu setzen, sowohl bei zwei- als auch bei vierstelliger Jahresangabe (24.12.2002). Diese Schreibweise ist auch allgemein üblich. Wenn Sie den Monat ausschreiben, sollten Sie allerdings davor und danach ein einfaches geschütztes Leerzeichen anfügen (24. Dezember 2004).

DIN 5008

Gliedern Sie eine Datumsangabe nach DIN 5008 durch Trennstriche; die Reihenfolge ist Jahr, Monat, Tag, wobei die Jahresangabe auch hier zwei- oder vierstellig erfolgen kann (1970-01-01 bzw. 70-01-01). Diese Schreibweise ist die internationale Standardschreibweise, im Deutschen aber weniger geläufig. Bei aufeinanderfolgenden Jahreszahlen verwenden Sie den Schrägstrich (*slash*) ohne Trennzeichen (2000/2001).

Für Uhrzeitangaben gibt es im Deutschen keine einheitliche Schreibweise. Folgende Notationen und Kombinationen daraus sind möglich:

- Es ist 8 Uhr.
- Es ist 8:30 Uhr.
- Es ist 8.30 Uhr.
- Es ist 08:30 Uhr.
- Es ist 08.30 Uhr.
- Es ist 830 Uhr.
- Es ist 08:30:25 Uhr.
- Es ist 08.30.25 Uhr.

Wichtig ist, dass Sie innerhalb eines Dokuments und am besten innerhalb der gesamten Website konsequent die gleiche Schreibweise verwenden und zwischen Zahl und Trennzeichen *keine* Leerzeichen setzen.

! ! ! ACHTUNG

Sie haben bei der Wahl der Schreibweise von Datum und Uhrzeit in den meisten Fällen freie Hand. In bestimmten Anwendungsbereichen jedoch wird eine festgelegte Syntax verlangt. Der klassische Fall sind vorgegebene Felder in Formularen oder bestimmte Anwendungen in der elektronischen Datenverarbeitung beziehungsweise Informatik wie zum Beispiel Datenbanken. Die Feldtypen `date` *(Format:* `jjjj-mm-tt`*),* `time` *(Format:* `hh:mm:ss`*) und* `datetime` *(Format:* `jjjj-mm-tt hh:mm:ss`*) der Abfragesprache für relationale Datenbanken SQL verlangen beispielsweise jeweils ein bestimmtes Format; andernfalls kommt es zu Fehlern.*

2.11.6 Mathematische Symbole und Formeln

Zeichenname	Zeichen	Zeichenreferenz		
		dezimal	hexadezimal	benannt
Plus	+	+	+	
Minus	-	−	−	−
Plusminus	±	±	±	±

Tabelle 2.16: Mathematische Symbole und Formeln[41]

42 Falls Sie ein bestimmtes Zeichen suchen, das ich nicht aufgeführt habe, schauen Sie am besten in der Unicode-Tabelle *http://www.unicode.org/charts/PDF/U2200.pdf2* nach, die alle Zeichen von U+2200 bis U+22FF übersichtlich auflistet.

Zeichenname	Zeichen	Zeichenreferenz		
		dezimal	hexadezimal	benannt
Multiplikation	×	×	×	×
Punkt-Operator	⋅	⋅	⋅	⋅
Division	÷	÷	÷	÷
Bruchstrich	/	⁄	⁄	⁄
Gleich	=	=	=	
Ungleich	≠	≠	≠	≠
Kleiner als	<	<	<	<
Größer als	>	>	>	>
Kleiner-gleich	≤	≤	≤	≤
Größer-gleich	≥	≥	≥	≥
Für alle	∀	∀	∀	∀
Es existiert	∃	∃	∃	∃
Leere Menge	∅	∅	∅	∅
Element von	∈	∈	∈	∈
Nicht Element von	∉	∉	∉	∉
Produkt	∏	∏	∏	∏
Summe	∑	∑	∑	∑
Unendlich	∞	∞	∞	∞
Logisches Und	∧	∧	∧	∧
Logisches Oder	∨	∨	∨	∨

Tabelle 2.16: Mathematische Symbole und Formeln[41] (Fortsetzung)

Rechen- und Formelzeichen sollten Sie zwischen Zahlen und Mengen mit kleinem Zwischenraum (schmales geschütztes Leerzeichen) setzen. Da dieses, wie bereits erläutert, nicht überall vorhanden ist, empfehle ich, entweder kein oder das einfache geschützte Leerzeichen zu setzen (5–3=2 oder 5–3 = 2). Setzen Sie Vorzeichen kompress vor die Zahl (–5).

Mathematische, physikalische und chemische Formeln sollten Sie einrücken und freistellen sowie nach Möglichkeit *nicht* trennen.

MathML

Umfangreiche Formeln syntaktisch korrekt ins Web zu portieren, ist in reinem (X)HTML nicht möglich, daher behelfen Webautoren sich bis heute mit Grafiken; eine sehr umständliche und wenig effiziente Notlösung. 1997 wurde die *W3C Math Working Group* gegründet, die sich mit dieser Problematik auseinandersetzt. Im April 1998 veröffentlichte diese MathML, seit Oktober 2003 in der aktuellen Version 2.0 verfügbar. MathML ist ein Dokumentformat zur Darstel-

lung von Formeln und komplexen Ausdrücken. Wie in allen XML-Dokumenttypen werden die Inhalte eines Dokuments in MathML in einer logischen Struktur abgelegt. Mehr Informationen erhalten Sie auf der offiziellen MathML-Seite des W3C[43]. Dort finden Sie auch eine Übersicht über alle MathML-Implementierungen[44].

2.11.7 Pfeile

Zeichenname	Zeichen	Zeichenreferenz		
		dezimal	hexadezimal	benannt
Pfeil nach links	←	←	←	←
Pfeil nach oben	↑	↑	↑	↑
Pfeil nach rechts	→	→	→	→
Pfeil nach unten	↓	↓	↓	↓
Nordwestpfeil	↖	↖	↖	
Nordostpfeil	↗	↗	↗	
Südostpfeil	↙	↘	↘	
Südwestpfeil	↘	↙	↙	

Tabelle 2.17: Pfeile

Pfeile haben typografisch gesehen keine Bedeutung. Allerdings können Sie sie gut für dekorative Zwecke einsetzen. Der Nordostpfeil beispielsweise ist ein gern verwendetes Symbol zur Kennzeichnung von externen Links (siehe das Beispiel in Kapitel 6.5.6).

43 *http://www.w3.org/Math/*
44 *http://w3.org/Math/implementations.html*

3 Hilfsmittel

3.1 Editoren

XHTML- und CSS-Dokumente sind keine ausführbaren oder Binärdateien, sondern einfache Textdateien, das heißt, sie bestehen aus für den Menschen lesbarem Text und lassen sich mit jedem einfachen textorientierten Editor (auch: ASCII-Editor) öffnen und editieren. Jedoch greifen viele Webentwickler auf umfangreiche und zum Teil verhältnismäßig teure WYSIWYG-Programmpakete namhafter Hersteller zurück, die die Erstellung und Bearbeitung von Dokumenten unabhängig von deren Quelltext erlauben. Dieser Abschnitt stellt Vor- und Nachteile beider Editorarten gegenüber.

3.1.1 WYSIWYG-Programme

DEFINITION

WYSIWYG *ist das Akronym für das Prinzip* What You See Is What You Get *(Was du siehst, ist das, was du erhältst). Bei echtem WYSIWYG wird ein Dokument während der Bearbeitung am Bildschirm exakt so angezeigt, wie es bei der Ausgabe über ein anderes Gerät aussieht, beispielsweise einem Drucker. Übertragen auf die Webentwicklung ist ein WYSIWYG-Programm ein Werkzeug, dessen Darstellung während der Bearbeitung der XHTML-Datei deren späterer Anzeige im Browser exakt entspricht.*

Microsofts Word gehört zu den bekanntesten WYSIWYG-Programmen. Der Benutzer sieht anfangs eine leere weiße Seite vor sich und beginnt, sie zu editieren. Er kann Texte formatieren, Bilder einfügen und Tabellen anlegen. Die gewünschten Inhalte lassen sich über Menübefehle und Dialoge einfügen. Die Ergebnisse seiner Aktionen kann er direkt auf dem Bildschirm überprüfen.

Nach demselben Prinzip soll ein WYSIWYG-Programmpaket zur Erstellung von Webseiten funktionieren. Das Programm setzt hierbei die vom Benutzer ausgeführten Aktionen im Hintergrund in Quelltext um. Wenn er es nicht möchte, braucht der Benutzer den Raum der grafischen Darstellung also gar nicht zu verlassen. Viele WYSIWYG-Editoren bieten auch die Möglichkeit, zwischen Layout- und Quelltextansicht umzuschalten, sodass Sie die Entwicklung der Seite auch direkt im Quelltext verfolgen und dort Korrekturen und Änderungen vornehmen können.

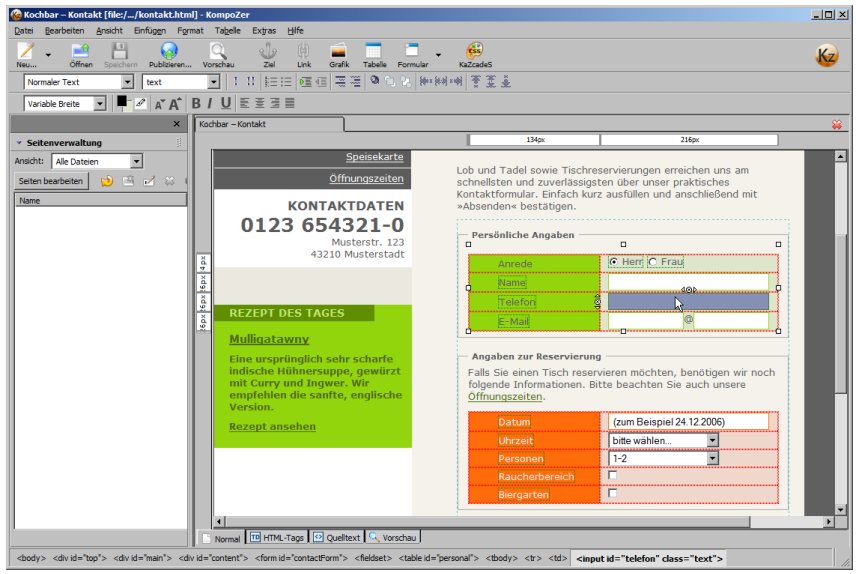


Abbildung 3.1: Der WYSIWYG-Editor KompoZer

Vor- und Nachteile

Die Vorteile eines WYSIWYG-Programms liegen in der Natur der Sache:

Vorteile

- Sie sind sehr schnell in der Lage, eine Seite fertigzustellen oder ein bestehendes Grundgerüst mit Inhalten zu füllen.
- Das Transkribieren von der abstrakten Quelltextebene hin zu der Seitendarstellung im Browser fällt weg.

- WYSIWYG-Programme verlangen keine oder kaum XHTML-Kenntnisse zur Fertigstellung einer Seite und verfügen zusätzlich oft über Layoutvorlagen, die Ihnen die Konzeption eines Screendesigns weitgehend abnehmen.
- Viele WYSIWYG-Programme verfügen über ein ausgereiftes Projektmanagement, das Ihnen die konzeptionelle Arbeit zum Teil abnimmt.

WYSIWYG-Programmpakete scheinen ein guter Einstieg in die Webentwicklung zu sein, da Sie mit minimalen XHTML- und CSS-Kenntnissen rasche Ergebnisse erhalten und schnell ein Gefühl dafür entwickeln, welche Möglichkeiten klassisches Webauthoring bietet und welche nicht. Leider führt diese Arbeitserleichterung insgesamt zu einer Simplifizierung des Erstellens von Webseiten. Dadurch dass das Programm zum einen die Gedankenarbeit sowohl beim Projektmanagement als auch beim Erstellen des Seitenkonzepts und zum anderen auch die Erstellung des Quelltexts abnimmt, bekommt der Anwender schnell das Gefühl, die Erstellung von Webseiten sei eine sehr einfache Sache. Er hat damit nur dann recht, wenn er sich mit den Ergebnissen des WYSIWYG-Programms zufriedengibt. Dies sollten Sie jedoch nicht, denn leider haben derartige Programme auch sehr viele Nachteile:

Nachteile

- WYSIWYG-Programme erzeugen in einigen Fällen unerklärlicherweise viel überflüssigen Code, was längere Ladezeiten im Browser und eine große Unübersichtlichkeit des Quelltexts bedeutet.
- Oftmals entspricht der erzeugte Quelltext nicht den W3C-Empfehlungen; Sie erzeugen folglich ungültige Dokumente. Dies kann dazu führen, dass kein Browser die Webseite tatsächlich so darstellt, wie es die Vorschau Ihres WYSIWYG-Programmpakets, auf dem Sie sie entworfen haben, zeigt. Dies bedeutet unter Umständen kleinen Abweichungen, aber auch das völlige Fehlen wichtiger Seiteninhalte (siehe auch Kapitel 3.2 »Validatoren«).
- Ein nicht unwesentlicher Nachteil ist die häufig mangelhafte Trennung von Markup und Layout und das inkonsequente Einsetzen von CSS, was Sie als Webautor per Hand wesentlich effizienter gestalten können.

Das Web kennt kein WYSIWYG

WYSIWYG-Programmpakete werden zu Unrecht so genannt. Das WYSIWYG-Prinzip stimmt immer nur für den Autor einer Webseite selbst und für einen gewissen Nutzerkreis, der über die gleichen Voraussetzungen und die gleiche Umgebung wie der Autor verfügt. Im World Wide Web müssen Sie davon ausgehen, dass die Besucher die erstellten Seiten mit anderen Browsern auf anderen Betriebssystemen über andere Auflösungen betrachten. Dadurch können die Seiten bei den Besuchern anders aussehen als beim Autor. WYSIWYG-Programme sind somit vielmehr

- WYSIWYMG- (*What You See Is What You Might Get*, Was du siehst, ist das, was du vielleicht erhältst),
- WYSIMOLWYG- (*What You See Is More Or Less What You Get*, Was du siehst, ist mehr oder weniger das, was du erhältst),
- WYTYSIWYTYG- (*What You Think You See Is What You Think You Get*, Was du zu sehen glaubst, ist das, was du zu erhalten glaubst) oder
- WYSIWYM-Programme (*What You See Is What You Mean*, Was du siehst, ist, was du meinst) – vielleicht die treffendste Bezeichnung.

Aus diesem Grund verwendet kaum ein Hersteller eines Programms, das man in die Reihe der WYSIWYG-Programme einordnet, diesen Begriff selbst; da scheinen sich eher Bezeichnungen durchzusetzen wie »Werkzeug für Web-Authoring« (*Macromedia Dreamweaver*) oder »Tool für die Anwendungsentwicklung und die Webseitenerstellung« (*Microsoft FrontPage*).

3.1.2 Texteditoren

DEFINITION

Ein textorientierter Editor ist ein Programm zum Bearbeiten von Texten, beispielsweise von XHTML-Dokumenten auf Quelltextebene. Der Editor lädt das zu bearbeitende Textdokument und zeigt seinen Inhalt auf dem Bildschirm an. Durch diverse Aktionen können die Daten dann bearbeitet werden, beispielsweise durch Einfügen, Löschen oder Kopieren.

Vor- und Nachteile

Die Nachteile von Texteditoren sind – gerade für Anfänger – offensichtlich:

Nachteile

- Sie benötigen XHTML- und CSS-Kenntnisse, um eine Seite zu erstellen.
- Das Tippen des Quelltexts dauert länger als das Zusammenklicken eines Dokuments in einem WYSIWYG-Programmpaket, selbst wenn Sie ausgiebig mit Includes (Mehrfachverwendung von in der Regel zentral abgelegtem Quelltext) oder Kopieren und Einfügen arbeiten.
- Texteditoren verfügen in den meisten Fällen über kein umfangreiches Projekt- und Dokumentenmanagement.
- Sie sind auf sich allein gestellt, wenn es um die Erarbeitung einer Informationsarchitektur geht.
- Das Umsetzen eines Entwurfskonzepts in einen gut strukturierten XHTML-Quelltext erfordert große Abstraktionsfähigkeiten.

Die meisten Vorteile von Texteditoren offenbaren sich erst nach kurzer Eingewöhnungszeit:

Vorteile

- Sie haben direkte Kontrolle über die Entstehung des Seitenquelltexts.
- Sie können die Auswirkungen jeder noch so geringen Änderung am Quelltext sofort im Browser überprüfen.
- Sie beschränken sich beim Schreiben Ihrer Seite auf das Wesentliche und schreiben dadurch keinen überflüssigen Code.
- Sie ordnen Ihren Quelltext für Sie verständlich und logisch.
- Sie haben enorme Lernfortschritte mit jeder Seite, die Sie schreiben. Sie gewinnen immer mehr Verständnis für die Zusammenhänge und sehen die Textauszeichnung im Quelltext getrennt von der Darstellung im Browser. Dadurch entwickeln Sie ein besseres Verständnis für stilistisch gutes Webauthoring.
- Ein Wechsel des Editors ist leichter als bei WYSIWYG-Programmen, da der eigene Quellcode mit einem anderen Editor nach dem gleichen Prinzip weiterverarbeitet werden kann.
- Es liegt in Ihrer Hand, nicht in der Hand eines Programms, ob Sie beim Erstellen der Seite Fehler machen oder nicht.

Erfahrene Webautoren bevorzugen in der Regel reine Texteditoren. Wie bereits erwähnt, reichen einfachste Texteditoren wie Notepad unter Windows vollkommen aus. Björn Höhrmann schrieb vor einigen Jahren überspitzt, aber nicht ganz zu Unrecht:

> *»Notepad ist kein HTML-Editor, Notepad ist das Schweizer Armeemesser mit integriertem Whirlpool für jeden, der nur ein bisschen auf sich hält und effektiv arbeiten will.«*

Dennoch sollten Sie bei der Wahl Ihres Editors auf einige nützliche Funktionen achten, die Ihnen die Arbeit an Ihrer Website ein wenig erleichtern:

Kriterien für gute Texteditoren

- **Syntaxhervorhebung** (*syntax highlighting*) von mindestens XHTML und CSS. Dadurch werden einzelne syntaktische Bestandteile eines Dokuments und eigentliche Inhalte unterschiedlich eingefärbt, wodurch die Orientierung im Quelltext erleichtert wird und Fehler schneller sichtbar werden. Syntaxhervorhebung für weitere relevante Sprachen wie JavaScript, PHP, Perl und andere ist wünschenswert und in den meisten Editoren auch vorhanden oder integrierbar.

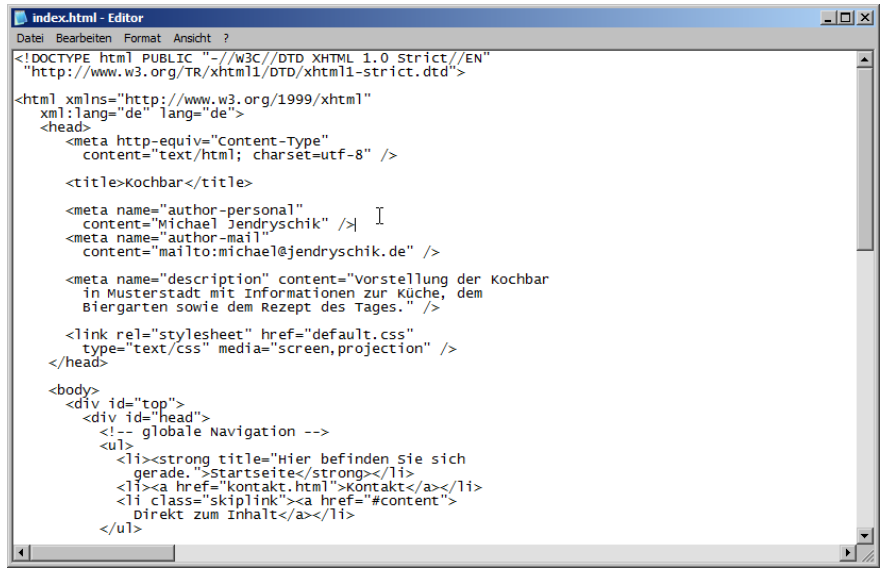


Abbildung 3.2: Minimalausstattung: Notepad

- **Dateiübergreifendes Suchen und Ersetzen**. Müssen Änderungen in einem Dokument oder in mehreren vorgenommen werden, führt eine einfache Suchen-und-Ersetzen-Funktion zu schnellen Ergebnissen. Die Verwendung von **regulären Ausdrücken** (*regular expressions*) wird der Profi ebenfalls zu schätzen wissen.
- **UTF-8-Fähigkeit** und ganz allgemein die Unterstützung und Konvertierung von möglichst vielen Kodierungen und Zeichensätzen, wie den Familien ISO-8859-n, Windows 125 oder MacRoman, um alle nötigen Zeichen eingeben, anzeigen und abspeichern zu können (siehe Kapitel 2.10 »Zeichenkodierung«).
- Zusätzlich eingebaute oder nachrüstbare **Werkzeuge** wie Validatoren, Rechtschreibprüfung, Quelltextvervollständigung, Makrofähigkeit, Verwendung von Textbausteinen, automatische Einrückung von Quelltextzeilen und Includes können die Arbeit ebenso vereinfachen wie benutzerdefinierte Werkzeugleisten mit häufig verwendeten Kommandos oder eigene Tastenkombinationen.

An dieser Stelle soll nicht verschwiegen werden, dass die Welt der Editoren nicht nur schwarz-weiß ist. Die meisten WYSIWYM-Programme ermöglichen beide Arbeitsformen, und es gibt Entwickler, die eine zweigleisige Arbeitsweise als die effektivste empfinden. Die WYSIWYM-Oberfläche dient dabei nur dem bequemen Editieren und Auszeichnen mit Standardelementen, während der Quelltext jederzeit parallel sichtbar ist und die Gestaltung vollständig

über CSS vorgenommen wird – häufig assistiert durch das jeweilige Programm. Abbildung 3.3 zeigt ein Beispiel für diese Arbeitsweise.

Abbildung 3.3: Kombinierte Quelltext- und WYSIWYG-Ansicht in Macromedia Dreamweaver 8 (Screenshot von Andreas Borutta)

3.1.3 Fazit

Je nach verwendetem Betriebssystem, Geschmack und der erforderlichen Einarbeitungszeit mag die Wahl auf den einen oder anderen Editor fallen. Manche bevorzugen universelle und extrem mächtige Editoren wie *vim* oder *UltraEdit*, deren Bedienung allerdings längere Zeit erlernt werden will, die dann aber keine Wünsche offen lassen und zum Teil auf zahlreichen Betriebssystemen verfügbar sind. Andere bevorzugen einfach zu bedienende Editoren wie *Notepad++* oder *BBEdit*. Auf Websprachen spezialisierte Editoren sind *Phase 5*, *Homesite* und *Bluefish*. Programmierer werden XHTML- und CSS-Dokumente mit dem Quelltexteditor ihrer Software-Entwicklungsumgebung bearbeiten, beispielsweise *Eclipse*. Tabelle 3.1 und Tabelle 3.2 listen einige empfehlenswerte Editoren auf. Ausführliche Auflistungen finden Sie darüber hinaus bei Dr. Web: *http://www.drweb.de/editoren/wysiwyg-editoren-markt.shtml* listet nützliche WYSIWYG-Programme auf, *http://www.drweb.de/editoren/html-editoren-markt.shtml* eine Reihe von Texteditoren.

Editor	**OS**	**Kostenlos?**	**URL**
Arachnophilia	Win	ja	http://www.arachnoid.com/arachnophilia/
BBEdit	Mac	nein	http://www.barebones.com/products/bbedit/
BlueFish	Unix	ja	http://bluefish.openoffice.nl/
Eclipse	Win/Mac/Unix	ja	http://www.eclipse.org/
HomeSite	Win	nein	http://www.adobe.com/products/homesite/
Notepad++	Win	ja	http://notepad-plus.sourceforge.net/de/site.htm
Notepad2	Win	ja	http://www.flos-freeware.ch/notepad2.html
PageSpinner	Mac	nein	http://www.optima-system.com/pagespinner/
Phase 5	Win	eingeschränkt	http://www.phase5.info/
UltraEdit	Win	nein	http://www.ultraedit.com/
vim	Win/Mac/Unix	ja	http://www.vim.org/

Tabelle 3.1: Empfehlenswerte Texteditoren

Editor	OS	Kostenlos?	URL
Dreamweaver	Win/Mac	nein	http://www.adobe.com/products/dreamweaver/
Namo WebEditor	Win	nein	http://www.namo.com/products/webeditor.php
NetObjects Fusion	Win	nein	http://www.netobjects.de/
KompoZer	Win/Mac/ Unix/OS2	ja	http://www.kompozer.net/

Tabelle 3.2: Empfehlenswerte WYSIWYM-Editoren

Es gibt Hunderte guter Editoren für die unterschiedlichsten Ansprüche. Schauen Sie sich in Ruhe einige an und probieren Sie sie längere Zeit aus. Ich selbst arbeite seit mehreren Jahren mit dem Texteditor *UltraEdit*. Auf diesen oder einen anderen Texteditor sollten Sie vor allem dann zurückgreifen, wenn Sie diese Einführung durcharbeiten.

3.2 Validatoren

! ! ! ACHTUNG

Um diesen Abschnitt umfassend verstehen zu können, benötigen Sie Kenntnisse über die Syntax und das Vokabular von XHTML und sollten wissen, wie HTML und XHTML aus SGML und XML entstanden sind. Zum Teil greife ich Inhalten weiterer Teile dieser Einführung vor, allerdings denke ich, dass ein Abschnitt über Validatoren in diesem Kapitel am besten aufgehoben ist – schließlich gehören Validatoren zu den wichtigsten Hilfsmitteln eines Webautors überhaupt. Anfänger sollten diesen Abschnitt zunächst überspringen und ihn sich für einen späteren Zeitpunkt vormerken.

3.2.1 Natürliche und formale Sprachen

Damit sich zwei Menschen miteinander verständigen können, benötigen sie eine gemeinsame (natürliche) Sprache. Zum einen müssen sich beide ein gemeinsames Vokabular teilen, also über eine ausreichend große Menge an Wörtern verfügen und sich über deren Bedeutung und Funktion einig sein. Zum anderen müssen sie zumindest in groben Zügen dieselben Regeln dazu befolgen, wie aus einzelnen Wörtern Bedeutungsgruppen und daraus wiederum Sätze geformt werden. Im Idealfall lassen sich dann Sätze formulieren, deren Bedeutung eindeutig und beiden Beteiligten klar ist.

Mehrdeutigkeiten natürlicher Sprachen

So einfach ist dies jedoch nicht. Jeder weiß aus der alltäglichen Kommunikation mit Menschen in seiner Umgebung, dass sich die Bedeutung von Ausdrücken oft nur aus dem Zusammenhang ergibt. Wesentliche Bestandteile einer

natürlichen Sprache sind nicht nur Vokabular und Grammatik, sondern auch Melodie und Rhythmus, Mimik und Gestik, Ironie und Sarkasmus, Emotion und Beziehungsdefinitionen. Hinzu kommt, dass viele Äußerungen in natürlicher Sprache nicht nur in ihrer Bedeutung (Semantik), sondern auch in ihrer grammatikalischen Struktur (Syntax) mehrdeutig sind. Der Ausdruck »die Auswahl des Mannes mit dem Zeigestock« veranschaulicht dieses Problem. Der Mann könnte ausgewählt worden sein oder selbst eine Auswahl getroffen haben; es kann sowohl »die Auswahl mit dem Zeigestock« gemeint sein als auch »der Mann mit dem Zeigestock«.

Natürliche Sprachen sind nicht vollständig durch ein festgelegtes Regelwerk, sondern durch ihren Gebrauch definiert und unterliegen ständigem Wandel. Es kann daher nicht immer zweifelsfrei bestimmt werden, ob eine Äußerung oder ein Satz einer bestimmten natürlichen Sprache zuzuordnen und in diesem Zusammenhang »richtig« oder »falsch« ist. Daher sind natürliche Sprachen für die Kommunikation zwischen Menschen und Computern oder Computerprogrammen untereinander nicht geeignet. Anders als wir Menschen kann ein Computer Mehrdeutigkeiten nicht auflösen und ist auch nicht in der Lage, zwischen den Zeilen zu lesen und die Bedeutung eines Ausdrucks aus dem Kontext heraus zu interpretieren. Besser eignen sich formale Sprachen, deren Semantik und syntaktische Regeln anders als bei natürlichen Sprachen eindeutig sind.

Syntax und Grammatik

HTML als SGML-Dokumenttyp und XHTML als XML-Dokumenttyp sind formale Sprachen. Die Grammatik wird durch die syntaktischen Regeln gebildet, die für SGML beziehungsweise XML definiert sind, sowie durch die Elementstrukturen, die die jeweilige **Dokumenttyp-Definition** (DTD) oder das Schema vorgibt – also welche Elemente auf welche Weise ineinander verschachtelt werden dürfen, welche Attribute sie haben dürfen oder müssen und welche Werte diese annehmen können. Das folgende Beispiel ist aus zwei Gründen falsch: Es verstößt gegen die Syntax von XHTML, weil ein Element `p` keine anderen Blockelemente enthalten darf, und hält sich nicht an das vorgegebene Vokabular, da es einen Elementtyp `et` verwendet, der nicht in XHTML enthalten ist:

```
<p>Das folgende Beispiel ist aus zwei Gründen falsch: Es
  <ul>
  <li>verstößt gegen die Syntax von XHTML, weil ein
    Element <et>p</et> keine anderen Blockelemente
    enthalten darf, und</li>
  <li>hält sich nicht an das vorgegebene Vokabular, da es
    einen Elementtyp <et>et</et> verwendet, der nicht in
    XHTML enthalten ist.</li>
  </ul>
</p>
```

Ein Computer, genauer gesagt ein Validator, kann HTML- und XHTML-Dokumente mithilfe eines Parsers überprüfen und Fehler so eindeutig erkennen.

3.2.2 Syntaktische Fehler und proprietäre Erweiterungen

Viele sogenannte WYSIWYG-Editoren produzieren noch heute teilweise unsauberes oder gar fehlerhaftes (X)HTML. Aber auch Webautoren, die ihre Dokumente per Hand schreiben, sind vor Fehlern nicht gefeit; wie schnell vergisst man, ein schließendes Tag zu setzen, oder fügt einen URL ein und übersieht ein nicht maskiertes Ampersand (`&`). Derartige Fehler fallen bei mit dem MIME-Typ `text/html` ausgelieferten Dokumenten nicht auf, zumindest nicht auf den ersten Blick; es sei denn, man bemerkt fehlende oder falsch angezeigte Inhalte in der Darstellung. Dennoch bringen sie Probleme mit sich.

Fehlerbehandlung

Sind Dokumente fehlerhaft geschrieben, muss der Browser die falschen Stellen interpretieren und sich für eine mögliche Darstellung entscheiden. Das bläht Webseiten auf und macht ihre Verarbeitung langsam. Darüber hinaus ist Fehlerbehandlung nicht genormt. Wer syntaktisch falsches (X)HTML schreibt, kann sich nicht darauf verlassen, dass aktuelle und zukünftige Browser die Dokumente richtig verarbeiten können und morgen noch genau so darstellen wie heute. Werden XHTML-Dokumente mit dem empfohlenen MIME-Typ `application/xhtml+xml` oder einem anderen XML-MIME-Typ ausgeliefert, zumindest an Browser wie Mozilla oder Firefox, die damit etwas anfangen können, wird die Verarbeitung des Dokuments sofort abgebrochen und eine Fehlermeldung angezeigt (siehe Abbildung 3.4) – die Informationen, die dargestellt werden sollten, sind somit nicht mehr zugänglich. Zudem erhalten potenzielle Besucher oder Kunden natürlich auch einen denkbar schlechten Eindruck von der fachlichen Qualifikation des jeweiligen Webautors.

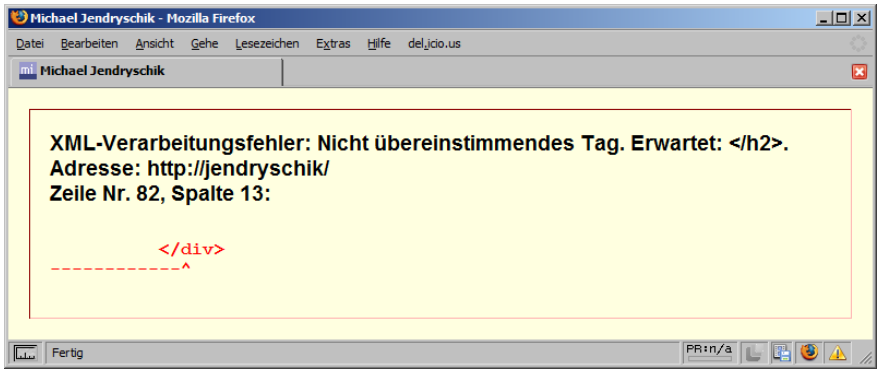


Abbildung 3.4: Firefox zeigt Verstöße gegen die Wohlgeformtheit eines XHTML-Dokuments gnadenlos an

Erweiterungen

Bei Verstößen gegen das Vokabular der verwendeten Sprachversion handelt es sich oft um Elemente oder Attribute, die nur in der Transitional-DTD enthalten

sind, aber in einem strikten Dokument verwendet werden, oder um proprietäre Erweiterungen, die in keiner (X)HTML-DTD enthalten, also nicht genormt sind. Letzteres sind in den meisten Fällen Erweiterungen einzelner Browserentwickler, die oft auch nur in Browsern dieser Hersteller interpretiert werden. Vom Gebrauch solcher Erweiterungen ist abzuraten, da es sehr wahrscheinlich ist, dass sie in späteren Generationen des Browsers, in anderen Browsern oder auf anderen Plattformen nicht dargestellt werden oder zumindest anders als gewünscht, was viel schlimmer ist. Darüber hinaus können Werkzeuge wie Tidy (siehe Kapitel 3.4.1) Dokumente, die unbekannte Erweiterungen enthalten, nicht bearbeiten, da sie nicht wissen, wie sie mit diesen Erweiterungen umzugehen haben. Beispielsweise bleibt bei unbekannten Elementen innerhalb von XHTML-Dokumenten die Frage offen, ob das **End-Tag** erforderlich (wie bei `a`), optional (`p`) oder verboten (`img`) ist.

3.2.3 Gültigkeit und Wohlgeformtheit

DEFINITION

*Ein Dokument ist **gültig** (valide), wenn es erfolgreich gegen eine zugrunde liegende Grammatik (DTD, Schema etc.) geprüft werden kann. Es dürfen also nur die Elemente, Attribute und Attributwerte auf die Art und Weise verwendet (und verschachtelt) werden, wie es für die gewählte Sprachversion definiert ist. Nur ein vollständig fehlerbereinigtes und korrekt geschriebenes Dokument verdient das Attribut gültig.*

DEFINITION

*In XML wird das Konzept der **Wohlgeformtheit** eingeführt. Demnach ist ein XML-Dokument genau dann wohlgeformt, wenn es nach den in Kapitel 2.1 der XML 1.0-Empfehlung definierten Regeln strukturiert ist. Kurz zusammengefasst besagen diese Regeln:*

- *Es existiert genau ein Wurzelelement. In XHTML-Dokumenten ist es das Element `html`.*
- *Alle Elemente werden durch ein Start- und ein End-Tag begrenzt oder bestehen aus einem Leeres-Element-Tag. Anders als bei HTML dürfen Webautoren bei XHTML-Dokumenten beispielsweise das schließende `</p>` nicht mehr weglassen.*
- *Elemente sind korrekt ineinander verschachtelt. `<strong><a href="foo">bar</strong></a>` mag gültiges HTML sein; XHTML ist dies nicht. In XHTML müssen Sie `<strong><a href="foo">bar</a></strong>` oder `<a href="foo"><strong>bar</strong></a>` schreiben: Die Tags dürfen sich nicht überkreuzen. Welche Variante Sie wählen, ist übrigens abhängig davon, ob Sie Links betonen oder Betonungen verlinken möchten. Sind Sie empfänglich für solche feinen Unterschiede?*
- *Attributwerte müssen immer in Anführungszeichen gesetzt werden.*
- *Es gibt keine Attributminimierung. `<input type="checkbox" checked />` ist somit nicht wohlgeformt, `<input type="checkbox" checked="checked" />` hingegen schon.*

Gültigkeit als Voraussetzung für Wohlgeformtheit

Die XML-Spezifikation definiert den Begriff »gültig« nur für XML-Dokumente, und genau genommen ist ein XML-Dokument nur dann ein solches, wenn es wohlgeformt ist. XHTML-Dokumente sind XML-Dokumente bestimmter Ausprägung, folglich ist Wohlgeformtheit ein notwendiges Kriterium, um XHTML-Dokumente auf Gültigkeit gegen eine bestimmte Dokumenttyp-Definition prüfen zu können. Diese Voraussetzung gibt es für HTML als SGML-Dokumenttyp nicht, denn SGML kennt das Konzept der Wohlgeformtheit nicht. Jedoch wird seit Aufkommen von X(HT)ML auch ein HTML-Dokument hin und wieder informell als »quasi-wohlgeformt« bezeichnet, wenn es den oben angegebenen, sofern in HTML möglichen, Regeln genügt.

3.2.4 W3C Markup Validation Service

Um zu überprüfen, ob ein Dokument gültig ist, verwenden Webautoren einen für diesen Zweck programmierten Validator. Davon gibt es mittlerweile eine ganze Reihe auf dem Markt und frei für jeden verfügbar im Web. Der *W3C Markup Validation Service*[1] ist der bekannteste. Er beschreibt sich selbst als freien Service, über den Webautoren Dokumente wie HTML- und XHTML-Dokumente bezüglich ihrer Konformität zu den W3C-Empfehlungen und anderen Standards überprüfen können. Aber auch der Validator der *Web Design Group*[2] ist sehr populär. Nebenbei bemerkt, bietet dieser über die Funktion *Validate entire site* die Möglichkeit, bis zu 100 Dokumente einer Website in einem Schritt zu überprüfen.

Typische Fehlermeldungen und ihre Bedeutung

Es gibt nicht viel, was Sie bei der Erstellung von (X)HTML-Dokumenten falsch machen können. Natürlich ist die Arbeit eines Webautors fehleranfällig, aber es handelt sich fast immer um dieselben Arten von Fehlern. Deswegen werden Sie, wenn Sie eine Weile mit dem W3C-Validator gearbeitet haben, das Gefühl haben, stets die gleichen Fehlermeldungen und Warnungen zu lesen. Zu den am häufigsten vorkommenden Fehlern gehören die folgenden:

- *No DOCTYPE Declaration Found!*

  Das Dokument enthält keine **Dokumenttyp-Deklaration** (siehe Kapitel 4.2.2), oder sie ist nicht korrekt.

1 *http://validator.w3.org/*
2 *http://www.htmlhelp.com/tools/validator/*

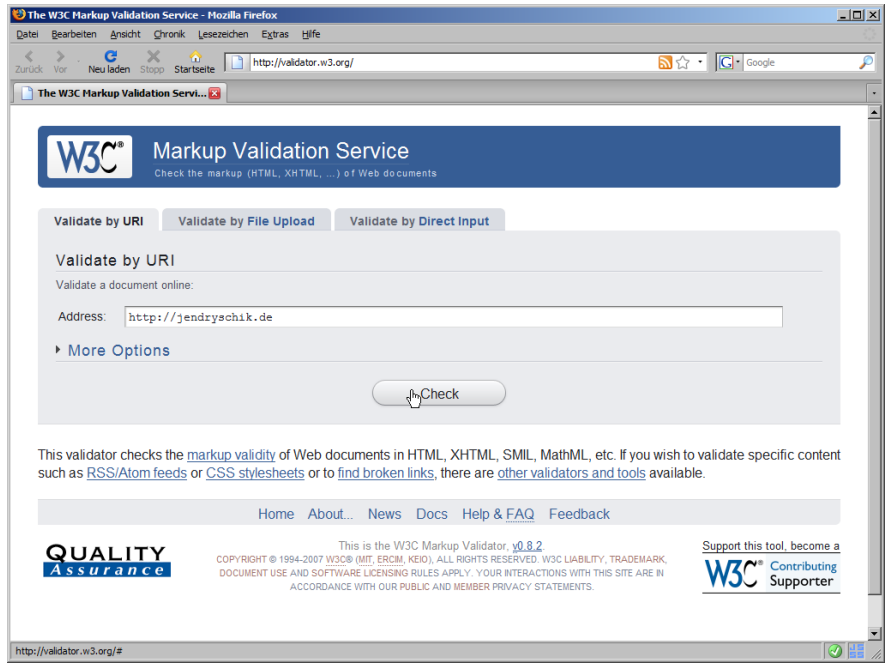


Abbildung 3.5: Der W3C Markup Validation Service

- *No Character Encoding Found!*

  Das Dokument wird ohne Informationen zur Kodierung ausgeliefert. Überprüfen Sie Ihre Webservereinstellungen. Falls Sie lokal testen oder ein Dokument hochgeladen haben, müssen Sie die Kodierungsinformation per `meta`-Element angeben (siehe Kapitel 4.5).

- *Document type does not allow element "p" here; assuming missing "li" start-tag.*

  Ein Element `p` steht im Quelltext an einer Stelle, an die es nicht gehört. Der Validator weist Sie darauf hin, dass offenbar ein Element `<li>` fehlt.

- *End tag for element "p" which is not open.*

  Im Quelltext steht ein `</p>`, für das es kein öffnendes `<p>` gibt. Vielleicht ist der Grund dafür in einer fehlerhaften Elementverschachtelung zu suchen (also beispielsweise `<p><strong>Es ist ein Fehler aufgetreten!</p></strong>`).

- *End tag for "div" omitted, but OMITTAG NO was specified.*

  Im Quelltext steht ein `<div>`, für das es kein schließendes `</div>` gibt. Auch hier könnten Sie Elemente falsch verschachtelt haben, meistens jedoch haben Sie das schließende Tag schlichtweg vergessen.

- *Element "u" undefined.*

  Sie verwenden ein Element (hier `u`), das in der verwendeten (X)HTML-Version nicht enthalten und somit nicht erlaubt ist.

- *There is no attribute "name".*

  Sie verwenden ein Attribut (hier `name`), das in der verwendeten (X)HTML-Version an dieser Stelle nicht erlaubt ist.

- *Required attribute "alt" not specified.*

  Sie haben ein erforderliches Attribut nicht angegeben (hier das `alt`-Attribut des `img`-Elements).

- *Character "<" is the first character of a delimiter but occurred as data.*

  Eine spitze Klammer steht an einer Stelle, an der sie nicht stehen darf. Möglicherweise steht sie irgendwo einsam und allein (beispielsweise `4 < 7`), dann können Sie sie ziemlich einfach finden. Häufiger jedoch ist die Meldung eher irreführend. Der Validator zeigt denselben Fehler an, wenn Sie das abschließende Anführungszeichen eines Attributs vergessen (zum Beispiel `<p class="error>Es ist ein Fehler aufgetreten.</p>`).

- *Unclosed start-tag requires SHORTTAG YES.*

  Sie haben ein **Start-Tag** nicht ordnungsgemäß geschlossen (beispielsweise `<div` anstatt `<div>`).

- *Character "&" is the first character of a delimiter but occurred as data.*

  Das Dokument enthält ein allein stehendes Ampersand (`&`). Weshalb das falsch ist und wie Sie das Zeichen korrekt notieren, lesen Sie in Abschnitt 4.1.4.

- *Cannot generate system identifier for general entity "ap".*

  Sie haben eine benannte Zeichenreferenz (hier `≈`) notiert, die es nicht gibt.

- *XML Parsing Error: EntityRef: expecting ';'.*

  Sie haben eine Zeichenreferenz ohne abschließendes Semikolon notiert (beispielsweise `&160` anstatt `&160;`).

* * * TIPP

Versuchen Sie niemals, die komplette Fehlerliste auf einmal abzuarbeiten, es sei denn, sie besteht nur aus einer Handvoll Meldungen, die Sie unmittelbar zuordnen können. Der W3C-Validator nimmt es sehr genau und gibt für die gleiche Fehlerursache häufig unterschiedliche Meldungen aus. Zudem können Fehler einander bedingen. Beginnen Sie folglich stets am Anfang der Liste, beheben Sie den ersten oder die ersten beiden Fehler und überprüfen Sie Ihr Dokument anschließend erneut. Gehen Sie so Schritt für Schritt vor, bis Ihr Dokument fehlerfrei ist.

Einschränkungen

Ein Validator ist ein gutes Werkzeug, das Webautoren effizient dabei unterstützt, fehlerfreie Dokumente zu erzeugen. Er ist aber lediglich ein Werkzeug. Es ist nicht gesagt, dass Dokumente, die ein Validator als gültig klassifiziert, auch wirklich fehlerfrei im Sinne der Spezifikation sind. Eine DTD kann nicht alle formalen Einschränkungen beschreiben, denen ein Dokumenttyp unterliegt. So lassen sich keine Datentypen wie Integer oder Float definieren; Daten sind praktisch nur Zeichenketten. Das Element

```
<a href="ich bin kein uri">Anker</a>
```

beispielsweise ist unsinnig im Sinne der Empfehlung, die als Wert des `href`-Attributs einen URI fordert, aber es ist gültig im Sinne der DTD, die für das `href`-Attribut den Inhaltstyp CDATA definiert, also lediglich eine Folge von Zeichen. Darüber hinaus soll nicht unerwähnt bleiben, dass es einige merkwürdig anmutende SGML-Konstrukte gibt, die ein SGML-Validator als gültig erkennt, die Darstellung von Dokumenten jedoch gehörig durcheinanderbringt. Nicht nur

```
<p><em>Foo</em></p>
```

ist gültiges HTML. Webautoren dürften auch

```
<p><em/Foo/</p>
```

schreiben (Stichwort: Shorttags), es gibt jedoch kaum ein Benutzerprogramm, das dieses Element dann noch anzeigen kann.

3.2.5 Schema-Validator

XML Schema

Webautoren, die XHTML schreiben, können die meisten Validatoren derzeit nur mit Einschränkungen verwenden. Die Gründe liegen zum einen in den angesprochenen Beschränkungen von DTDs, zum anderen darin, dass es signifikante Unterschiede zwischen SGML und XML gibt, die bisher nicht weitreichend genug beachtet werden. Beim W3C-Validator wird es noch eine Weile dauern, bis er über eine vollständige XML-Unterstützung verfügt. Es gibt

allerdings bereits Abhilfe. Die W3C-Notiz *XHTML 1.0 in XML Schema*[3] vom September 2002 definiert die XHTML-Varianten Strict, Transitional und Frameset als XML Schema, einem XML-Dokumenttyp, mit dem sich die Struktur und die Daten eines XML-Dokuments wesentlich genauer beschreiben und prüfen lassen als mithilfe von DTDs. Es können unter anderem Datentypen sowie genaue Wertebereichsbeschränkungen angegeben und die formale Grammatik von XHTML dadurch deutlich präziser beschrieben werden. Christoph Schneegans stellt einen Schema-Validator[4] bereit, über den XHTML-Dokumente gegen das entsprechende Schema validiert werden können. Dabei werden Fehler aufgedeckt, die herkömmliche Validatoren nicht finden. Der W3C-Validator beanstandet den fehlerhaften Wert des `width`-Attributs nicht:

```
<p><img src="uri" alt="foo" width="bar" /></p>
```

Der Schema-Validator hingegen erkennt den Fehler und wirft eine Fehlermeldung: *The 'width' attribute is invalid. The value 'bar' is invalid according to its datatype 'http://www.w3.org/1999/xhtml:Length'. The Pattern constraint failed.*

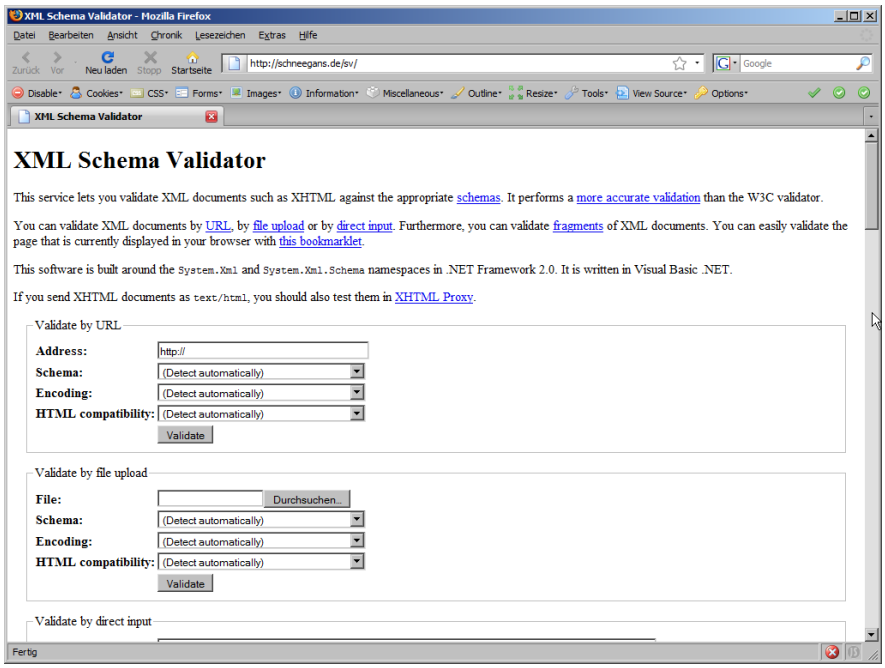


Abbildung 3.6: Der Schema-Validator von Christoph Schneegans

3 *http://www.w3.org/TR/xhtml1-schema*
4 *http://www.schneegans.de/sv/*

3.2.6 Grenzen

Ein weiterer wichtiger Aspekt sollte nicht unerwähnt bleiben: Validatoren überprüfen, ob Dokumente syntaktisch korrekt sind und sich an das vorgegebene Vokabular halten; sie können aber nicht beurteilen, ob Webautoren alle Informationen sinnvoll strukturiert haben oder ob Dokumente barrierefrei und zugänglich sind. Sie könnten auf nahezu alle strukturellen Elemente zugunsten einer reinen `div`- und `span`-Suppe verzichtet oder Dutzende von Layouttabellen ineinander verschachtelt haben – das Dokument wäre gültig. Folgender Schachtelsatz[5] ist in jeder Hinsicht formal korrekt, dabei aber vollkommen unverständlich.

> *»Denken Sie, wie schön der Krieger, der die Botschaft, die den Sieg, den die Athener bei Marathon, obwohl sie in der Minderheit waren, nach Athen, das in großer Sorge, ob es die Perser nicht zerstören würden, schwebte, erfochten hatten, verkündete, brachte, starb!«*

Niemand würde solche Sätze formulieren. Aber es gibt genügend Webautoren, die vergleichbares Markup schreiben. Prägen Sie sich ein: Gültigkeit ist nur ein Aspekt guten Webdesigns – aber längst nicht der wichtigste.[6]

3.2.7 CSS-Validator

Natürlich gibt es auch CSS-Validatoren, also Werkzeuge, die es ermöglichen, Ihre CSS-Stylesheets sowie in XHTML-Dokumente eingebettetes CSS in Bezug auf Standardkonformität zu überprüfen. Der bekannteste und beste CSS-Validator ist der CSS-Validierungsservice des W3C[7]. Wie der *W3C Markup Validation Service* bietet auch er mehrere Möglichkeiten der Validierung.

Bevor Sie Ihr CSS überprüfen, sollten Sie zunächst sicherstellen, dass die Dokumente, auf die Sie das CSS anwenden, korrekt sind. Fehler im XHTML können zur Folge haben, dass unerwünschte und unerwartete Effekte auftreten, obwohl das CSS fehlerfrei ist.

Auch für die CSS-Validierung gilt: Beheben Sie einen Fehler nach dem anderen und führen Sie mehrere Validierungsabläufe durch, bis Ihre Dokumente und Stylesheets fehlerfrei sind.

Weitere Tipps zur Benutzung des W3C-Online-Validators sowie Erläuterungen zu den häufigsten Fehlermeldungen finden Sie auf der Website von Klaus Langenberg unter der Adresse *http://www.thestyleworks.de/basics/validation.shtml*.

---

5 entnommen aus ***[Schneider 2001]***
6 siehe auch ***[Korpela 2008]***
7 *http://jigsaw.w3.org/css-validator/*

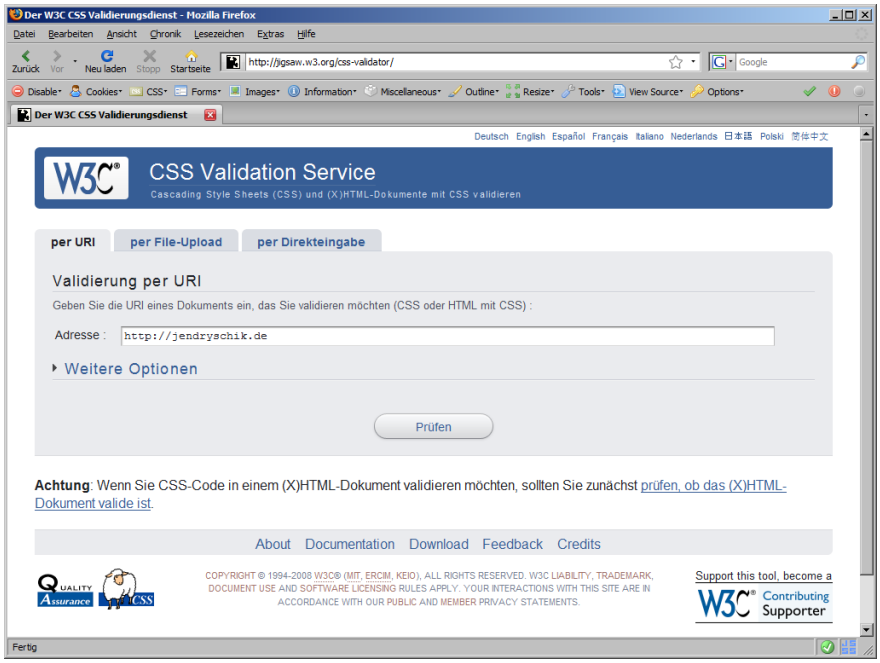


Abbildung 3.7: Der CSS-Validierungsservice des W3C

3.3 Bildbearbeitung

Im Web haben sich vier Dateiformate für Grafiken durchgesetzt: die Formate GIF, JPEG, PNG und – mit Einschränkungen – auch SVG. Die ersten drei Formate beschreiben Rastergrafiken, das vierte ein Vektorformat.

3.3.1 Rastergrafiken

DEFINITION

Rastergrafiken *(auch: Pixelgrafiken, Bitmaps) sind eine Methode zur Beschreibung zweidimensionaler Bilder und bestehen aus einer matrixförmigen Anordnung von Pixeln, denen jeweils eine Farbe zugeordnet ist. Die Hauptmerkmale einer Rastergrafik sind die Breite und die Höhe in Pixeln sowie die Farbtiefe. Rastergrafiken eignen sich zur Repräsentation komplexer Bilder, beispielsweise Fotos.*

GIF

DEFINITION

GIF *(*Graphics Interchange Format*) ist ein digitales Bildformat mit guter, verlustfreier Komprimierung für Bilder mit geringer Farbtiefe (bis zu 256 Farben). Das Format wurde 1987 vom US-Onlinedienst* CompuServe *eingeführt und ist heute vor allem wegen seiner effizienten Kompression LZW (Lempel-Ziv-Welch-Algorithmus) populär.*

GIF ermöglicht das Speichern mehrerer Bilder in einer einzigen Datei, was vor allem für einfache Animationen verwendet wird. Darüber hinaus ist es möglich, eine Farbe der GIF-Grafik als transparent zu definieren, das heißt, alle Elemente, die unter der Grafik liegen, scheinen an den Stellen durch, an denen die Grafik transparent ist.

Anwendung

GIF ist aufgrund seiner geringen Farbtiefe nicht für hochauflösende Grafiken und Fotos geeignet, dagegen hervorragend für weniger aufwendige Grafiken wie Hintergrundverläufe, Icons, Schaltflächen, Symbole und Zeichen.

JPEG

DEFINITION

JPEG *ist ein 1992 durch die* Joint Photographic Experts Group *entwickeltes und nach ihr benanntes standardisiertes Verfahren zur sowohl verlustbehafteten wie verlustfreien Kompression von digitalen Grafiken.*

Anwendung

Das Format unterstützt Grafiken mit hohen Farbtiefen (24 Bit) und eignet sich damit optimal zum Speichern von Fotos. Nicht geeignet ist JPEG zur Speicherung von Grafiken, in denen es viele benachbarte Bildpunkte mit exakt identischen Farbwerten, wenigen Farben und harten Kanten gibt (zum Beispiel Strichzeichnungen), sowie von reinen Schwarz-Weiß-Bildern ohne Grauabstufungen.

Komprimierung

JPEG bietet ein standardisiertes, verlustbehaftetes Kompressionsverfahren, den *Baseline Codec*, der im Wesentlichen wie folgt arbeitet: Es wird nach redundanten Informationen gesucht, wie zum Beispiel Ähnlichkeiten oder sich wiederholenden Mustern, die dann herausgerechnet werden, möglichst ohne die Bildqualität zu verringern. Dadurch lässt sich die Dateigröße um rund 90 Prozent reduzieren, ohne dass das ungeschulte Auge eine Verringerung der Qualität bemerken würde. Eine Kompression auf 60 Prozent ist ein guter Richtwert.

PNG

DEFINITION

PNG *(*Portable Network Graphics*) ist ein universelles Format, das die Vorteile von GIF und JPEG vereint und über noch weitere interessante Eigenschaften verfügt. Im Jahr 1995 wurde mit der Entwicklung begonnen. Im März 1997 wurde PNG als RFC 2083* ***[RFC-2083 1997]*** *veröffentlicht. Die aktuelle W3C-Empfehlung PNG 1.1 stammt vom Januar 2003* ***[REC-PNG 2003]****.*

Eigenschaften

PNG kann wie GIF Pixel aus einer Farbpalette mit bis zu 256 Einträgen verarbeiten. Darüber hinaus ist die Speicherung von Graustufenbildern mit 1, 2, 4, 8 oder 16 Bit und Farbbildern mit 8 oder 16 Bit pro Kanal (also 24 beziehungsweise 48 Bit pro Pixel) möglich. PNG ermöglicht das Abspeichern zusätzlicher Informationen in der Grafikdatei, zum Beispiel Autoren- und Urheberhinweise, und unterstützt Farbkorrekturmechanismen, die sicherstellen, dass ein Bild auf verschiedenen Systemen annähernd gleich aussieht. Während es bei GIF nur möglich ist, einer einzigen Farbe vollständige Transparenz zuzuweisen, können PNG-Grafiken Transparentinformationen für jedes einzelne Pixel oder für jede Farbe der Farbpalette enthalten. Dabei sind 8 oder 16 Bit, also 256 oder 65.536 Transparenzabstufungen möglich.

Browserunterstützung

Veraltete Browser, darunter der Internet Explorer bis Version 6, haben Probleme mit der Darstellung von Transparenzen bei PNG-Grafiken. Aus diesen Gründen konnte sich das Format lange nicht so durchsetzen, wie es zu wünschen gewesen wäre. Mittlerweile kann man von einer guten PNG-Unterstützung in aktuellen Browsern sprechen, und für den IE 6 gibt es mehrere **Workarounds**. Zu den besten gehört PNG Behavior[8].

Software

Auch Webautoren benötigen Programme zum Erstellen und Bearbeiten digitaler Fotos und Grafiken, schließlich kommt kaum eine professionelle Website ohne Grafiken aus – angefangen beim Logo über Farbverläufe, Schatteneffekte und abgerundete Ecken bis hin zu Symbolen, Illustrationen und Fotografien. Auch wenn es nur darum geht, fertige Designvorlagen zu »schneiden«, das heißt einzelne grafische Elemente freizustellen, passend zu skalieren und im geeigneten Format abzuspeichern, benötigt man ein Bildbearbeitungsprogramm, mit dem man diese Arbeiten verrichten kann.

Adobe Photoshop

Das wichtigste Programm im Bereich der professionellen Bildbearbeitung ist *Adobe Photoshop* des US-amerikanischen Softwareunternehmens *Adobe Systems*.

8 *http://webfx.eae.net/dhtml/pngbehavior/pngbehavior.html*

Aufgrund seines hohen Preises (rund 1.000 Euro) ist es für Privatanwender jedoch nahezu unerschwinglich. Für knapp ein Zehntel des Preises gibt es mit *Adobe Photoshop Elements* mittlerweile eine abgespeckte, deutlich günstigere Version, die für die Belange von Heimanwendern optimiert wurde. Gegenüber Photoshop fehlen zwar einige Funktionen, aber Anwender, die wie Webautoren ohnehin nur einen kleinen Teil der Funktionen benötigen, haben dadurch keinen Nachteil. Für Photoshop steht eine Vielzahl hilfreicher Einsteiger-Tutorials im Web zur Verfügung, unter anderem bei Webmasterpro.de[9] oder PSD-Tutorials.de[10].

GIMP

Eine kostenlose Alternative ist *GIMP* (*GNU Image Manipulation Program*), ein Bildbearbeitungsprogramm, das als freie Software unter der GNU General Public Licence veröffentlicht wird und als ein Vorzeigeprogramm der Open-Source-Bewegung angesehen wird. Der Leistungsumfang ist mit dem Adobe Photoshops vergleichbar. Bekannt wurde das Programm hauptsächlich dadurch, dass Larry Ewing den Pinguin Tux, das bekannte Linux-Maskottchen, mit GIMP entwickelt hat. Anfänglich arbeiteten nur zwei Programmierer, die kalifornischen Informatikstudenten Peter Mattis und Spencer Kimball, an dem Projekt, mittlerweile geht die Anzahl der Entwickler und Tester in die Tausende.

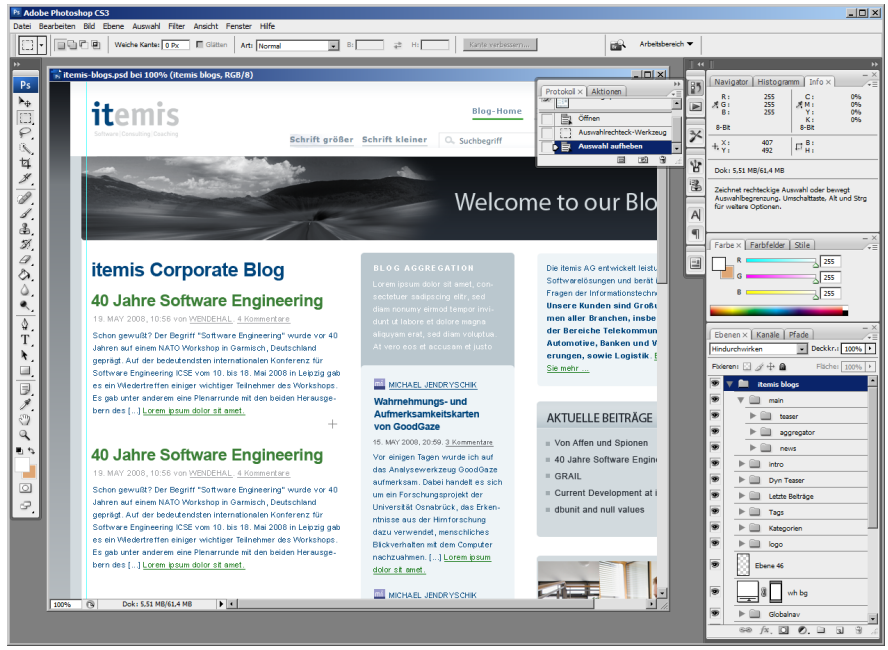


Abbildung 3.8: Adobe Photoshop CS3

9 *http://www.webmasterpro.de/design/photoshop.html*
10 *http://www.psd-tutorials.de*

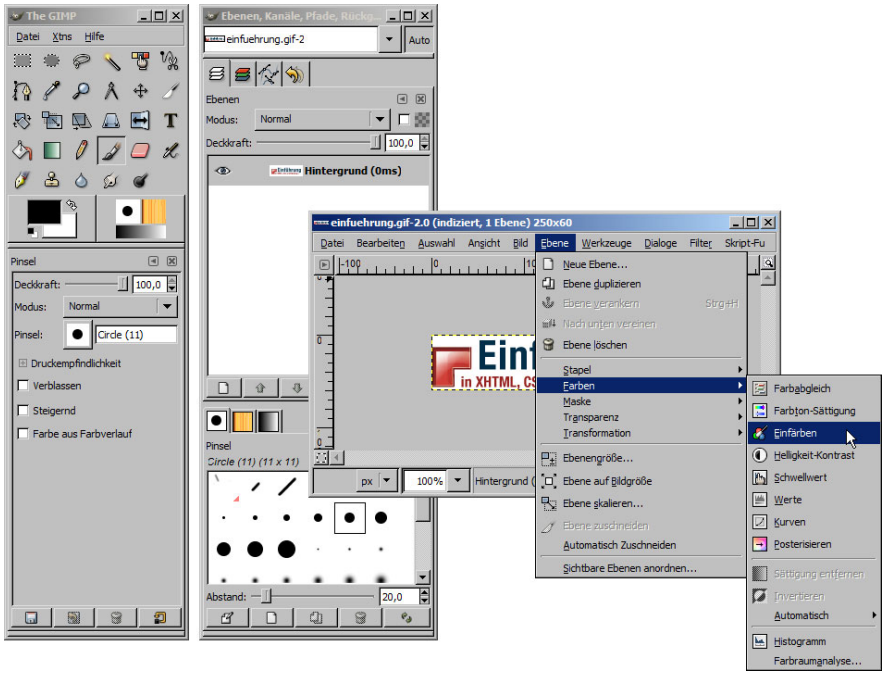


Abbildung 3.9: Das freie Bildbearbeitungsprogramm GIMP

Die aktuelle Version können Sie von der GIMP-Homepage unter der Adresse *http://www.gimp.org* herunterladen. Das deutschsprachige Benutzerhandbuch leistet Unterstützung bei der Einarbeitung. Sie finden es unter der Adresse *http://docs.gimp.org/de/*.

3.3.2 Vektorgrafiken

DEFINITION

*Eine **Vektorgrafik** ist ein zwei- oder dreidimensionales Computerbild, das aus sogenannten grafischen Primitiven wie Linien, Kreisen und Polygonen geometrisch modelliert ist. Um beispielsweise einen Kreis zu zeichnen, braucht es lediglich Informationen über die Koordinaten des Kreismittelpunkts sowie die Länge des Durchmessers. Diese Informationen bleiben im Bild erhalten und können gegebenenfalls durch weitere ergänzt werden, beispielsweise Informationen über Strich- und Füllfarben sowie Muster.*

Anwendung

Vektorgrafiken können im Gegensatz zu Rastergrafiken ohne Qualitätsverlust stufenlos skaliert und verzerrt werden. Allerdings sind sie ungeeignet für die Darstellung komplexer Grafiken wie Fotos, da diese sich kaum mathematisch modellieren lassen.

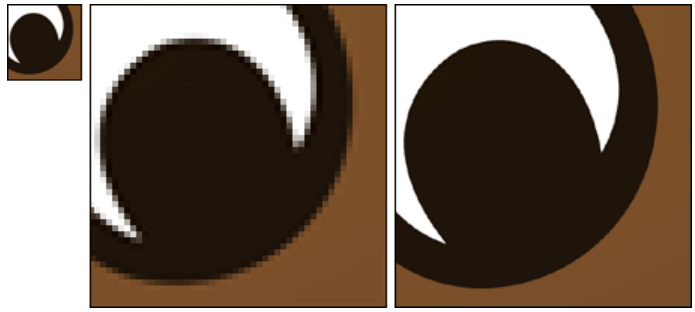


Abbildung 3.10: Rastergrafiken werden bei Vergrößerung pixelig, Vektorgrafiken hingegen nicht

Im Web haben sich zwei vektorgrafische Formate durchgesetzt: Flash, ein Grafik- und Animationsformat der amerikanischen Firma Macromedia, das hier nicht weiter behandelt wird, und SVG.

SVG

DEFINITION

SVG *wurde im September 2001 als W3C-Empfehlung* Scalable Vector Graphics (SVG) 1.0 Specification *veröffentlicht. Aktuell ist Version 1.1 vom September 2003* ***[REC-SVG 2003]****. SVG-Dateien sind XML-Dokumente und können daher – wie XHTML-Dokumente – mit einem gewöhnlichen Texteditor erstellt und bearbeitet werden.*

Ressourcen

Einführungen wie Ralf Pohlmanns SVG-Tutorial[11], Marco Rosenthals SELFSVG[12] oder die Site scale-a-vector.de von Petra Kukofka[13] erleichtern den Einstieg. Dr. Thomas Meinikes *Learning By Coding*[14] zeigt anhand vieler einfacher Beispiele, was mit SVG so alles möglich ist.

Es gibt mittlerweile spezielle Programme wie das kostenlos verfügbare Inkscape[15], die die Tipparbeit zum größten Teil abnehmen und die Erstellung von SVG-Grafiken deutlich vereinfachen.

11 *http://svg.tutorial.aptico.de*
12 *http://www.selfsvg.info*
13 *http://www.scale-a-vector.de*
14 *http://svglbc.datenverdrahten.de*
15 *http://www.inkscape.org/*

Abbildung 3.11: Mit Inkscape lassen sich verhältnismäßig einfach anspruchsvolle SVG-Grafiken erstellen

Browserunterstützung

Viele Browser können SVG darstellen, dazu gehören Opera ab Version 8, Safari 3, Konqueror sowie Firefox ab Version 1.5. Andere Browser benötigen ein spezielles Plug-in wie den kostenlosen *Adobe SVG Viewer*[16], dessen Entwicklung jedoch Anfang 2009 eingestellt wird. Aufgrund des enormen Potenzials ist es allerdings nur eine Frage der Zeit, bis Alternativen zur Verfügung stehen werden. Der Renesis Player[17] der examotion GmbH könnte eine solche sein.

Dadurch dass SVG ein Klartextformat ist, können Programmierer und Grafiker SVG-Grafiken serverseitig generieren, sie mit Datenbanken kommunizieren lassen, die DOM-Struktur (siehe Kapitel 4.3) einer Grafik clientseitig per Script manipulieren, auf Benutzereingaben reagieren und vieles mehr. SVG-Grafiken können auch aus anderen XML-Formaten generiert werden.

16 *http://www.adobe.com/svg/viewer/install/main.html*
17 *http://www.examotion.com*

3.4 Nützliche Werkzeuge

3.4.1 Tidy

DEFINITION

Tidy *(heißt so viel wie sauber, ordentlich) ist ein kleines und wirkungsvolles Werkzeug, das hauptsächlich dazu dient, (X)HTML-Dokumente zu prüfen und auf Wunsch automatisch so aufzubereiten, dass sie der Prüfung eines Validators (siehe Kapitel 3.2) standhalten. Tidy erkennt fehlerhafte Syntax, falsche oder redundante Auszeichnungselemente und beseitigt je nach Konfiguration die Fehler automatisch. Das Werkzeug bietet erstaunlich viele Funktionen und ist gut konfigurierbar.*

Funktionen

- Auf Wunsch konvertiert Tidy HTML in XHTML und umgekehrt,
- ändert die Zeichenkodierung,
- wandelt Attribute zur visuellen Auszeichnung von Inhalten, beispielsweise Angaben zu Schrift oder Farbe, in CSS-Angaben um,
- entfernt unerwünschte und veraltete Elemente,
- formatiert den Quelltext neu,
- korrigiert durch sogenannte WYSIWYG-Editoren (siehe Kapitel 3.1.1) produziertes, teilweise unsauberes oder gar fehlerhaftes (X)HTML und
- überführt durch Microsoft Word erzeugte HTML-Dokumente – in der Regel haarsträubend verkompliziertes HTML, durchsetzt von für viele Browser nutzlosen proprietären Auszeichnungselementen – in schlanke, gültige Dokumente.

Unter der Adresse *tidy.sourceforge.net* können Sie Tidy herunterladen. Beachten Sie, dass zwar ab und zu binäre Versionen von Tidy bereitgestellt werden, allerdings nicht regelmäßig und nicht für alle Plattformen. Um immer die aktuelle Version im Einsatz zu haben, nutzen Sie daher stets die allerneueste Version aus dem CVS des Projekts. Näheres dazu finden Sie auf der betreffenden CVS-Webseite des Tidy-Projekts: *http://sourceforge.net/cvs/?group_id=27659*. Die Projektverantwortlichen pflegen kein Versionierungsschema. Die Version von Tidy ist immer das Datum der letzten Änderung.

Entwicklung

Ursprünglich wurde Tidy von Dave Raggett, dem (Ko-)Autor von HTML 3 bis 4.01, als »HTML Tidy« unter dem Dach des W3C entwickelt. Der inzwischen verstorbene Terry Teague prägte in der Anfangszeit das Projekt durch eigenen Code entscheidend mit und half, es auf mehrere Plattformen zu portieren. Maßgeblich treiben inzwischen Arnaud Desitter und Björn Höhrmann in

enger Abstimmung mit dem W3C das Projekt als Open Source voran. Einige Entwickler arbeiten parallel engagiert in anderen Projekten des Konsortiums mit, unter anderem am *Markup Validation Service*, wodurch Webentwickler mehr und mehr durch einheitliche Fehlererkennung und -behandlung beider Werkzeuge profitieren.

TidyLib

Die *TidyLib*, das zentrale Herz aller Tidy-Erscheinungsformen, ist für viele Plattformen in unterschiedlichen Ausführungen erhältlich. Die wohl meistverbreitete und ursprünglichste Erscheinungsform ist die als eigenständiges Programm, das vor allem für Windows erhältlich ist. Aber auch verschiedene Linux-Distributoren bieten Tidy an, ebenso wie es mit *Balthisar Tidy*[18] eine binäre Cocoa-Version für Mac OS X gibt. In vielen WYSIWYG- und Texteditoren ist Tidy integriert oder lässt sich leicht manuell einbinden. Einigen aktuellen Linux-Distributionen liegt es als Paket bei. Für Mozilla und Firefox gibt es eine hervorragende Erweiterung namens Html Validator, die im nächsten Abschnitt, »Browsererweiterungen«, vorgestellt wird.

Automatisierter Einsatz

Meist nutzen Webautoren Tidy zur Bearbeitung einzelner Dokumente. Alle Dokumente einer Website auf diese Weise zu überprüfen, kann bei größeren Projekten mit Hunderten oder Tausenden von einzelnen Dokumenten umständlich sein. Tidy lässt sich jedoch automatisiert einsetzen, im einfachsten Fall über die Kommandozeile. Im Folgenden sei davon ausgegangen, dass Tidy für Windows als `C:\tidy\tidy.exe` vorliegt und im Suchpfad für ausführbare Programme eingetragen ist. Das Kommando

```
tidy -f ausgabe.txt -q foo.html
```

schreibt Fehlermeldungen und Warnungen in eine Datei und blendet die weiteren Informationszeilen zur Datei und zu Tidy aus. Bei einem fehlerfreien Dokument ist `ausgabe.txt` leer. Eine Überprüfung von Listing 3.1 führt zu einer Ausgabe, die unter anderem die Warnungen `missing <!DOCTYPE> declaration` und `missing </title> before </head>` enthält. Eine automatische Korrektur der untersuchten Datei erledigt der Parameter `-m`.

```
<html>
  <head><title>Tidy Beispiel</head>
  <body>Hallo Welt!</body>
</html>
```

Listing 3.1: Fehlerhaftes HTML-Dokument

18 *http://www.balthisar.com/products/tidy*

Sämtliche Konfigurationsoptionen sind in der Quick Reference[19] aufgeführt. Sie lassen sich jedem Aufruf direkt mitgeben oder, komfortabler, in einer Konfigurationsdatei notieren. Listing 3.2 zeigt eine solche. Die Angabe

```
tidy -m -config c:\tidy\tidy.cfg foo.html
```

wandelt Listing 3.1 in das XHTML-Dokument in Listing 3.3 um.

```
doctype: strict
enclose-text: true
indent: true
indent-spaces: 2
language: de
output-xhtml: true
tab-size: 2
tidy-mark: false
write-back: true
```

Listing 3.2: Tidy-Konfigurationsdatei

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN"
    "http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd">
<html xmlns="http://www.w3.org/1999/xhtml">
  <head>
    <title>
      Tidy Beispiel
    </title>
  </head>
  <body>
    <p>
      Hallo Welt!
    </p>
  </body>
</html>
```

Listing 3.3: Listing 1 nach Tidys Korrektur

Folgende Zeile wendet Tidy auf alle Dokumente mit der Dateinamenserweiterung `html` im aktuellen Verzeichnis und allen Unterverzeichnissen an – unter Berücksichtigung einer Konfigurationsdatei:

```
for /R %f in (*.html) do tidy.exe -m -config C:\tidy\tidy.cfg %f
```

Weitere Portierungen

Es geht jedoch produktiver. Viele Webentwickler nutzen für ihre Projekte eine lokale Entwicklungsumgebung, vor allem für Testzwecke. Idealerweise spiegelt sie die Bedingungen der späteren Serverumgebung für die jeweiligen Webprojekte wider. Zwei Alternativen bieten sich für solche Umgebungen an:

19 *http://tidy.sourceforge.net/docs/quickref.html*

Tidy als PHP- und als Webservererweiterung. In beiden Fällen durchläuft die (X)HTML-Ausgabe automatisch zunächst die Tidy-Bibliothek, wird von ihr geprüft und je nach Konfiguration überarbeitet. Entwickler oder Anwender merken bei fehlerfreien Dokumenten im Idealfall nichts von alledem, höchstens eine leichte zeitliche Verzögerung. Dies stellt eine große Arbeitserleichterung dar und ermöglicht ein zügiges und kostensparendes Entwickeln von Webseiten.

Dank John Coggeshall existiert Tidy als PECL-Erweiterung[20] für die Versionen 4.3+, die als *shared module* eingebunden wird. Seit PHP 5.0+ ist diese Tidy-Erweiterung Bestandteil der PHP-Distribution und muss nur noch in der Konfiguration aktiviert werden (vorausgesetzt, die Tidylib- beziehungsweise libtidy-Bibliothek ist installiert). Erfahrene PHP-Programmierer können Tidy daher leicht in ihre Projekte integrieren.

Sebastian Tusk hat vor Jahren ein Modul für den Apache 2 Webserver entwickelt: *mod_tidy*[21]. Da Tidy als PHP-Erweiterung nur in einer PHP-Entwicklungsumgebung funktionieren kann, bringt der Einsatz als Apache-Modul größeren Nutzen, weil es auf diese Weise jede Art von Webdokument automatisch überprüfen kann und sein Einsatz nicht auf PHP-Dokumente beschränkt bleibt. Mittlerweile hat Sierk Bornemann die Arbeit an *mod_tidy* übernommen.

Im Java Server Faces AJAX Framework ist eine gut gepflegte Tidy-Klasse enthalten, um innerhalb von Java Server Faces die Eigenschaften von Tidy zu nutzen. Für Java existiert mit Jtidy[22] außerdem eine direkte Tidy-Portierung, doch leider wird diese seit Längerem nicht mehr gepflegt und ist deswegen, was die Aktualität des Tidy-Codes betrifft, nicht mehr auf der Höhe der Zeit. Sie sei hiermit der Vollständigkeit halber trotzdem erwähnt.

Für die populäre Entwickler-IDE Eclipse existiert mit dem Tidy-Eclipse-Plugin[23] eine Möglichkeit, Eclipse um die Prüfmöglichkeiten von Tidy zu bereichern.

Ein Webautor, der lediglich ab und zu einzelne Webseiten erstellt, ist mit dem eigenständigen Programm oder der Version als Browsererweiterung gut bedient. Für solche Fälle lohnt sich der Aufbau einer lokalen Entwicklungsumgebung meistens nicht, oder der Autor könnte damit überfordert sein. Für Webautoren, die regelmäßig größere Webprojekte zu bearbeiten haben, sind die Eigenschaften der PHP-Erweiterung, des Apache-Moduls, der betreffenden Java-Klassen oder einfach der Firefox-Erweiterung im Alltag hilfreich.

20 *http://www.coggeshall.org/oss/tidy/index.php/4/*
21 *http://mod-tidy.sourceforge.net*
22 *http://jtidy.sourceforge.net*
23 *http://eclipsetidy.sourceforge.net*

Wenigstens eines dieser Werkzeuge zum Kontrollieren und gegebenenfalls Korrigieren der HTML-Syntax sollte in keiner professionellen Entwicklungsumgebung fehlen.

3.4.2 FTP

Ihre Webseiten erstellen Sie höchstwahrscheinlich lokal auf Ihrem Rechner. Damit ist es unwahrscheinlich, dass Sie – wie es ja im Allgemeinen gewünscht ist – für jeden Nutzer zu jeder Zeit erreichbar sind; schließlich wollen Sie Ihren Rechner gewiss nicht 24 Stunden pro Tag durchlaufen lassen und permanent online sein. In der Regel mieten Sie entsprechenden Speicherplatz bei einem Hosting-Provider an, der über eine entsprechende Technik und Anbindung verfügt. Dazu zählen in der Regel vollklimatisierte Serverräume, Brandschutz, unterbrechungsfreie Stromversorgung, Netzwerküberwachung und regelmäßige Backups. Es gibt eine Fülle von Anbietern unterschiedlicher Qualität, die sich in Preis und Leistung zum Teil deutlich unterscheiden.

Um die lokal auf Ihrer Festplatte liegenden Dateien Ihrer Website zu publizieren, das heißt der Allgemeinheit zugänglich zu machen, müssen Sie diese auf den Ihnen zugewiesenen Speicherplatz auf den Servern Ihres Hosting-Providers überspielen. Das klassische Protokoll für diesen Zweck ist FTP.

DEFINITION

FTP *(*File Transfer Protocol*) ist ein Netzwerkprotokoll zur Übertragung von Dateien vom Server zum Client (Download), vom Client zum Server (Upload) oder clientgesteuert zwischen zwei Servern.*

Von Ihrem Provider erhalten Sie alle erforderlichen Daten, die Sie benötigen, um Ihre Dateien auf den Server übertragen, sie von dort auf Ihren Rechner herunterladen oder direkt auf dem Server verwalten zu können. Zusätzlich benötigen Sie einen FTP-Client, in den meisten Fällen ein Programm, das so aufgebaut ist wie ein Dateimanager, beispielsweise der Windows Explorer.

SFTP

Falls Sie Wert auf mehr Sicherheit legen, sollten Sie auf SFTP (SSH File Transfer Protocol), eine Art »sicheres FTP«, und entsprechende Clients zurückgreifen.

FileZilla

FileZilla[24] gehört zu den besten FTP-Progammen unter Windows, ist kostenlos erhältlich und bietet auch SFTP-Unterstützung. Für Macintosh-Rechner ist *Cyberduck*[25] empfehlenswert.

24 *http://filezilla.sourceforge.net*
25 *http://cyberduck.ch*

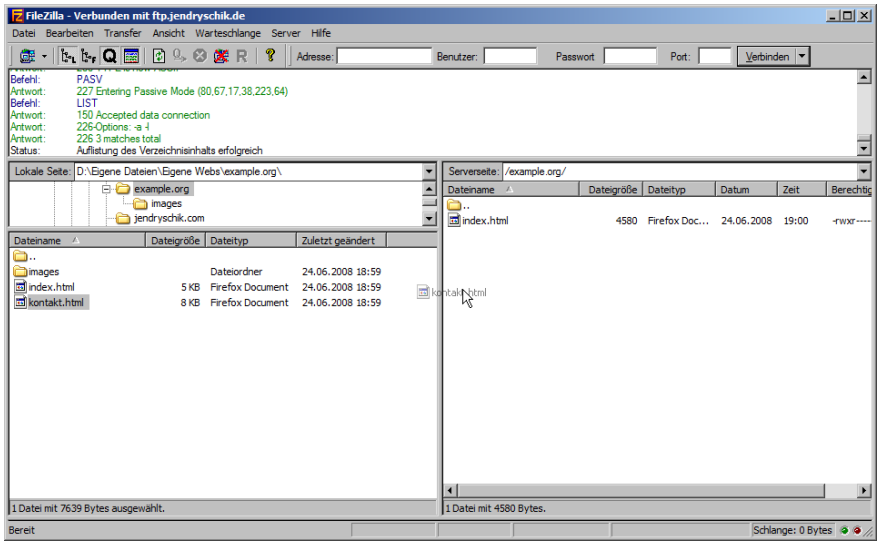


Abbildung 3.12: Der FTP-Client FileZilla

Literatur

Jürgen Schlottke: FileZilla Dokumentation: FTP-Schnelleinstieg
Diese Anleitung zeigt Ihnen in aller Kürze die Bedienung des FileZilla-FTP-Clients.
URL: *http://www.filezilla.de/schnelleinstieg.htm*

Tim Kosse: FileZilla documentation
Vollständige englischsprachige FileZilla-Dokumentation.
URL: *http://filezilla.sourceforge.net/documentation/*

3.4.3 Xenu's Link Sleuth

Wer kennt das nicht? Man ruft eine Webressource auf – und vollkommen unerwartet findet man sich im Niemandsland des Webs wieder. Dafür kann es folgende Gründe geben:

- Der URI der Ressource wurde falsch eingetippt, entweder in der Adresszeile oder beim Anlegen des Links.
- Der Autor der Site wusste nicht, dass sich coole URIs nicht ändern,[26] und hat die Ressource gelöscht oder verschoben. Dadurch stimmen Lesezeichen, Links auf die Site sowie Einträge in Suchmaschinen nicht mehr.

26 vgl. *http://www.w3.org/Provider/Style/URI.html*

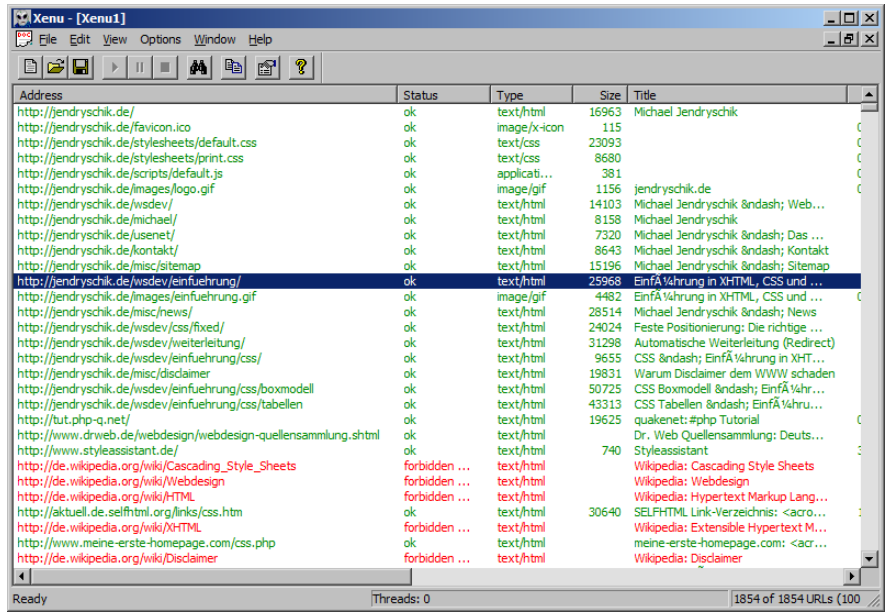


Abbildung 3.13: Xenu's Link Sleuth

Um tote Links innerhalb Ihrer Website aufzufinden, brauchen Sie nicht Dokument für Dokument einzeln zu durchsuchen und jeden Link auszuprobieren. Mit *Xenu's Link Sleuth*[27] von Tilman Hausherr steht ein Programm zur Verfügung, das dies automatisch und sehr komfortabel für Sie erledigt. Auf Knopfdruck können Sie eine komplette Website nach ungültigen Verweisen durchsuchen. Das Programm kann auch mit SSL verschlüsselte sowie lokale Webseiten überprüfen – gerade letzterer Umstand macht das Werkzeug so nützlich. Aus den gewonnenen Informationen wird eine Zusammenfassung in Form eines HTML-Dokuments erstellt, das im Browser angezeigt und ausgewertet werden kann.

Eine deutschsprachige Anleitung[28] von Christian Holm hilft Ihnen beim Einstieg.

3.4.4 Color Cop

Color Cop[29] ist ein Farbpicker, mit dem Sie mittels einer Pipette einen Farbwert eines beliebigen Punkts auf dem Desktop messen können. Der Farbwert wird als RGB- wie auch als Hexadezimalwert angegeben. Eine Bildschirmlupe hilft Ihnen dabei, den richtigen Bildschirmpunkt zu treffen.

27 *http://home.snafu.de/tilman/xenulink.html*
28 *http://www.aspheute.com/artikel/20010518.htm*
29 *http://colorcop.net*

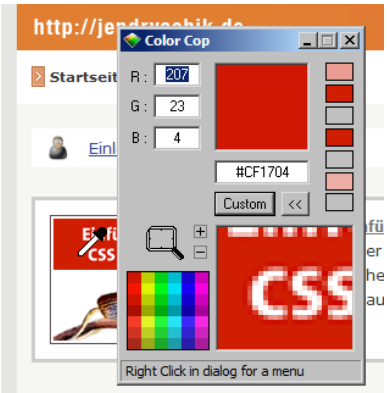


Abbildung 3.14: Color Cop

Color Cop erfüllt nur diesen einen Zweck und ist entsprechend einfach zu bedienen. Dabei gehört das Programm zu den nützlichsten Hilfsmitteln, die Sie sich als Webentwickler auf Ihrem Arbeitsrechner installieren können.

3.5 Browsererweiterungen

DEFINITION

*Eine **Browsererweiterung** (*Add-on*, auch: Plug-in) ist ein optionales Modul, das Browserfunktionen ergänzt oder erweitert. Browserhersteller definieren Schnittstellen, mit deren Hilfe Erweiterungen Dritter eingebunden werden können. Plug-ins sind in der Regel ohne die Anwendung, für die sie geschrieben wurden, nicht funktionsfähig.*

Auch Programme wie der *Flash Player* oder *Shockwave* sowie das *Reader*-Plug-in zur Anzeige von PDF-Dokumenten oder der *SVG Viewer* der Firma Adobe sind Browsererweiterungen. Im Folgenden geht es jedoch nicht um Programme, die zur Anzeige bestimmter Multimedia- oder Dokumentformate bestimmt sind, sondern um eine andere Art der Browsererweiterung: Es handelt sich hier um Erweiterungen, die den Browser um nützliche Funktionen für Webautoren ergänzen.

Firefox

Erweiterungen gibt es für alle großen Browser, wirklich interessant sind jedoch nur die Erweiterungen für Browser der Mozilla-Familie, speziell für Firefox. Auf der Seite *https://addons.mozilla.org* finden Sie Hunderte von Erweiterungen aller Art. Auf Erweiterungen für andere Browser wird in diesem Abschnitt nicht eingegangen.

3.5.1 Firebug

Die Browsererweiterung Firebug[30] von Joe Hewitt, einem der Firefox-Entwickler, ermöglicht Webautoren, Webseiten zu »debuggen«, das heißt nach Fehlern zu suchen und diese zu korrigieren. Firebug integriert sich in Form einer Konsole in der unteren Browserhälfte – links ist der Quelltext der Seite in Form einer auf- und zuklappbaren Baumstruktur zu sehen, rechts alle (auch die vererbten) CSS-Eigenschaften. Damit können Sie anfangen, den Code Ihrer Webseite zu inspizieren. Fahren Sie mit der Maus über die Seite, und Firebug markiert den dazugehörigen Code – oder umgekehrt. Das Besondere an Firebug ist, dass Sie den Quellcode nicht nur inspizieren, sondern auch ändern können. Dazu bietet das Werkzeug die Funktion »Edit«, über die Sie markierte Teile des Quelltext korrigieren, beispielsweise CSS per Inline-Style hinzufügen können. Die Auswirkungen der Änderungen sehen Sie – und nur Sie (das ist wichtig zu wissen) – direkt live auf der Seite.

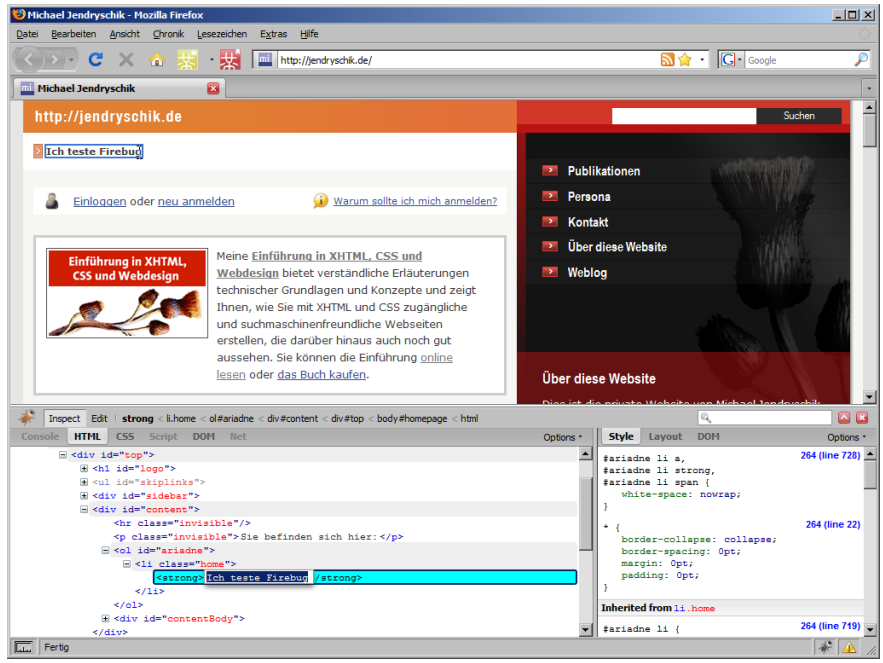


Abbildung 3.15: Mit Firebug können Webautoren live sehen, wie sich Änderungen am Quelltext auswirken

30 *https://addons.mozilla.org/de/firefox/addon/1843*

3.5.2 Web Developer Toolbar

Die *Web Developer Toolbar*[31] von Chris Pederick gehört zu den Browsererweiterungen, auf die Sie bereits nach kurzer Zeit nicht mehr werden verzichten wollen. Für Webautoren ist ihre Installation Pflicht. Die Web Developer Toolbar basiert auf der *PNH-Developer-Toolbar*[32] von Chris Casciano und verfügt über eine derartige Vielzahl von Diagnosewerkzeugen und nützlichen Hilfsmitteln, dass es eine Stunde dauern kann, jede einzelne Funktion an seiner Website auszuprobieren. Zu den interessantesten Funktionen gehören die Möglichkeiten,

Funktionen

- JavaScript, Java, Cookies, den Browsercache und vieles andere mit einem Mausklick aus- und wieder einzuschalten,
- Stylesheets vollständig oder teilweise auszuschalten sowie live zu editieren (Änderungen sind direkt auf der Webseite sichtbar, auch wenn es sich nicht um eine lokale Site handelt),
- das Verhalten von Formularen zu beeinflussen,
- Informationen zu allen Bildern auf der Webseite anzuzeigen sowie die Darstellung von Bildern zu unterbinden oder auf die Anzeige ihres alternativen Attributs zu beschränken,
- das zugrunde liegende (X)HTML beziehungsweise DOM zu manipulieren und so versteckte Elemente und Kommentare anzuzeigen oder ein Dokument zu linearisieren,

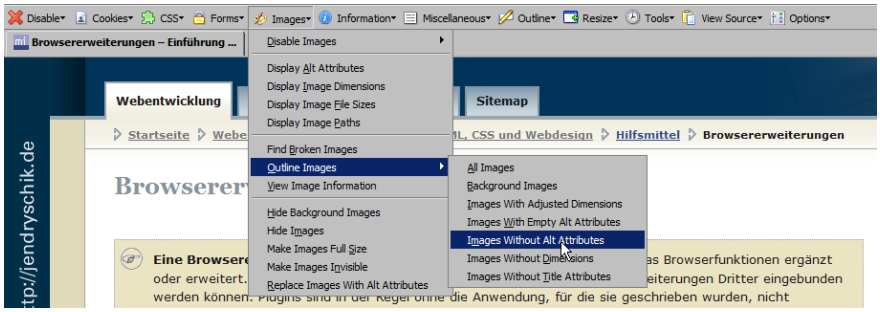


Abbildung 3.16: Die Web Developer Toolbar in Aktion

31 *http://chrispederick.com/work/webdeveloper/*
32 *http://www.placenamehere.com/pnhtoolbar/*

- Elemente zur Analyse sichtbar zu machen, etwa alle Tabellen, alle Blockelemente oder alle missbilligten Elemente, sowie
- das aktuelle Dokument – auch lokale Dokumente – direkt einer Vielzahl von Validatoren und Prüfwerkzeugen vorzulegen.

3.5.3 Html Validator (based on Tidy)

Die Erweiterung *Html Validator (based on Tidy)*[33] von Marc Gueury ergänzt die Quelltextanzeige des Firefox-Browsers um einige nützliche Funktionen. So lassen sich automatisch Fehler und Warnungen anzeigen und in einem weiteren Schritt korrigieren.

Lange Zeit als reine Implementation der TidyLib im Einsatz, beherbergt dieses Firefox-Plug-in neben der TidyLib zum Überprüfen der HTML-Syntax mittlerweile auch OpenSP, eine weitere Syntaxbibliothek aus dem Dunstkreis des W3C. OpenSP ist derselbe SGML-Parser, der im *W3C Markup Validation Service* zum Einsatz kommt. Dadurch dass dieses Firefox-Add-on nun zwei mächtige »Syntaxprüfer« an Bord hat, ist es zwar etwas größer als in der Vergangenheit, dafür aber doppelt so mächtig. Diese beiden Syntaxprüfer können, je nach Konfiguration, entweder einzeln verwendet oder sogar miteinander verkettet werden.

3.5.4 Fangs

Fangs[34] ist ein Screenreader-Emulator von Peter Krantz, also eine Erweiterung, die eine Webseite so aufbereitet, wie ein Screenreader sie vorlesen würde. Nach dessen Installation steht ein neuer Eintrag VIEW FANGS im Kontextmenü zur Verfügung, über den ein neues Fenster mit der aufbereiteten Ansicht geöffnet werden kann. Angezeigt wird unter anderem, wie viele Überschriften und Links auf einer Seite vorhanden sind und wie Links, Listen, Tabellen etc. angekündigt würden. Abbildung 3.18 zeigt die Darstellung der Google-Homepage im Fangs-Fenster.

33 *http://users.skynet.be/mgueury/mozilla/*
34 *http://www.standards-schmandards.com/index.php?show/fangs*

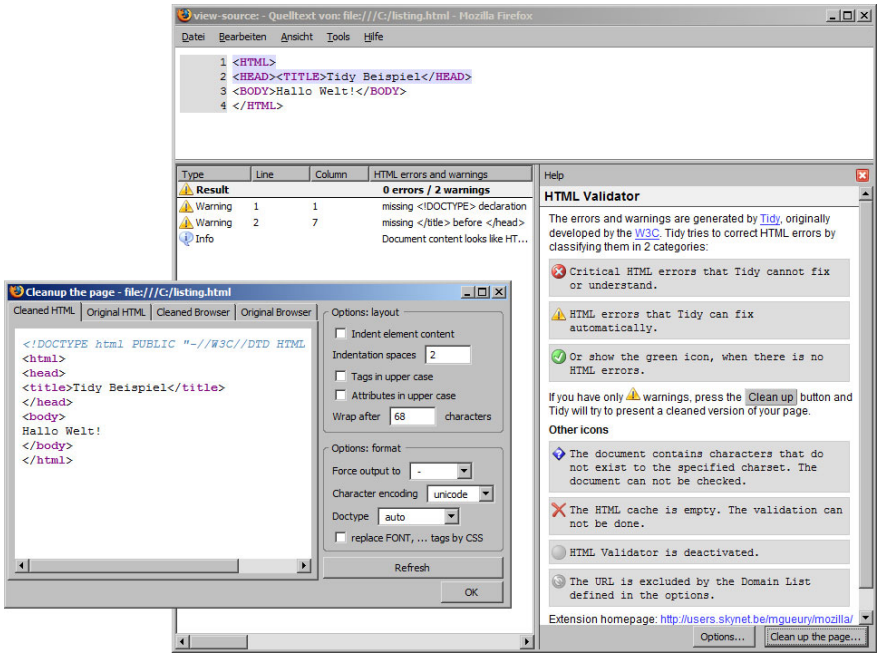


Abbildung 3.17: Der Html Validator zeigt Fehler und Warnungen an

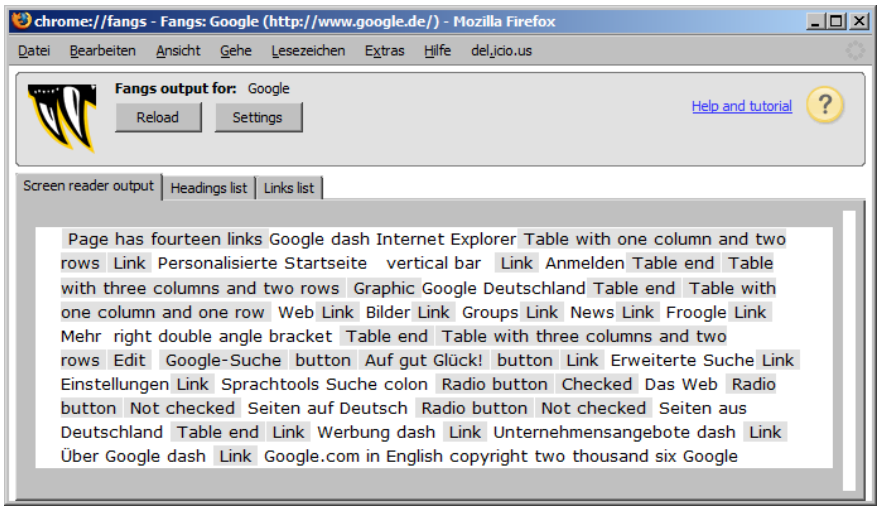


Abbildung 3.18: Der Screenreader-Emulator Fangs

3.5.5 MeasureIt

Die Erweiterung *MeasureIt*[35] von Kevin A. Freitas ermöglicht es, Breite und Höhe eines Bereichs einer Webseite direkt auf dem Bildschirm nachzumessen. Die maßnehmende Box lässt sich pixelweise auf dem Bildschirm verschieben.

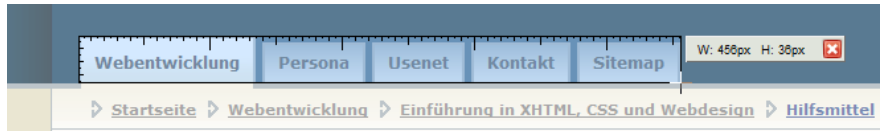


Abbildung 3.19: MeasureIt

3.5.6 LinkChecker

Wenn es darum geht, alle Links einer Website zu überprüfen, sollten Webautoren auf Werkzeuge wie *Xenu's Link Sleuth* (siehe Kapitel 3.4.3) zurückgreifen. Um sicherzustellen, dass alle Links auf einer einzigen Webseite gültig sind, reicht die Erweiterung *LinkChecker*[36], ebenfalls von Kevin A. Freitas, vollkommen aus. Über den Aufruf eines Eintrags im Kontextmenü werden alle Links auf der Seite durchlaufen und farbig markiert:

- Grün für gültige Links,
- Rot für ungültige,
- Gelb für Weiterleitungen oder geschützte Bereiche sowie
- Grau für Links, die nicht geprüft werden konnten und übersprungen wurden.

3.5.7 Right-Click Lynx viewer

Die Erweiterung *Right-Click Lynx viewer*[37], bereitgestellt durch die Yellowpipe Internet Services, fügt einen Kontextmenüeintrag hinzu, über den ein neues Fenster geöffnet werden kann, in dem die aktuelle Webseite im Lynx viewer der Yellowpipe Internet Services angezeigt wird. Webautoren können so überprüfen, wie eine Website im Textbrowser Lynx aussieht (wobei es in den meisten Fällen besser ist, Lynx zu installieren und sich direkt anzuschauen, wie die Seite darin aussieht und ob sie sich gut bedienen lässt).

35 *http://www.kevinfreitas.net/extensions/measureit/*
36 *http://www.kevinfreitas.net/extensions/linkchecker/*
37 *http://www.yellowpipe.com/yis/tools/lynx/rightlynx/*

3.5.8 WAVE-Toolbar

Mit WAVE (Web Accessibility Evaluation Tool) stellt die WebAIM (Web Accessibility in Mind) Webentwicklern unter *http://wave.webaim.org* ein Onlinewerkzeug zur Verfügung, mit dem sie die Zugänglichkeit einer Webseite überprüfen können. Fehler oder mögliche Probleme zeigt WAVE direkt auf der zu überprüfenden Seite in Form grafischer Symbole an. Unter der Adresse *http://wave.webaim.org/toolbar* können Sie sich die WAVE-Toolbar installieren, die den Aufruf des Prüfwerkzeugs erleichtert und darüber hinaus einige weitere Funktionen bietet.

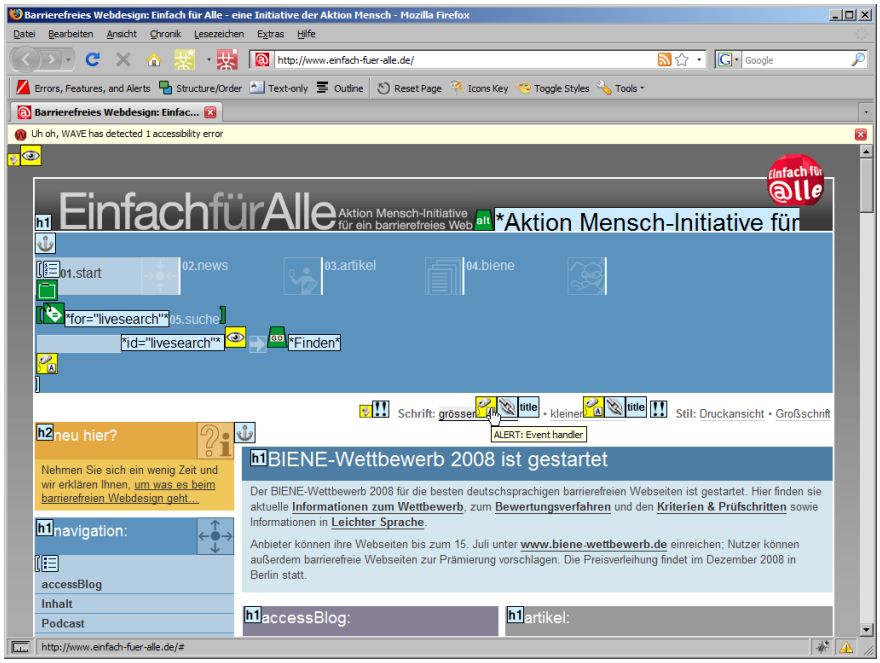


Abbildung 3.20: Die WAVE-Toolbar zeigt mögliche Accessibility-Probleme auf Webseiten an

4 Einführung in XHTML

Nachdem Sie sich bisher – ich hoffe erfolgreich – durch die theoretischen Grundlagen gearbeitet und die wichtigsten Hilfsmittel kennengelernt haben, beginnt mit diesem Kapitel endlich der praktische Teil dieses Buchs.

Dieses Kapitel erläutert die Syntax und das Vokabular von HTML und XHTML sowie den Aufbau von XHTML-Dokumenten. Das Document Object Model wird ebenso vorgestellt wie die HTML-Kompatibilitätsrichtlinien, die es Ihnen ermöglichen, XHTML-Dokumente so zu verfassen, dass heutige HTML-Benutzerprogramme und -browser sie ohne Probleme anzeigen können.

4.1 Syntax und Vokabular

4.1.1 Elementtypen, Elemente, Tags, Elementinhalte

DEFINITION

*XHTML besteht aus einer Reihe von **Elementtypen**, die zum Beispiel Absätze, Überschriften, Tabellen, Listenpunkte, Bilder oder Hyperlinks repräsentieren. Welche Elementtypen es gibt, in welchem Zusammenhang diese verwendet und wie sie verschachtelt werden dürfen, wird in der Dokumenttyp-Definition (DTD) oder im zugehörigen Schema festgelegt.*

DEFINITION

*Ein **Element** ist eine konkrete Ausprägung (eine konkrete Instanz) eines Elementtyps und wird wie folgt notiert:*

```
<Elementname>Inhaltsmodell</Elementname>
```

DEFINITION

*Der **Elementinhalt** wird von einem **Start-Tag** und einem **End-Tag** umschlossen. Ein Tag besteht aus einer öffnenden spitzen Klammer (<), dem **Elementnamen** und einer schließenden spitzen Klammer (>). Das End-Tag enthält zusätzlich das **Ende-Kennzeichen**, einen Schrägstrich (/), vor dem Elementnamen.*

Tabelle 4.1 verdeutlicht die Bezeichnungen anhand eines Elements vom Typ `h1`, einer Überschrift erster Ordnung.

Bezeichnung	Beispiel
Element	`<h1>Überschrift erster Ordnung</h1>`
Elementname	`<h1>Überschrift erster Ordnung</h1>`
Start-Tag	`<h1>Überschrift erster Ordnung</h1>`
Elementinhalt	`<h1>Überschrift erster Ordnung</h1>`
End-Tag	`<h1>Überschrift erster Ordnung</h1>`

Tabelle 4.1: Element mit Elementinhalt

DEFINITION

*Einige Elementtypen haben keinen Inhalt, zum Beispiel der Elementtyp `hr`, der eine horizontale Linie erzeugt, `br` für einen Zeilenwechsel und `img` zur Einbindung von Bildern. Diese sogenannten **leeren Elemente** werden über ein **Leeres-Element-Tag** erzeugt; das Ende-Kennzeichen wird hierbei in das Tag hineingezogen:*

```
<Elementname />
```

Tabelle 4.2 veranschaulicht die neuen Begriffe anhand des leeren Elements `br` (Zeilenumbruch).

Bezeichnung	Beispiel
Element	` `
Elementname	` `
Leeres-Element-Tag	` `

Tabelle 4.2: Leeres Element

4.1.2 Attribute

DEFINITION

*Oft hat ein Element einen Zusatz, um eine Eigenschaft zu beschreiben. Einen solchen Zusatz nennt man **Attribut**, dem ein **Attributwert** zugeordnet wird. Paare aus **Attributnamen** und Attributwert stehen immer im Start-Tag eines Elements und werden vom Elementnamen sowie voneinander durch Leerraum abgetrennt. Die Reihenfolge der Attribute innerhalb eines Elements ist beliebig. Attributwerte werden entweder von doppelten (") oder einfachen (') Anführungszeichen eingeschlossen.*

Als Beispiel: Das Element `a` (Anker) wird durch die Tags `<a>` und `</a>` ausgezeichnet. Um zu bestimmen, auf welches Ziel der Link verweisen soll, wird ihm das Attribut `href="http://www.example.com"` verliehen. `href` ist der Attributname, der URI zwischen den Anführungszeichen ist der Attributwert. Tabelle 4.3 zeigt noch einmal alle Begriffe und deren exakte Zuordnung übersichtlich unteinander.

Bezeichnung	Beispiel
Element	`<a href="http://www.example.com">Dokumentation</a>`
Elementname	`<a href="http://www.example.com">Dokumentation</a>`
Start-Tag	`<a href="http://www.example.com">Dokumentation</a>`
Elementinhalt	`<a href="http://www.example.com">Dokumentation</a>`
End-Tag	`<a href="http://www.example.com">Dokumentation</a>`
Attribut	`<a href="http://www.example.com">Dokumentation</a>`
Attributname	`<a href="http://www.example.com">Dokumentation</a>`
Attributwert	`<a href="http://www.example.com">Dokumentation</a>`

Tabelle 4.3: Das Element `a` mit dem Attribut `href`

4.1.3 Kommentare

DEFINITION

*XHTML bietet die Möglichkeit, **Kommentare** einzubinden. Kommentare sind dazu geeignet, interne Angaben zum Autor, das Erstellungsdatum, Anmerkungen und Notizen einzufügen oder um verwendete Elemente auszukommentieren. Diese stehen sichtbar im Quelltext der Seite, werden somit auch an den Browser übertragen, allerdings nicht auf der Seite angezeigt.*

Geöffnet werden XHTML-Kommentare durch den öffnenden Begrenzer der Auszeichnungsdeklaration (*markup declaration open delimiter*, `<!`) und den öffnenden Begrenzer des Kommentars (*comment open delimiter*, `--`). Geschlossen werden sie durch den schließenden Begrenzer des Kommentars (*comment close delimiter*, `--`) und den schließenden Begrenzer der Auszeichnungsdeklaration (*markup declaration close delimiter*, `>`):

```
<!-- Dies ist ein einzeiliger Kommentar. -->

<!-- Dies ist ein mehrzeiliger Kommentar, der durch
  die gleichen Zeichenfolgen eingeschlossen wird. -->
```

Da ein Kommentar mit der Zeichenfolge `--` eingeleitet und beendet wird, dürfen Sie die Zeichenkette `--` in XHTML-Dokumenten *nicht* innerhalb eines Kommentars verwenden.

4.1.4 Zeichenreferenzen

Das Zeichen < darf nicht Teil eines Elementinhalts sein, da der Browser dieses Zeichen sonst als Teil des Markups (Beginn eines Start-Tags) missinterpretieren würde. Ebenso dürfen doppelte (`"`) oder einfache (`'`) Anführungszeichen nicht in Attributwerten auftreten, die durch die gleichen Zeichen eingeschlossen werden. Deswegen können Sie nicht einfach schreiben:

```
<p>Das Zeichen < darf nicht Teil eines Elementinhalts sein.</p>
```

Stattdessen muss das Zeichen < in Form einer Zeichenreferenz notiert werden.

DEFINITION

*Eine **Zeichenreferenz** dient der Eingabe von Zeichen, die im Markup an dieser Stelle nicht erlaubt oder nicht in dem gewählten Zeichenvorrat vorhanden sind. Eine Zeichenreferenz beginnt mit einem Ampersand (`&`)[1] und endet mit einem Semikolon (`;`). Es werden numerische und benannte Zeichenreferenzen unterschieden.*

DEFINITION

*Eine **numerische Zeichenreferenz** (*numeric character reference*) gibt die Codeposition eines Zeichens im gewählten Dokumentzeichensatz an. Sie kann zwei Formen annehmen, wobei `D` und `H` als Variablen zu verstehen sind:*

- *Die Syntax `&#D;` weist auf die dezimale Zeichennummer D. Beispiel: `<`*
- *Die Syntax `&#xH;` weist auf die hexadezimale Zeichennummer H. Beispiel: `<`*

1 Aus diesem Grund muss das Zeichen & ebenfalls umschrieben werden, da der Browser sonst glauben könnte, eine Zeichenreferenz würde eingeleitet.

DEFINITION

*Eine **benannte Zeichenreferenz** (character entitiy reference) verwendet zwischen & und ; statt einer Zahl eine mnemonische Zeichenkombination, die das Zeichen beschreibt und sich mehr oder weniger leicht merken lässt. Für jedes Zeichen der Codepositionen 160 bis 255 in Unicode existiert eine benannte Zeichenreferenz, darüber hinaus auch für viele weitere (zum Beispiel mathematische Symbole oder griechische Zeichen).*

XHTML verwendet als Zeichenvorrat Unicode; das heißt, Sie können prinzipiell jedes Zeichen aus Unicode in XHTML-Dokumenten verwenden. Da es mehrere Möglichkeiten gibt, ein Zeichen im Quelltext einer Webseite zu referenzieren, können Sie sogar Zeichen aus Unicode unterbringen, obwohl Sie eine Kodierung mit eingeschränktem Zeichenvorrat wie ISO-8859-1 oder US-ASCII für den Großteil Ihres Texts verwenden.

Grundsätzlich können Sie Ihre XHTML-Dokumente in jeder beliebigen Zeichenkodierung schreiben und abspeichern, solange sie mit der korrekten Information zur Kodierung vom Webserver ausgeliefert werden.

Tabelle 4.4 stellt numerische und benannte Zeichenreferenzen von im XHTML-Markup verwendeten Zeichen gegenüber.

Zeichen	**Bezeichnung**	**Zeichenreferenz**		
		dezimal	**hexadezimal**	**benannt**
<	Kleiner als	<	<	<
>	Größer als	>	>	>
"	Anführungszeichen	"	"	"
'	Apostroph	'	'	'
&	Ampersand	&	&	&

Tabelle 4.4: Zeichenreferenzen für Markup-Zeichen

Entgegen einem weit verbreiteten Irrglauben ist es übrigens in den meisten Fällen nicht nötig, das Anführungszeichen, den Apostroph und sogar das Größer-als-Zeichen per Zeichenreferenz zu notieren. Dies ist nur dann notwendig, wenn diese Zeichen in Attributwerten auftreten, wobei Anführungszeichen und Apostroph auch nicht umschrieben werden müssen, wenn der Attributwert im jeweils anderen Zeichen eingeschlossen wird.

> > > HINWEIS

***XHTML ist keine Programmiersprache**, sondern eine Auszeichnungssprache. Es gibt keine bedingten Anweisungen, keine Variablen, Aufgabenlösungen durch Befehlsabfolgen oder Algorithmen. Dementsprechend gibt es auch keine XHTML-Befehle, sondern Elementtypen, Elemente, Tags, Attribute ... XHTML wird nicht programmiert oder »gecodet« (oder was man sonst noch hört und liest), sondern geschrieben. Jemand, der in XHTML Webseiten verfasst, ist daher ein Autor, kein Programmierer.*

*Auch **die Bezeichnung »Seitenbeschreibungssprache« ist nicht zutreffend**, da eben keine Seite, also kein festgelegtes Format zur Verfügung steht, auf dem Objekte nach genauen Vorgaben arrangiert werden. Und wenn es keine Seite gibt, kann auch keine Seite beschrieben werden. XSL-FO (XSL Formatting Objects) oder PDF (Portable Document Format) sind Seitenbeschreibungssprachen.*

4.2 Die globale Struktur eines XHTML-Dokuments

Ein XHTML-Dokument lässt sich grob in vier Abschnitte aufteilen:

Abschnitte eines XHTML-Dokuments

1. Die XML-Deklaration; eine Zeile, die unter anderem die XML-Version angibt.
2. Die Dokumenttyp-Deklaration; eine Zeile, die den XHTML-Dokumenttyp angibt.
3. Einen Dokumentkopf `head`, der den Dokumenttitel (das `title`-Element) und zusätzliche (Meta-)Informationen enthält.
4. Einen Dokumentrumpf `body`, der unter anderem den Inhalt des Dokuments enthält, der im Browser dargestellt wird.

Vor und nach jedem Abschnitt kann Leerraum vorkommen.

• • • DEFINITION

***Leerraum** (white space) ist eine in der Datenverarbeitung und Programmierung gebräuchliche allgemeine Bezeichnung für Zeichen in einem Text, die im Editor nicht sichtbar sind und dennoch Platz in Anspruch nehmen. Je nach Kontext werden verschiedene Zeichen als Leerraumzeichen angesehen, in HTML und XHTML die Zeichen Leerzeichen, Tabulator, Wagenrücklauf und Zeilenvorschub.*

4.2.1 Die XML-Deklaration

DEFINITION

*Die **XML-Deklaration** ist eine Erkennungszeichenfolge im Prolog eines XML-Dokuments. Sie wird durch die Zeichenfolge `<?xml` eingeleitet und durch `?>` abgeschlossen. Dazwischen werden die Version sowie – jeweils optional – die Zeichenkodierung und die Standalone-Dokumentdeklaration notiert:*

```
<?xml Version Zeichenkodierung Standalone-Dokumentdeklaration ?>
```

Eine XML-Deklaration kann beispielsweise wie folgt aussehen:

```
<?xml version="1.0" encoding="utf-8" ?>
```

Gültige Werte der Standalone-Dokumentdeklaration sind `yes` und `no`. `yes` bedeutet, dass das Dokument über *keine* Dokumenttyp-Definition (DTD) verfügt, `no` bedeutet, *dass* es über eine solche verfügt. Wird das Attribut, wie in diesem Beispiel, weggelassen, wird der Standardwert `no` angenommen.

Die XML-Deklaration ist in XHTML optional. Sie können sie weglassen, wenn Sie die Zeichenkodierung anderweitig, am besten serverseitig, angeben. Dazu reicht beim Apache Webserver bereits die Zeile

```
AddType text/html;charset=utf-8 html
```

in der Zugriffskontrolldatei (*.htaccess*), sofern Sie den Apache Webserver verwenden, XHTML-Dokumente als `text/html` ausliefern und die Endung `html` verwenden.

4.2.2 Die Dokumenttyp-Deklaration

DEFINITION

*Die **Dokumenttyp-Deklaration** benennt die Dokumenttyp-Definition (DTD), die für ein Dokument verwendet wird. Sie teilt der verarbeitenden Software (zum Beispiel einem Browser) mit, um welchen Dokumenttyp, das heißt um welche XHTML-Version, es sich bei einem Dokument handelt.*

Eine Dokumenttyp-Deklaration für XHTML-Dokumente ist folgendermaßen aufgebaut:

```
<!DOCTYPE html PUBLIC "FPI" "SI">
```

Der öffentliche Bezeichner (*Formal Public Identifier*, FPI) gibt an, welcher DTD das Dokument entspricht. Der **Systembezeichner** (*System Identifier*, SI) enthält den URI der DTD.

Für XHTML 1.0 existieren insgesamt drei Dokumenttyp-Deklarationen. Sie unterscheiden sich in den Elementen, die sie unterstützen.

! ! ! ACHTUNG

Ob Sie eine Dokumenttyp-Deklaration notieren und welche Sie verwenden, hat in einigen Browsern Einfluss auf die Darstellung Ihrer Dokumente. Lesen Sie dazu das Kapitel 6.8.2 »Doctype Switching«.

XHTML 1.0 Transitional

Hierbei handelt es sich sozusagen um die »tolerante« XHTML-Version. Alle Elemente und Attribute aus HTML 4.01 Transitional wurden beibehalten. Transitional wird vor allem dann geschrieben, wenn auf die Darstellung veralteter Browser noch Wert gelegt wird und daher auch Layoutelemente und -attribute verwendet werden sollen.

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

XHTML 1.0 Strict

Dies ist die »strenge« Version von XHTML. Hier wurden viele missbilligte (das heißt durch neuere Konstrukte überholte) Elemente und Attribute, die nur der Darstellung, nicht mehr nur der Auszeichnung des Texts dienlich waren, gestrichen. Ich empfehle Anfängern, die Inhalte und Informationen zu deren Darstellung konsequent voneinander trennen wollen, diese XHTML-Variante zu wählen, zumindest so lange, bis es einen guten Grund gibt, davon abzuweichen.

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

XHTML 1.0 Frameset

Diese Dokumenttyp-Deklaration wird für Framesets verwendet und hier nur der Vollständigkeit halber aufgeführt. Auf Frames werde ich in dieser Einführung nur am Rande eingehen. Viele Usability-Experten raten ausdrücklich vom Einsatz von Frames ab, daher spielen sie im modernen Web nahezu keine Rolle mehr. Professionelle Websites setzen nur noch selten Frames ein, und auch im privaten Umfeld geht die Anzahl an Framesites immer weiter zurück.

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

4.2.3 Wurzelelement: Der Elementtyp html

DEFINITION

*Das **Element html** ist das Wurzelelement* (root element) *eines XHTML-Dokuments, also das Element, das alle anderen Elemente enthält.*

Durch die Angabe des Attributs `xmlns="http://www.w3.org/1999/xhtml"` werden standardmäßig alle Elemente eines XHTML-Dokuments dem XHTML-**Namensraum** zugeordnet. Die Attribute `xml:lang` und `lang` definieren die Sprache des Dokuments (mehr dazu in Kapitel 4.6 »Universalattribute«).

Namensräume

XML erlaubt die Definition eines neuen Dokumenttyps, indem man zwei oder mehr Dokumenttypen miteinander kombiniert. XHTML ist ein solcher Dokumenttyp. In diesen »gemischten« Dokumenttypen können Elemente oder Attribute doppelt auftreten. Wenn man sie kombiniert, muss man entsprechend vorsichtig sein: Ist ein Element `p` im aktuellen Kontext ein »Absatz« aus dem einen oder eine »Person« aus dem anderen Format? Um Namenskollisionen auszuschließen, werden Elemente und Attribute durch sogenannte **Namensräume** qualifiziert, die wiederum – aus Gründen der Eindeutigkeit – durch URIs identifiziert werden. Dabei spielt es keine Rolle, ob dieser URI tatsächlich abrufbar ist.

Das Element `html` enthält die Elemente `head` und `body`.

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

<html xmlns="http://www.w3.org/1999/xhtml"
  xml:lang="de" lang="de">
  <!-- head und body -->
</html>
```

Listing 4.1: Der Elementtyp `html`

4.2.4 Dokumentkopf: Der Elementtyp head

DEFINITION

*Das **head-Element** enthält Informationen über das aktuelle Dokument, beispielsweise seinen Titel, Schlüsselwörter, die sinnvoll für Suchmaschinen sein können, und andere Daten, die nicht als Dokumentinhalt betrachtet werden.*

Browser stellen Elemente, die im `head` stehen, im Allgemeinen nicht als Inhalt dar. Sie können jedoch Informationen aus dem Dokumentkopf über andere Mechanismen für Nutzer verfügbar machen.

Das einzige Element, das das `head`-Element jedes XHTML-Dokuments enthalten *muss*, ist das Element `title`. Auf weitere Elemente werden Sie im weiteren Verlauf dieser Einführung stoßen.

Dokumenttitel: Der Elementtyp title

DEFINITION

*Das **title-Element** dient dazu, den Inhalt eines Dokuments zu beschreiben. Jedes XHTML-Dokument muss ein `title`-Element im Dokumentkopf besitzen. Es darf Zeichenreferenzen enthalten, aber keine andere Auszeichnung, auch keine Kommentare.*

Viele Experten bezeichnen das `title`-Element als das wichtigste Element einer Website[2]. Es wird von Browsern in der Titelzeile des Anzeigefensters und bei Setzen von Lesezeichen (Favoriten) angezeigt. Suchmaschinen dient es als Hilfsmittel zur Indizierung, Bewertung und Anzeige einer Webseite. Daher sollte der Dokumenttitel möglichst aussagekräftig, aber nicht zu lang sein; beschränken Sie sich auf 60 bis 70 Zeichen.

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

<html xmlns="http://www.w3.org/1999/xhtml"
  xml:lang="de" lang="de">
  <head>
    <title>Mein erstes XHTML-Dokument</title>
  </head>

  <!-- body -->
</html>
```

Listing 4.2: Das Element `head` enthält ein `title`-Element

2 siehe unter anderem ***[Connolly 2006]***

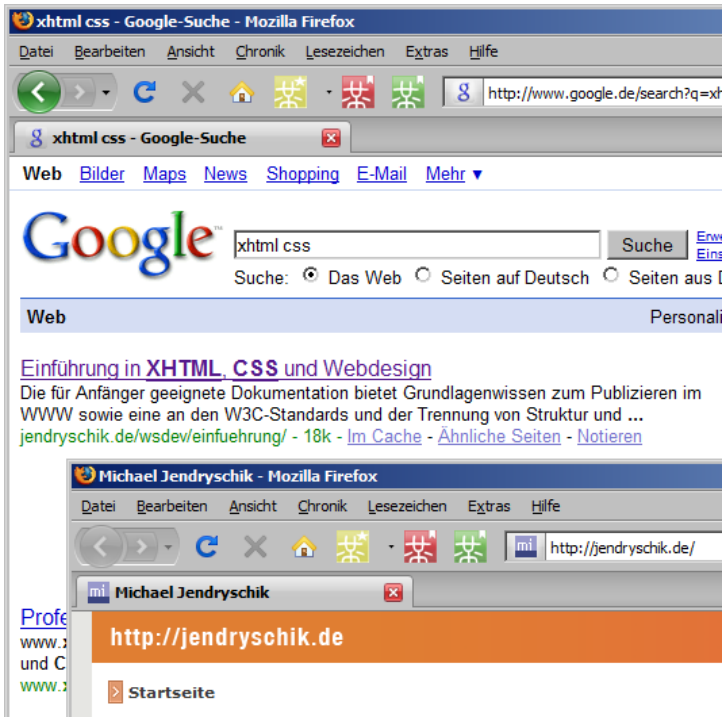


Abbildung 4.1: Der Dokumenttitel wird unter anderem in der Titelzeile des Browsers und auf der Trefferseite von Suchmaschinen angezeigt

4.2.5 Dokumentrumpf: Der Elementtyp body

DEFINITION

*Der **Dokumentrumpf body** enthält den Inhalt des Dokuments. Ein Benutzerprogramm kann den Inhalt auf unterschiedlichste Art präsentieren. Für visuelle Browser zum Beispiel kann man sich den Rumpf als Leinwand vorstellen, auf der der Inhalt erscheint. Für akustische Benutzerprogramme kann derselbe Inhalt vorgelesen werden.*

Block- und Inline-Elemente

DEFINITION

Bestimmte Elemente, die im Dokumentrumpf `body` *eines XHTML-Dokuments vorkommen, werden **Blockelemente** genannt, während andere Elemente **Inline-Elemente** sind. Diese Unterscheidung basiert vor allem auf drei Gedanken:*

1. *Im Allgemeinen gilt: Blockelemente können sowohl Inline-Elemente als auch andere Blockelemente enthalten; Inline-Elemente hingegen können nur Text und andere Inline-Elemente enthalten. In dieser strukturellen Entscheidung ist die Idee enthalten, dass Blockelemente größere Strukturen schaffen als Inline-Elemente. Natürlich gibt es auch hierbei Ausnahmen: Elemente vom Typ `p` dürfen keine anderen Blockelemente enthalten, ebenso wenig wie die Überschriftenelemente `h1` bis `h6`. Ich habe die Erfahrung gemacht, dass Webautoren intuitiv lernen, wie Elemente ineinander verschachtelt werden dürfen. Fragen Sie zu Beginn Ihrer Arbeit regelmäßig einen Validator, der Sie auf Fehler aufmerksam machen wird. Bereits nach kurzer Zeit werden Sie ihn nicht mehr als Ratgeber einsetzen, sondern nur noch zur Qualitätssicherung.*
2. *Blockelemente beginnen in einer neuen Zeile, während Inline-Elemente inzeilig dargestellt werden. Oftmals wird zwischen Blockelementen standardmäßig ein Abstand dargestellt, zum Beispiel bei Absätzen oder Überschriften, um diese voneinander abzugrenzen.*
3. *Die Breite von Blockelementen umfasst die gesamte Zeile, während Inline-Elemente nur so breit sind, wie ihr Inhalt es erfordert.*

Man spricht dabei von **Blockboxen** oder **Inlineboxen**, die von Block- beziehungsweise Inline-Elementen generiert werden. Listen- und Tabellenelemente erzeugen spezielle Arten von Boxen. Später im CSS-Kapitel werden Sie lernen, die Darstellung beliebiger Elemente zu spezifizieren, und dabei unter anderem, ob ein Element als Block oder inzeilig angezeigt werden soll.

4.2.6 Hallo-Welt-Beispiel

Alle Beispiele in diesem Buch basieren auf demselben Grundgerüst. Wir setzen XHTML 1.0 Strict ein, um uns schon direkt zu Beginn eine maximale Trennung von Struktur und Darstellung anzugewöhnen. Das XHTML-Grundgerüst sieht wie folgt aus:

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

<html xmlns="http://www.w3.org/1999/xhtml"
  xml:lang="de" lang="de">
  <head>
    <title>XHTML-Grundgerüst</title>
  </head>

  <body>
    <p>Hallo Welt!</p>
  </body>
</html>
```

Listing 4.3: XHTML-Grundgerüst

Öffnen Sie ein leeres Editorfenster und kopieren Sie das obige Quelltextbeispiel hinein. Speichern Sie die Datei als `index.html` ab. Wenn Sie diese Datei in einem Browser öffnen, sehen Sie eine Darstellung wie in Abbildung 4.2. Zu lesen ist der Text »Hallo Welt!« in einem Fenster mit dem Titel »XHTML-Grundgerüst« in der Titelzeile.

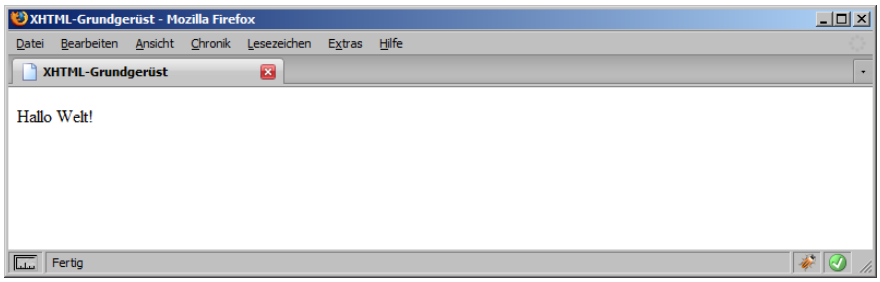


Abbildung 4.2: XHTML-Grundgerüst

Formatierungskonventionen

Vielleicht ist Ihnen bereits jetzt aufgefallen, dass es für die Darstellung Ihres Quelltexts im Browser (bis auf wenige Ausnahmen) irrelevant ist, an welchen Stellen Sie Leerzeilen oder Leerzeichen einfügen. Beliebig viel Leerraum hintereinander wird als ein einziges Leerzeichen interpretiert. Probieren Sie es aus! Das Layout Ihres Quelltexts hat im Allgemeinen keinen Einfluss auf die Darstellung des Dokuments. Dies hat den Vorteil, dass Sie Ihren Quelltext so, wie Sie es wünschen, strukturieren können, ohne die Darstellung im Browser zu beeinflussen.

Sie sollten sich angewöhnen, Ihren Quelltext stets sauber und vor allem einheitlich zu formatieren. Folgende Regeln können Ihnen dabei helfen:

- Schreiben Sie Blockelemente stets in eine neue Zeile. Schließen Sie sie in derselben Zeile wieder, wenn sie keine weiteren Blockelemente beinhalten. Schreiben Sie nicht `<div><p>Absatz</p></div>`, sondern:

  ```
  <div>
    <p>Absatz</p>
  </div>
  ```

- Rücken Sie verschachtelte Blockelemente ein. Einrücken können Sie im Allgemeinen mittels der Tabulatortaste, entweder einzelne Zeilen oder ganze Blöcke, wenn Sie alle Zeilen zusammen markieren. Entfernen können Sie Einrückungen zumeist mittels ⇧+⇤. Editoren rücken unterschiedlich ein, das heißt, sie fügen verschiedene Leerraumzeichen ein. Es ist wichtig, dass Sie alle Editoren, die Sie verwenden, so konfigurieren, dass sie sich ein-

heitlich verhalten. Ich empfehle eine Einrückung mittels zweier Leerzeichen. Abbildung 4.3 zeigt, wie Sie diese Einstellung in UltraEdit vornehmen.

- Trennen Sie Bereiche mit einer Leerzeile voneinander ab. Im Grundgerüst sind die Dokumenttyp-Deklaration vom `html`-Element und das `head`- vom `body`-Element getrennt. Verwenden Sie Leerzeilen bewusst, aber sparsam. Mehrere Leerzeilen hintereinander sollten Sie vermeiden, ebenso zu viele Leerzeilen, da der Quelltext sonst optisch auseinanderfällt und die Lesbarkeit darunter leidet. Denken Sie in Strukturbereichen! Nehmen Sie sich die Listings in diesem Buch als Beispiel. Irgendwann werden Sie selbst ein Gefühl dafür entwickeln.
- Text sollte stets direkt von Tags umschlossen sein. Schreiben Sie nicht `<strong> stark betont </strong>`, sondern `<strong>stark betont</strong>`.
- Verwenden Sie doppelte Anführungszeichen. Schreiben Sie nicht `<a href='http://example.org/'>Link</a>`, sondern `<a href="http://example.org/">Link</a>`.

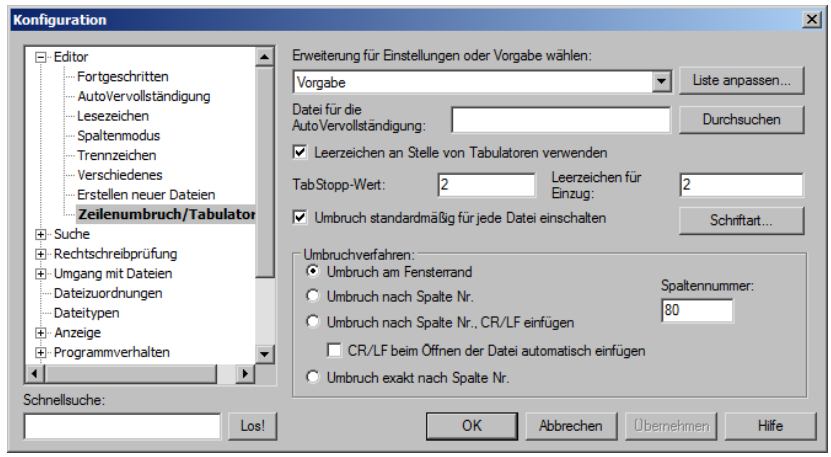


Abbildung 4.3: Konfiguration der Einrückung in UltraEdit

4.3 Das Document Object Model

DEFINITION

*Das **Document Object Model** (DOM) ist eine W3C-Empfehlung, die Programmen eine allgemeine Vorgehensweise für den Zugriff auf Bestandteile eines Dokuments liefert. Das DOM beschreibt Objekte, Eigenschaften und Methoden und ist sowohl im XML- als auch im JavaScript-Umfeld von großer Bedeutung.*

Dabei wird die Struktur des Dokuments, beispielsweise eines XHTML-Dokuments, so dargestellt wie das Verhältnis verwandter Familien zueinander in einem Familienstammbaum. Es gibt Vorfahren, Nachfahren, Eltern, Kinder und Geschwister. Daher spricht man in diesem Zusammenhang auch häufig von einem »Dokumentstammbaum«.

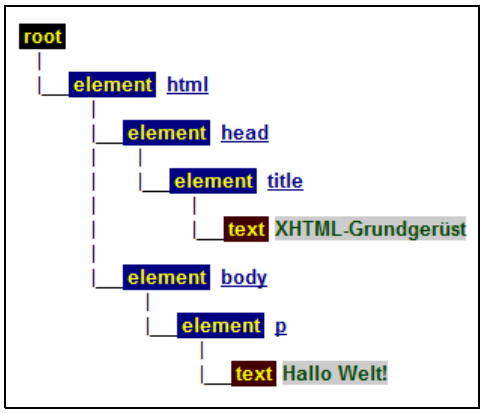


Abbildung 4.4: Der Dokumentstammbaum des XHTML-Grundgerüsts

Abbildung 4.4 zeigt den Dokumentstammbaum des XHTML-Dokuments aus Listing 4.3. Die Bedeutung der aus dem alltäglichen Sprachgebrauch bekannten Begriffe lässt sich sehr einfach auf die Abbildung übertragen, zur besseren Verständlichkeit erläutere ich sie jedoch kurz:

Beziehungen im Dokumentstammbaum

- **Kindelement**: Ein Element *A* ist Kind eines Elements *B*, wenn *A* direkt von *B* abstammt. Hier ist `head` Kind von `html`, `title` wiederum Kind von `head`.
- **Elternelement**: Ein Element *A* ist das Elternelement eines Elements *B*, wenn *B* Kind von *A* ist. Folglich ist `html` Elternelement von `head` und `head` wiederum Elternelement von `title`.
- **Geschwisterelement**: Ein Element *A* ist Geschwisterelement eines Elements *B*, wenn beide Elemente vom selben Elternelement abstammen. Die Elemente `head` und `body` sind Geschwisterelemente. Darüber hinaus unterscheidet man zwischen vorhergehenden und nachfolgenden Geschwisterelementen: `head` ist vorhergehendes Geschwisterelement von `body`, `body` wiederum ist nachfolgendes Geschwisterelement von `head`.
- **Vorfahrenelement**: Ein Element *A* ist Vorfahrenelement eines Elements *B*, wenn sich die Abstammungslinie von *A* nach unten direkt zu *B* verfolgen lässt. Für das Element `p` sind sowohl `body` als auch `html` Vorfahrenelemente.

- **Nachkommenelement**: Ein Element *A* ist Nachkommenelement eines Elements *B*, wenn *B* in direkter Linie von *A* abstammt. Für das Element `html` sind die Elemente `head` und `title` sowie `body` und `p` Nachkommenelemente.
- **Stammelement** oder **Wurzelelement**: Ein Element *A* ist Stammelement eines Dokuments, wenn es nur Nachkommen-, aber keine Vorfahrenelemente besitzt. Das Stammelement jedes XHTML-Dokuments ist das Element `html`.

Knoten

Die Bestandteile eines Dokumentstammbaums werden **Knoten** genannt. Alle bisher vorgestellten Knoten sind Elementknoten, das heißt, der Knoten entspricht einem bestimmten Element beziehungsweise einem Element und dessen Nachkommen. Es gibt jedoch noch weitere Knoten: Attributknoten, Textknoten und Namensraumknoten. Abbildung 4.5 ergänzt Abbildung 4.4 entsprechend. Die Textknoten, die lediglich ein `\n` enthalten, sind Zeilenumbrüche. Zwei Zeilenumbrüche (`\n \n`) ergeben eine Leerzeile.

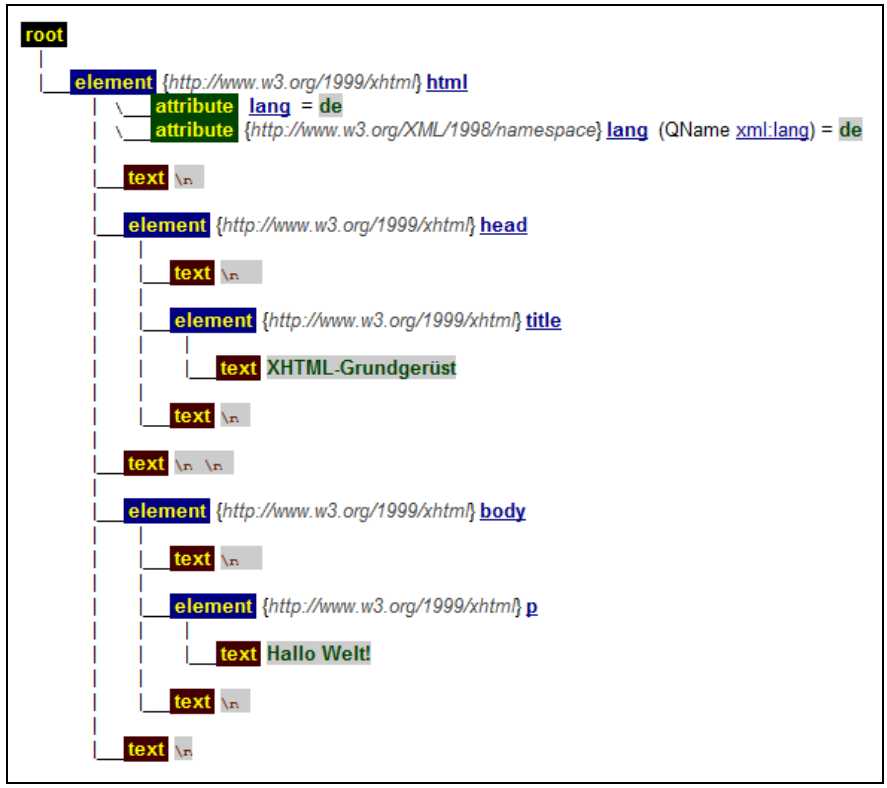


Abbildung 4.5: Das DOM des XHTML-Grundgerüsts, ergänzt um Attribut-, Text- und Namensraumnoten

4.4 Kompatibilitätsrichtlinien

Wie bereits an mehreren Stellen angesprochen, unter anderem in Kapitel 2.6.3, in dem es um die Frage geht, weshalb Webautoren XHTML 1.0 einsetzen sollten und nicht HTML 4.01, unterscheiden sich HTML und XHTML nur in wenigen Punkten. Nahezu alle Unterschiede zwischen HTML und XHTML resultieren aus der Tatsache, dass HTML ein SGML-Dokumenttyp ist, während XHTML der strengeren XML-Syntax folgt. Es ist nicht das Ziel dieser Einführung, Sie im Detail mit HTML vertraut zu machen. Jedoch sollten Sie einen Blick auf die in der XHTML-Empfehlung aufgeführten HTML-Kompatibilitätsrichtlinien werfen, die es Ihnen ermöglichen, XHTML-Dokumente so zu verfassen, dass heutige HTML-Benutzerprogramme und -browser sie ohne Probleme anzeigen können. Dieser Abschnitt stellt die wichtigsten Regeln vor.

Die wichtigsten Regeln

- **Fügen Sie ein Leerzeichen vor dem abschließenden /> bei Leeres-Element-Tags ein.**

  Schreiben Sie `<br />` und nicht `<br/>`. Auch auf die zugelassene, aber unübliche Form `<br></br>` sollten Sie verzichten. Sie könnten sonst Schwierigkeiten bei der Darstellung in älteren Browsern bekommen.

- **Fassen Sie Start- und End-Tag von Elementen ohne Inhalt nicht zusammen.**

  Elemente, die eigentlich mit Start- und End-Tag notiert werden, dürfen – wenn sie keinen Inhalt haben – in XML mit einem Leeres-Element-Tag notiert werden. Dies bezeichnet man als Elementminimierung. Davon sollten Sie jedoch Abstand nehmen. Falls es einen guten Grund gibt, ein Element ohne Inhalt zu notieren, schreiben Sie zum Beispiel `<div></div>`, aber niemals `<div />`.

- **Notieren Sie den Inhalt von script- und style-Elementen innerhalb von CDATA-Abschnitten.**

  In XHTML-Dokumenten werden die Inhalte des `script`- und des `style`-Elements (siehe die Kapitel 5.11.3 und 6.3.2) anders behandelt als in HTML. Dort ist es so, dass der Inhalt des `script`-Elements nicht geparst wird, das heißt, (X)HTML-Markup-Zeichen wie `<` oder `&` (siehe Kapitel 4.1.4 »Zeichenreferenzen«) innerhalb eines Scripts sind unkritisch. In XHTML ist das anders. Wenn Sie derartige Zeichen verwenden, werden diese als XHTML interpretiert – in den meisten Fällen wird Ihr Dokument dadurch ungültig. Um dies zu verhindern, müssen Sie diese Zeichen als Zeichenreferenz notieren, dann funktioniert allerdings häufig das Script nicht mehr. Um das

Problem zu lösen, schließen Sie den Inhalt des `script`-Elements in einen sogenannten CDATA-Abschnitt ein. Dadurch teilen Sie dem Parser mit, dass kein Markup folgt, sondern normaler Text.

CDATA-Abschnitte beginnen mit `<![CDATA[` und enden mit `]]>`. Leider gibt es Browser, die diese Zeichenketten dann als JavaScript-Code oder CSS zu interpretieren versuchen. Daher ist es gängige Praxis, sie durch entsprechende Kommentare zu umschließen; sowohl bei JavaScript als auch bei CSS werden diese mit `/*` eingeleitet und mit `*/` beendet.

```
<style type="text/css">
  /* <![CDATA[ */
  ... ungeschützter (unescaped) Inhalt ...
  /* ]]> */
</style>
```

CDATA-Abschnitte werden vom XML-Prozessor erkannt und erscheinen als Knoten im Dokumentstammbaum.

- **Verwenden Sie sowohl das Attribut lang als auch das Attribut xml:lang zur Festlegung der Sprache eines Elements oder Attributs.**
- **Liefern Sie XHTML-Dokumente mit dem MIME-Typ text/html aus.**

Das W3C sieht für XHTML-Dokumente den MIME-Typ `application/xhtml+xml` vor. Leider verstehen nicht alle Browser diesen MIME-Typ, allen voran der Internet Explorer, der auf diese Art ausgelieferte Ressourcen zum Download anbietet, anstatt sie im Browserfenster anzuzeigen. Die einfachste Lösung besteht darin, XHTML-Dokumente wie HTML-Dokumente als `text/html` auszuliefern. Dies ist für XHTML 1.0-Dokumente, die den Kompatibilitätsrichtlinien genügen, zulässig und unproblematisch. Leider entgehen Ihnen damit die Vorteile, die XHTML mit sich bringt; das Dokument wird nicht durch einen XML-Parser behandelt, sondern durch den gewöhnlichen HTML- beziehungsweise SGML-Parser gejagt.

Es gibt unterschiedliche Methoden, die es ermöglichen, XHTML-Dokumente je nach Fähigkeiten des Browsers als `text/html` oder `application/xhtml+xml` auszuliefern. Versteht ein Browser den XHTML-MIME-Typ, wird das Dokument mit diesem ausgeliefert, ansonsten mit `text/html`. Verschiedene Methoden schildert Christoph Schneegans in seinem Artikel »XHTML-Einmaleins« unter der Adresse *http://schneegans.de/web/xhtml/*.

4.5 Metaangaben: Der Elementtyp meta

DEFINITION

meta-Elemente *enthalten Informationen über das Dokument für Webserver, Webbrowser oder* ***Suchroboter****, zum Beispiel eine Liste von Schlüsselwörtern, Informationen über den Autor, das Verfallsdatum oder die verwendete Zeichenkodierung. Metaangaben werden im Dokumentkopf notiert und nicht vom Benutzerprogramm angezeigt.*

Die HTML- und XHTML-Empfehlungen schreiben keine Metaangaben vor, sondern treffen nur Aussagen darüber, wie Metaangaben notiert werden müssen. `meta` ist ein leeres Element mit in der Regel zwei Attributen: Das erste Attribut lautet entweder `name` oder `http-equiv`, das zweite stets `content`. Mit `name` werden Angaben benannt, die nicht näher spezifiziert sind und sich daher tendenziell eher an auslesende Clientprogramme und Suchroboter richten; `http-equiv` benennt Angaben für Webserver beziehungsweise Webbrowser. Der Webserver sollte Metaangaben vor der Auslieferung des Dokuments auslesen und sie in den HTTP-Header des Dokuments einbauen.

Unter anderem folgende Metaangaben sind auf vielen Webseiten zu finden:

- **Angabe des MIME-Typs und der Zeichenkodierung des Dokuments:**

  MIME-Typ und Zeichenkodierung

  ```
  <meta http-equiv="content-type"
    content="text/html; charset=utf-8" />
  ```

  Ursprünglich war es so gedacht, dass ein Webserver diese Angaben ausliest und das Dokument, mit entsprechenden HTTP-Headern ausgestattet, an den Browser schickt. Ein Webserver, der sich so verhält, ist mir jedoch nicht bekannt. Allerdings kann dieses `meta`-Element sinnvoll sein, wenn das Dokument lokal von einer Festplatte geladen wird, wo kein Webserver verfügbar ist, etwa wenn ein Besucher die Seite auf seiner Festplatte abspeichert. Browser sind verpflichtet, Angaben zur Zeichenkodierung festzuhalten; einige Browser tun das auch, viele andere jedoch nicht.

- **Angabe des MIME-Typs der verwendeten Stylesheetsprache:**

  Stylesheetsprache

  ```
  <meta http-equiv="content-style-type" content="text/css" />
  ```

  Auch hier ist es sinnvoll, die Sprache primär über den HTTP-Header festzulegen.

- **Angabe für Suchroboter:**

  Suchroboter

  ```
  <meta name="robots" content="index, follow" />
  ```

  Standardmäßig indiziert ein Suchroboter Ihre Webseite, wenn er zu ihr gelangt ist, und folgt allen auf der Seite befindlichen Links, um diese Seiten

dann ebenfalls zu indizieren. Mit der Angabe `content="index, follow"` können Sie es ihm noch einmal explizit erlauben, sollten Sie global für Ihre Website andere Angaben gemacht haben. Weitere mögliche Angaben sind `noindex` und `nofollow`, um dem Suchroboter zu verbieten, die Seite zu indizieren beziehungsweise den Links auf der Seite zu folgen. Dieses `meta`-Element wird bisher nur von wenigen Suchrobotern ausgewertet und funktioniert daher nicht sehr zuverlässig. Es sollte ergänzend eingesetzt werden. Eine sicherere Methode zur Steuerung von Suchrobotern ist die Datei *robots.txt*.

- **Angaben für Suchmaschinen:**

Beschreibung

```
<meta name="description" content="Beschreibung der Webseite" />
```

Der Wert des Attributs `content` kann von Suchdiensten in Trefferlisten angezeigt werden. Sie können dort eine kurze Beschreibung Ihrer Seite anführen, die dem Nutzer der Suchmaschine knapp vermittelt, um was es auf Ihrer Seite geht. Beschränken Sie sich dabei auf zwei bis drei Sätze oder 200 bis 250 Zeichen. In Abbildung 4.1 ist zu sehen, wie Google die Beschreibung auf seinen Trefferlisten anzeigt.

Schlagwörter

```
<meta name="keywords"
  content="Stichwörter, Begriffe, Schlagwörter" />
```

Hier sollten Sie wenige, zentrale Begriffe angeben, die den Inhalt Ihrer Website charakterisieren. Sie werden durch Kommata getrennt und können aus einem oder mehreren Wörtern bestehen. Ganze Sätze haben an dieser Stelle jedoch nichts verloren.

Autor

```
<meta name="author" content="Martin Mustermann" />
```

Hier können Sie die Person oder mehrere Personen angeben, die verantwortlich für den Inhalt der Website sind.

Sie können beliebig weitere `meta`-Elemente einfügen, um Informationen über Ihr Dokument zu geben, Hauptsache, sie genügen den formalen Regeln. Hier einige Beispiele:

```
<meta name="author-personal" content="Martin Mustermann" />
<meta name="author-mail" content="mail@example.com" />
<meta name="date" content="2008-06-27T14:57:35+01:00" />
```

4.5.1 Metaangaben und Suchmaschinen

Suchmaschinen werten `meta`-Elemente kaum noch oder gar nicht mehr aus, daher hat ihre Bedeutung für das Gefundenwerden im World Wide Web stark abgenommen. Der Grund dafür liegt im Missbrauchspotenzial, das `meta`-Elemente bieten und das allzu oft ausgenutzt wird. Auf der Seite *http://de.selfhtml.org/html/kopfdaten/meta.htm* wird dies treffend ausgedrückt:

»Da die meisten Meta-Angaben keine unmittelbar nachvollziehbare Wirkung am Bildschirm haben und andererseits mit Suchdiensten zu tun haben, regen sie offenbar magisch die Phantasie mancher Geister an, die meinen, mit einem Mammutaufgebot irgendwo aufgeschnappter Meta-Angaben würden sie über Nacht zum Superstar, da ihre Homepage dann immer ganz oben bei den Suchtreffern eines Suchdienstes erscheint. Nachdem Seitenautoren massenhaft widersinnige und irreführende Meta-Angaben verwendet hatten, sind die namhaften Suchdienste dazu übergegangen, Meta-Angaben keine besondere Bedeutung mehr zuzumessen.«

Literatur

Klaus Schallhorn: Die Suchmaschinen-Site: Eintrag, Optimierung & Suchbegriffe
Auf dieser Site finden Sie alles, was Sie brauchen, wenn Sie bei sämtlichen großen Suchmaschinen eine Top-Ten-Position anstreben.
URL: *http://www.kso.co.uk/*

4.6 Universalattribute

Universalattribute sind solche Attribute, die im Start-Tag vieler oder fast aller XHTML-Elemente stehen dürfen. Diese Attribute haben verschiedene Aufgaben. Die in diesem Abschnitt vorgestellten Universalattribute können Elementen einen Namen geben oder ihnen eine Klasse zuweisen, Angaben über die Sprache des Elements treffen sowie Formatierungs- oder kommentierende Informationen hinzufügen.

4.6.1 Element-Identifikatoren: Die Attribute id und class

Die Universalattribute `id` und `class` dienen dazu, ein Element genauer zu bezeichnen.

DEFINITION

*Das **Attribut id** definiert einen eindeutigen Bezeichner für ein Element. Eine ID gleichen Namens darf in einem Dokument nur ein einziges Mal vorkommen.*

Funktionen von IDs

IDs haben folgende Funktionen:

- Definition eines Zielankers.
- Feinere Strukturierung des Quelltexts.

- Universelle Verarbeitung durch Benutzerprogramme (zum Beispiel bei der Identifizierung von Feldern beim Extrahieren von Daten aus Webseiten oder bei der Übersetzung von XHTML-Dokumenten in andere Formate, beispielsweise via **XSLT**).
- Ein Element einer bestimmten ID lässt sich in CSS über einen speziellen ID-Selektor ansprechen.
- Die JavaScript-Methode `getElementById()` greift auf ein Element zu, das die entsprechende ID besitzt.

DEFINITION

*Das **Attribut class** weist einem Element einen Klassennamen oder einen Satz von Klassennamen zu. Derselbe Klassenname kann einer beliebigen Anzahl von Elementen zugeordnet werden. Multiple Klassennamen werden durch Leerzeichen voneinander getrennt.*

Funktionen von Klassen

Klassen haben folgende Funktionen:

- Feinere Strukturierung des Quelltexts.
- Universelle Verarbeitung durch Benutzerprogramme.
- Ein Element oder auch Elemente einer bestimmten Klasse lassen sich über einen speziellen Klassenselektor ansprechen.

Fast jedem Element können eine ID und Klassen zugeordnet werden. Wie alle Attribute werden sie im Start-Tag des Elements notiert:

```
<div id="navigation">
<em class="wichtig">
```

Wenn Elemente zu mehreren Klassen gehören, zum Beispiel zu der Klasse `wichtig` und zu der Klasse `extern`, zählen Sie die Klassennamen mit einem Leerzeichen getrennt auf:

```
<em class="wichtig extern">
```

ACHTUNG

Element-Identifikatoren sollten nach Konzepten *benannt werden, nicht nach optischen Gesichtspunkten*[3]*. Ein Bezeichner `beispiel` ist sinnvoller als ein Bezeichner `blau-kursiv`. Das merken Sie spätestens dann, wenn Sie das Layout der Seite ändern und ein Beispiel eben nicht mehr blau und kursiv sein soll. Darüber hinaus erhöhen sinnvolle Bezeichnungen die Lesbarkeit des Quelltexts. IDs drücken Konzepte aus, die innerhalb eines Dokuments einmalig sind (`hauptmenue`, `inhalt`, `kontaktfeld`), während Klassen Konzepte beschreiben, die mehrfach vorkommen (können).*

3 siehe auch ***[Meiert 2008]***

4.6.2 Angabe der natürlichen Sprache: Die Attribute lang und xml:lang

Die meisten Webseiten werden in durchgehend einer Sprache verfasst. Werden jedoch mehrere Sprachen auf einer Seite eingesetzt, beispielsweise Zitate oder andere Texte in englischer Sprache auf einer primär deutschsprachigen Seite, müssen Sie die fremdsprachigen Texte in Hinblick auf die Sprachausgabe als solche auszeichnen.

Screenreader

Screenreader arbeiten mit Sprachtabellen, das heißt, sie erfassen einen Text und geben ihn entsprechend der voreingestellten Sprache an die Ausgabe weiter. Liest ein blinder Nutzer eine deutschsprachige Webseite, die aber nicht korrekt ausgezeichnete, fremdsprachige Texte enthält, werden diese mit hoher Wahrscheinlichkeit falsch vorgelesen.

DEFINITION

*Das **Attribut lang** spezifiziert in* XHTML-Dokumenten *die Sprache der Attributwerte und des Textinhalts eines Elements.*

Wie bereits mehrfach angesprochen, handelt es sich bei XHTML um einen XML-Dokumenttyp. In XML-Dokumenten kann das Attribut `xml:lang` verwendet werden, um die verwendete Sprache anzugeben.

DEFINITION

*Das **Attribut xml:lang** spezifiziert in* XML-Dokumenten *die Sprache der Attributwerte und des Textinhalts eines Elements.*

Kompatibilitätsrichtlinien

Die HTML-Kompatibilitätsrichtlinien (siehe Kapitel 4.4) empfehlen, für XHTML-Dokumente sowohl `lang` als auch `xml:lang` zu notieren und beiden Attributen den gleichen Wert zuzuweisen. Damit können sowohl (X)HTML- als auch XML-Benutzerprogramme die Sprache des Dokuments auslesen und sich entsprechend verhalten.

Vererbung

Bei einer einsprachigen Seite reicht es aus, die Sprache für das `html`-Element zu definieren. Diese Angabe wird an alle Nachfahrenelemente vererbt:

```
<html xmlns="http://www.w3.org/1999/xhtml"
  xml:lang="de" lang="de">
```

Enthält ein primär deutschsprachiges Dokument beispielsweise englischsprachige Inhalte, müssen diese als solche spezifiziert werden:

```
Die <acronym title="Extensible Hypertext Markup Language" lang="en"
xml:lang="en">XHTML</acronym>-Kompatibilitätsrichtlinien empfehlen:
»<span lang="en" xml:lang="en">Use both the <code>lang</code> and
<code>xml:lang</code> attributes when specifying the language of an
element.</span>«
```

Bestimmung der Sprache eines Elements

Die Sprache eines Elements wird in folgender Reihenfolge bestimmt (vom höchsten zum niedrigsten):

1. von einem für das Element selbst gesetzten `lang`- beziehungsweise `xml:lang`-Attribut
2. vom nächsten Vorfahrenelement, in dem das `lang`- beziehungsweise `xml:lang`-Attribut gesetzt ist (Vererbung)
3. vom *HTTP-Content-Language-Header* (der serverseitig konfiguriert sein kann)
4. von Standardwerten von Benutzerprogrammen und Benutzereinstellungen

Sprachen- und Länderkürzel

Der Wert der Attribute `lang` und `xml:lang` ist ein Sprachcode laut RFC 1766 ***[RFC-1766 1995]***. Diese Sprachenkürzel basieren auf den Normen ISO 639-1 zur Kurzbezeichnung von Sprachen und ISO 3166-1 zur Kurzbezeichnung von Ländern. Diese von den beiden ISO-Normen vordefinierten Kürzel erlauben solche Angaben wie `de` für Deutsch, `en` für Englisch oder `en-US` für US-Englisch. Mittlerweile gibt es RFC 3066 ***[RFC-3066 2001]***, nach dem auch dreibuchstabige Sprachenkürzel nach der Norm ISO 639-2 erlaubt sind.

Beispiele

Kurz gefasst, bestehen Sprachcodes aus einem primären Code und eventuell einer Serie von Untercodes, die mithilfe von Bindestrichen (`-`) abgeteilt werden. Listing 4.4, der XML-Empfehlung entnommen und erweitert, zeigt einige Beispiele:

```
<!-- Deutsch -->
<p lang="de" xml:lang="de">
  In welcher Straße hast du geparkt?</p>
<p lang="de-DE" xml:lang="de-DE">
  In welcher Straße hast du geparkt?</p>

<!-- Schweizer Deutsch -->
<p lang="de-CH" xml:lang="de-CH">
  In welcher Strasse hast du parkiert?</p>
```

```
<!-- Englisch -->
<p lang="en" xml:lang="en">Where did you park your car?</p>

<!-- Spanisch -->
<p lang="es" xml:lang="es">¿Dónde has aparcado tu coche?</p>
```

Listing 4.4: Beispiele für Sprachcodes

Literatur

ISO 639, ISO 3166, ISO-3166-1-Kodierliste
Ausführliche Wikipedia-Grundlagenartikel mit Sprachen- und Länderkürzeltabellen.
URL: *http://de.wikipedia.org/wiki/ISO_639*
URL: *http://de.wikipedia.org/wiki/ISO_3166*
URL: *http://de.wikipedia.org/wiki/ISO_3166-1_Kodierliste*

4.6.3 Inzeilige Formatierung: Das Attribut style

DEFINITION

*Das **Attribut style** gibt Formatierungsinformationen für das aktuelle Element an. Die Syntax des Werts wird von der Stylesheetsprache festgelegt, üblicherweise CSS.*

```
<div style="border: 1px solid white;">
  Container mit weißem Rahmen</div>
```

Weitere Informationen zum `style`-Attribut erhalten Sie in Kapitel 6.3.1.

4.6.4 Kommentierende Informationen: Das Attribut title

DEFINITION

*Das **Attribut title** bietet kommentierende Informationen über das Element, für das es gesetzt ist.*

Darstellung

Das `title`-Attribut kann von Benutzerprogrammen auf verschiedene Art und Weise dargestellt werden. Screenreader können den Attributwert vorlesen, visuelle Browser können es beispielsweise als Tooltipp (siehe Abbildung 4.6) oder als Text in der Statusleiste anzeigen, wenn Nutzer das Element mit der Maus überfahren. In den meisten Fällen ist der Inhalt des `title`-Attributs nicht unmittelbar zugänglich, daher sollte es nur ergänzende Informationen enthalten, zum Beispiel Informationen, die für die Bedienung oder das Verständnis eines Formulars hilfreich sein können, oder Hinweise zum Ziel eines Hyperlinks.

```
<input type="text" name="website" title="Angabe in der Form http://
www.example.com/" />

<strong title="Hier befinden Sie sich gerade">Webentwicklung</
strong>
```

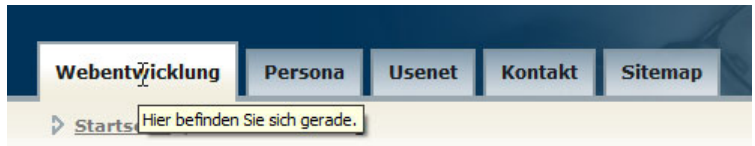


Abbildung 4.6: Das `title`-Attribut wird häufig als Tooltipp dargestellt

5 Auszeichnung der Inhalte

5.1 Überschriften und Absätze

5.1.1 Überschriften: Die Elementtypen h1, h2, h3, h4, h5 und h6

DEFINITION

Eine Überschrift beschreibt einführend das Thema eines Abschnitts. Dokumente enthalten ein oder mehrere `h1`*, die die Hauptüberschriften (Überschriften der ersten Ordnung) der jeweiligen Abschnitte auszeichnen. Die Überschriften* `h2` *sind dann die gering weniger wichtigen Überschriften (die Überschriften zweiter Ordnung) der Unterabschnitte. Überschriften dritter Ordnung* `h3` *unterteilen wiederum diese Unterabschnitte. Dieses Prinzip lässt sich hierarchisch weiter fortführen bis zur Überschrift* `h6`*, der geringstwichtigen Überschrift.*

Darstellung

Wenn Sie einen Blick auf Abbildung 5.1 werfen, werden Sie sehen, dass Ihr Browser die Überschriften je nach Ordnung unterschiedlich groß und fett oder kursiv darstellt. Die Darstellung lässt sich allerdings beliebig mit CSS beeinflussen. Ihr individuelles Empfinden

der einzelnen Schriftgrößen als zu groß, zu klein oder genau richtig sollte Sie daher nicht davon abhalten, Überschriften inhaltlich und logisch richtig einzusetzen. Dabei sollten Sie folgende Regeln beachten:

Regeln beim Definieren von Überschriften

- Wenn das Dokument für sich allein steht, also ein in sich abgeschlossenes Thema behandelt, sollte es genau eine Überschrift `h1` geben, die dem `title`-Element des Dokuments entspricht.
- In Bedingung 3.5 der BITV heißt es:

  »Zur Darstellung der Struktur von mittels Markup-Sprachen geschaffener Dokumente sind Überschriften-Elemente zu verwenden.«

  Für Nutzer assistiver Technologien, beispielsweise Screenreader, gehören Überschriften zu den wichtigsten Strukturelementen eines Dokuments. So kann aus den Überschriften ein Inhaltsverzeichnis generiert oder von Überschrift zu Überschrift gesprungen werden – nicht zu unterschätzende Navigationshilfen!
- Überschriften bilden eine *Hierarchie* ab. Daher gilt es als schlechter Stil, Überschriftenebenen zu überspringen. So sollte beispielsweise auf eine Überschrift erster Ordnung niemals eine Überschrift dritter Ordnung folgen, sondern stets nur Überschriften zweiter Ordnung. Ein Durcheinander der Überschriftenhierarchie verwirrt vor allem Nutzer von Screenreadern, die sich anhand von Überschriften orientieren.

5.1.2 Absätze: Der Elementtyp p

DEFINITION

*Das **Element p** zeichnet einen Textabsatz aus. Es darf keine Blockelemente enthalten. Absätze dienen primär der Strukturierung des Texts Ihrer Website.*

Zwischen Textabsätzen hält der Browser automatisch einen gewissen Abstand ein. Sehen Sie sich dazu einfach Abbildung 5.1 an.

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

<html xmlns="http://www.w3.org/1999/xhtml"
  lang="de" xml:lang="de">
  <head>
    <meta http-equiv="Content-Type"
      content="text/html; charset=utf-8" />
```

```
    <title>Überschriften und Absätze</title>
  </head>

  <body>
    <h1>Überschrift erster Ordnung</h1>
    <p>Lorem ipsum dolor sit amet, consectetuer (...)</p>
    <p>Duis autem vel eum iriure dolor in hendrerit (...)</p>
    <h2>Überschrift zweiter Ordnung</h2>
    <p>Lorem ipsum dolor sit amet, consectetuer (...)</p>
    <h3>Überschrift dritter Ordnung</h3>
    <p>Lorem ipsum dolor sit amet, consectetuer (...)</p>
    <h3>Überschrift dritter Ordnung</h3>
    <p>Lorem ipsum dolor sit amet, consectetuer (...)</p>
    <h2>Überschrift zweiter Ordnung</h2>
    <p>Lorem ipsum dolor sit amet, consectetuer (...)</p>
  </body>
</html>
```

Listing 5.1: Überschriften und Absätze

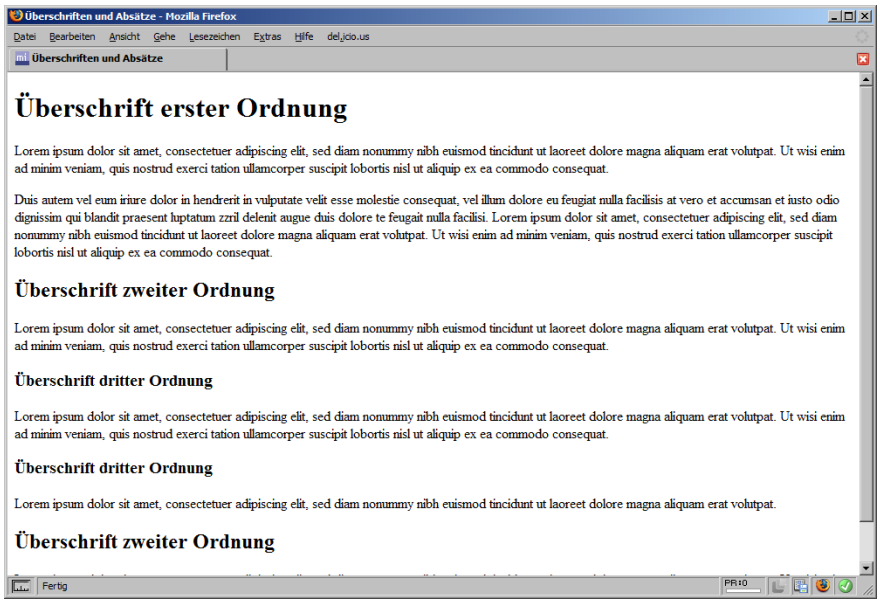


Abbildung 5.1: Die Darstellung von Listing 5.1

Lorem ipsum dolor sit amet

Sie werden sich vielleicht fragen, was der Text in diesem Beispiel bedeuten soll. Es handelt sich hierbei um einen klassischen, in ähnlicher Form seit Jahrhunderten verwendeten Lückenfüller zum Testen des Schriftsatzes im Druckbereich, siehe *http://www.lorem-ipsum.info/blindtext-lorem-ipsum*. Der Hintergrund ist, zu vermeiden, dass die Information im Text vom Layout ablenkt. Für diese Einführung ist der Text gut geeignet, schließlich steht in den Beispielen die Anwendung der XHTML-Elemente (und später der CSS-Eigenschaften und -Werte) im Vordergrund, nicht der textuelle Inhalt. Viele Beispiele werden diesen Dummy-Text verwenden.

Automatischer Umbruch

Spätestens hier, wenn Sie einen längeren Text mit `p` ausgezeichnet haben, werden Sie eine der Stärken von XHTML festgestellt haben: Der Browser bricht den Text automatisch am Fensterrand um. Wenn Sie das Browserfenster verkleinern und in der Größe verändern, wird der Text mit der Fenstergröße skaliert. Dieses Verhalten werden Sie bei fast allen Elementen beobachten. Diese Fähigkeit von XHTML, sich an das Ausgabemedium anzupassen, ist ein Grund für die Plattformunabhängigkeit des Mediums World Wide Web.

5.1.3 Zeilenumbrüche: Der Elementtyp br

DEFINITION

*Das **leere Element br** umbricht (beendet) zwingend die aktuelle Textzeile.*

Ein Gedicht, beispielsweise »Der Panther« (1903) von Rainer Maria Rilke, könnten Sie wie folgt auszeichnen:

```
<p>Sein Blick ist vom Vorübergehn der Stäbe<br />
  so müd geworden, dass er nichts mehr hält.<br />
  Ihm ist, als ob es tausend Stäbe gäbe<br />
  und hinter tausend Stäben keine Welt.</p>

<p>Der weiche Gang geschmeidig starker Schritte,<br />
  der sich im allerkleinsten Kreise dreht,<br />
  ist wie ein Tanz von Kraft um eine Mitte,<br />
  in der betäubt ein großer Wille steht.</p>

<p>Nur manchmal schiebt der Vorhang der Pupille<br />
  sich lautlos auf -. Dann geht ein Bild hinein,<br />
  geht durch der Glieder angespannte Stille -<br />
  und hört im Herzen auf zu sein.</p>
```

Listing 5.2: Sinnvoller Einsatz des Elements `br`

Mehrere Zeilenumbrüche hintereinander sollten Sie vermeiden. Eine Leerzeile erreichen Sie durch Absätze, vertikale Abstände zwischen Elementen mit CSS.

5.2 Links und Anker

Der große Unterschied zwischen XHTML und den meisten anderen Auszeichnungssprachen ist die Tatsache, dass es sich bei XHTML-Dokumenten um Hypertext handelt. Das wichtigste Element im Hypertext ist der Link.

DEFINITION

*Ein **Link** (auch: Hyperlink, Weblink) ist ein Verweis von einem **Quellanker** zu einem **Zielanker**, in den meisten Fällen von einem Dokument zu einem anderen oder von einer Stelle in einem Dokument zu einer anderen im selben Dokument (Fragmentlink), der über ein Benutzerprogramm effizient verfolgt werden kann.*

Der Zielanker kann irgendeine Ressource sein, zum Beispiel ein Bild, ein Film, ein Programm, ein Download, ein XHTML-Dokument oder ein bestimmtes Element innerhalb eines XHTML-Dokuments. Wie der Browser sich beim Aktivieren eines Links verhält, ist abhängig von den Browser- und Betriebssystemeinstellungen. So kann er die Zielressource im Browserfenster anzeigen, zum Herunterladen anbieten oder sie an ein anderes, passenderes Programm übergeben.

5.2.1 Anker: Der Elementtyp a

DEFINITION

*Das **Element a** definiert*

- *einen Zielanker für beliebig viele Links, wenn ein Attribut `name` definiert wird (`<a name="bezeichner">Ankertext</a>`), oder*
- *einen Quellanker[1] genau eines Links, wenn er ein Attribut `href` enthält (`<a href="uri">Ankertext</a>`). Der Wert des `href`-Attributs ist ein URI.*

Darstellung

Browser stellen Links so dar, dass sie für den Nutzer als solche erkennbar sind. Die exakte Wiedergabe hängt von der Art des Benutzerprogramms ab. Grafische Browser unterstreichen Links und unterscheiden in der Farbdarstellung zwischen besucht und unbesucht. Wurde der Zielanker noch nicht besucht, wird der Linktext blau angezeigt, wurde er bereits besucht, dann erscheint der

1 Der Einfachheit halber wird ein Quellanker im Allgemeinen auch als Link bezeichnet. Diese sprachliche Ungenauigkeit übernehme ich ebenfalls in dieser Einführung.

Linktext in Violett. Dieses Verhalten lässt sich durch CSS beeinflussen. Welche Ressourcen ein Nutzer bereits besucht hat, merkt der Browser sich in seiner History.

Mauszeiger

Fährt man mit dem Mauszeiger über einen Link, verändert er sich in der Regel von einem einfachen Pfeil zu einer Hand mit ausgestrecktem Zeigefinger (siehe Abbildung 5.2).

Ein Link auf die Onlinevariante dieses Buchs kann wie folgt aussehen:

```
<a href="http://jendryschik.de/wsdev/einfuehrung/">Einführung in
XHTML, CSS und Webdesign</a>
```

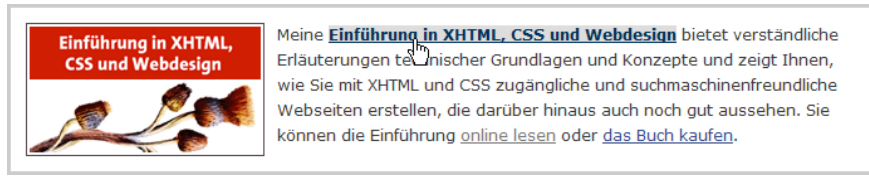


Abbildung 5.2: Der Mauszeiger verändert sich beim Überfahren eines Links

Ressourcen zum Download anbieten

XHTML bietet keine spezielle Notation, um Dateien explizit zum Download anzubieten. Wie bereits erwähnt, wird eine Zielressource je nach Server-, Browser- und Systemkonfiguration entweder im Browserfenster dargestellt, zum Herunterladen angeboten oder an ein anderes Programm übergeben, das diese Ressource dann anzeigt oder anderweitig verarbeitet. Entscheidend ist, mit welchem MIME-Typ der Webserver die Ressource ausliefert. Welche dieser drei Möglichkeiten gewählt wird, kann ein Webautor *nicht* beeinflussen!

Um ein Dokument zum Download anzubieten, setzen Sie folglich einen einfachen Link:

```
<a href="http://example.com/downloads/file.zip">Download</a>
```

Fragmentlinks: In die Mitte von Webseiten verlinken

Bei sehr umfangreichen Dokumenten kann es sinnvoll sein, Querverweise innerhalb des Dokuments zu setzen oder ein Inhaltsverzeichnis an den Anfang zu stellen, das zu den einzelnen Abschnitten (Fragmenten) des Dokuments oder anderer Dokumente verweist. Ursprünglich realisierten Webautoren dies über ein Element `a` mit dem Attribut `name`.

Kennzeichnung mit name

Angenommen, Sie möchten von einem Ursprungsanker aus auf den zweiten Abschnitt eines Dokuments foo.html verweisen. Zunächst definieren Sie den Zielanker:

```
<h2><a name="abschnitt2">Zweiter Abschnitt</a></h2>
```

oder:

```
<h2><a name="abschnitt2"></a>Zweiter Abschnitt</h2>
```

Kennzeichnung mit id

Das Attribut name ist in diesem Zusammenhang jedoch veraltet! Webautoren sollten Zielanker über das id-Attribut kennzeichnen:

```
<h2 id="abschnitt2">Zweiter Abschnitt</h2>
```

In vielen Fällen wird auf Fragmente verwiesen, die ohnehin schon über ein id-Attribut verfügen. Darüber hinaus ist das name-Attribut in XHTML (nicht in HTML) missbilligt (**deprecated**), insofern ist für alle Versionen nach XHTML 1.0 diese Vorgehensweise die einzig mögliche. Lediglich sehr alte Browser wie der Netscape Navigator 4 können id-Attribute nicht als Zielanker interpretieren.

Der Verweis auf den Fragmentanker ist ein ganz normaler Link zur Seite foo.html, jedoch hängen Sie eine Raute (#) und den Namen des Ankers an:

```
<a href="foo.html#abschnitt2">zweiter Abschnitt</a>
```

Beziehungen: Die Attribute rel und rev

Die Attribute rel und rev nehmen entgegengesetzte Rollen ein – das rel-Attribut spezifiziert eine *Vorwärts-*, das rev-Attribut eine *Rückwärtsbeziehung*.

DEFINITION

*Das **Attribut rel** beschreibt die Beziehung des aktuellen Dokuments zu der durch das href-Attribut angegebenen Zielressource. Das **Attribut rev** dient dazu, einen Rückwärtslink von der durch das href-Attribut angegebenen Zielressource zum aktuellen Dokument zu beschreiben. Der Wert beider Attribute ist eine durch Leerzeichen getrennte Liste von Beschreibungen.*

So sagt beispielsweise

```
<a href="glossar" rel="glossary">Glossar</a>
```

aus, dass die verlinkte Ressource ein Glossar für das aktuelle Dokument bereitstellt. Mit

```
<a href="foo" rev="contents">Foo</a>
```

wird angezeigt, dass das aktuelle Dokument ein Inhaltsverzeichnis (*table of contents*) für die verlinkte Ressource darstellt. Die Information zur Vorwärtsbeziehung mit `rel` ist meistens etwas nützlicher und interessanter als die Rückwärtsbeziehung, insofern ist das Attribut `rel` häufiger anzutreffen als sein Äquivalent `rev`.

Bei zusammenhängenden Webseiten könnten die zwei aufeinanderfolgenden Dateien `foo.html` und `bar.html` gegenseitig verlinkt und mit den Attributwerten `next` (nächste/r/s) und `prev` (*previous*, vorherige/r/s) in Beziehung gesetzt werden. Dazu notieren Webautoren im Dokument `foo.html`:

```
<a href="bar" rel="next" rev="prev">Weiter</a>
```

und im Dokument `bar.html`:

```
<a href="foo" rel="prev" rev="next">Zurück</a>
```

Attributwerte für die Attribute `rel` und `rev` finden sich unter anderem in der HTML 4.01-Spezifikation, aber auch in HTML 3.2 und in DTD-Kommentaren von (X)HTML sowie in mehreren nicht offiziellen Arbeitsentwürfen und Dokumenten. Einige Browser unterstützen auch weitere Werte. Zu den am häufigsten verwendeten Werten gehören die im Folgenden vorgestellten.

`nofollow`

Am 18. Januar 2005 kündigte Google an, gemeinsam mit MSN und Yahoo einen neuen Wert für das Attribut `rel` einzuführen: `nofollow`. Damit erreichen Sie, dass Suchroboter Links zwar verfolgen, den Inhalt der verlinkten Ressource jedoch nicht in den Index aufnehmen und neu bewerten. Der Grund für diese Lösung ist Spam (unerwünschte Werbung für Webseiten) in Gästebüchern und Weblog-Kommentarsystemen. Der Sinn ist jedoch zweifelhaft; schließlich gehört starke Verlinkung der Weblogs untereinander zur Blog-Kultur, und auch diese hilfreichen und weiterführenden Links würden mit dieser Methode abgewertet werden.

XHTML Friends Network

Eine weitere Anwendung sind die bei vielen Webautoren beliebten im *XHTML Friends Network*[2] definierten Linkbeziehungen. So kann ein Webautor oder Blogger beispielsweise beim Verlinken eines Kollegen angeben, dass er diese Person als Freund ansieht, bereits im echten Leben getroffen hat und in seiner Nähe wohnt:

```
<a href="http://www.intermitto.net/ich/"
  rel="friend met co-resident">Jens</a>
```

Mit CSS könnten diese Beziehungen farblich oder auch über Icons sichtbar gemacht werden. Spezialisierte Suchmaschinen und Social-Network-Tools könnten diese Daten ebenfalls auswerten. Natürlich ist diese Kennzeichnung

2 *http://www.gmpg.org/xfn/*

eher eine Spielerei als eine besonders nützliche Anwendung – aber immerhin zeigt sie, dass diese Art von Metadaten in Linkattributen genutzt wird.

tag

Auf vielen Websites ist es mittlerweile üblich, dass Autoren ihre Inhalte verschlagworten, auf Neudeutsch: taggen. Dabei weisen Autoren ihren Inhalten (darunter Texte, Videos, Fotos, Links oder Nutzerprofile) Begriffe (»Tags«) zu, von denen sie glauben, dass diese sie gut beschreiben. Tags sind im Allgemeinen verlinkt und führen zu Übersichtsseiten, auf denen alle Inhalte des Webangebots angezeigt werden, die mit diesem Tag verschlagwortet worden sind. Um Tag-Links als solche zu kennzeichnen, verwenden Sie das Attribut `rel="tag"`. Dies ist ein Beispiel für einen Link auf die Auflistung aller Ressourcen, die Nutzer bei Mister Wong[3], einem Social-Bookmark-Dienst, mit dem Tag »CSS« verschlagwortet haben:

```
<a href="http://www.mister-wong.de/tags/css/" rel="tag">CSS</a>
```

home

Einen Link zur Homepage einer Website kennzeichnen Sie mit dem Attribut `rel="home"`, beispielsweise wie folgt:

```
<a href="/" rel="home">Startseite</a>
```

Bereits heute gibt es spezielle Suchdienste, die Beziehungen von Dokumenten untereinander auswerten. Zukünftige, intelligente Browser und Suchmaschinen können dies noch verstärkt tun und damit bessere Navigation, Orientierung und Suchergebnisse bieten. Es schadet also nicht, sich jetzt schon einmal damit zu beschäftigen und – wenn es nicht zu viel Arbeit verursacht – einige Relationen in den Webseiten zu verwenden.

Weitere Relationen lernen Sie im folgenden Abschnitt über das `link`-Element kennen.

5.2.2 Dokumentbeziehungen: Der Elementtyp link

DEFINITION

*Das **leere Element link** vermittelt Informationen über die Beziehungen zweier Ressourcen, die von Benutzerprogrammen auf unterschiedliche Weise wiedergegeben werden können. `link`-Elemente dürfen nur im `head` eines Dokuments stehen. Über das Attribut `rel` oder `rev` wird der Typ der Beziehung angegeben, `href` definiert das Verweisziel. Das Attribut `title` kann von Benutzerprogrammen als Beschriftung des `link`-Elements herangezogen werden, wenn es dargestellt wird.*

3 *http://www.mister-wong.de*

Anwendung

Webautoren können `link`-Elemente unter anderem dafür verwenden, folgende Arten von Verweisen anzugeben:

- Verweise zu alternativen, in anderen Sprachen geschriebenen Versionen eines Dokuments,
- Verweise zu alternativen, für andere Medien gestalteten Versionen eines Dokuments, beispielsweise einer speziell für den Druck aufbereiteten Fassung,
- Verweise zur Startseite einer Sammlung von Dokumenten.

Damit könnte nicht nur die Behandlung eines Dokuments durch Suchmaschinen verbessert, sondern seitens der Browser auch eine völlig andere Art der Navigation bereitgestellt werden, beispielsweise in Form einer separaten Navigationsleiste, die die Dokumentbeziehungen auswertet und gruppiert. Leider ist dies nur graue Theorie. Das `link`-Element gehört zwar seit 1995 zum HTML-Sprachumfang, wird jedoch noch heute von kaum einem Benutzerprogramm ausgewertet. Die Browser iCab und Opera blenden auf Wunsch eine derartige Navigationsleiste ein, für Firefox gibt es eine entsprechende Erweiterung, die nachinstalliert werden kann. Auch der Textbrowser Lynx unterstützt `link`-Elemente.

Abbildung 5.3: Navigationsleiste in Opera bei der Darstellung der Cascading Style Sheets Home Page

Diese Browser können unter anderem folgende Angaben auswerten:

Angabe	Bedeutung
`rel="alternate"`	Bezeichnet andere Versionen des Dokuments. Diese Beziehung wird oft in Verbindung mit den Attributen `hreflang` und `xml:hreflang` oder `media` verwendet und bezeichnet dann eine Übersetzung oder eine Umsetzung für ein anderes Medium.
`rel="appendix"`	Verweist zum Anhang.
`rel="author"`	Verweist zu einer Seite mit Informationen über den Autor des Dokuments.
`rel="bookmark"`	Verweist zu einem allgemeinen Orientierungspunkt oder Schlüsselpunkt.
`rel="chapter"`	Verweist zum Anfang des Kapitels.
`rel="contents"`	Verweist zu einer Inhaltsübersicht oder einer Sitemap.
`rel="copyright"`	Verweist zu einem Copyrightvermerk für die aktuelle Seite.
`rel="first"`	Verweist zu dem ersten Dokument einer Folge von Dokumenten.
`rel="glossary"`	Verweist zu einer Seite mit Begriffserläuterungen für das aktuelle Dokument.
`rel="help"`	Verweist zu einer Hilfeseite für das Dokument oder die gesamte Website.
`rel="index"`	Verweist zu einem Dokument, das einen Index oder eine Übersicht zu dem aktuellen Dokument enthält.
`rel="last"`	Verweist zu dem letzten Dokument einer Folge von Dokumenten.
`rel="next"`	Verweist zu dem folgenden Dokument einer Folge von Dokumenten.
`rel="prev"`	Verweist zu dem vorhergehenden Dokument einer Folge von Dokumenten.
`rel="search"`	Verweist zur Suchfunktion.
`rel="section"`	Verweist zum Anfang des Abschnitts.
`rel="start"`	Verweist zur Startseite (wie `rel="top"`).
`rel="stylesheet"`	Diese mit Abstand meistverwendete Art von `link`-Elementen verweist auf ein externes Stylesheet. Mehr dazu lesen Sie in Kapitel 6.3.3.
`rel="subsection"`	Verweist zum Anfang des Unterabschnitts.
`rel="top"`	Verweist zur Startseite (wie `rel="start"`).
`rel="up"`	Verweist zur übergeordneten Kapitelseite.

Tabelle 5.1: Mögliche Angaben für das `link`-Element und deren Bedeutung

5.3 Bilder

5.3.1 Grafiken einbinden: Der Elementtyp img

DEFINITION

*Das **leere Element img** bettet ein Bild an der Stelle in das aktuelle Dokument ein, an der das Leeres-Element-Tag notiert ist.*

`img`-Elemente verfügen üblicherweise über die Attribute `src`, `width`, `height` und `alt`.

DEFINITION

*Das **Attribut src** gibt den Ort der Bildquelle über einen URI an.*

DEFINITION

*Die **Attribute width** und `height` in Pixel oder Prozent weisen Benutzerprogramme an, die Originalbildgröße zugunsten ihrer Werte (Pixel oder Prozent) zu überschreiben; das Bild wird entsprechend skaliert.*

Berechnung prozentualer Längen

Prozentual angegebene Längen beziehen sich auf den horizontal oder vertikal verfügbaren Raum, der dem Bild zur Verfügung steht, und nicht auf dessen Originalgröße. Steht beispielsweise einem 300 Pixel breiten Bild ein horizontaler Raum von 800 Pixeln zur Verfügung und wird eine Breite von 50 Prozent angeben, wird das Bild nicht auf 150 Pixel Breite skaliert (50 Prozent von 300), sondern auf 400 Pixel (50 Prozent von 800).

Bei Rastergrafiken wie beispielsweise PNG, GIF oder JPEG, also Grafiken, die aus einer matrixförmigen Anordnung von Pixeln bestehen, denen jeweils eine Farbe zugeordnet ist, führt eine Skalierung zu unschönen Effekten, da die Bilder verlustbehaftet einfach nur in die Länge gezogen oder gestaucht werden können.

Breite und Höhe stets angeben

Es gehört zum guten Stil, die Breite und Höhe eines Bilds stets anzugeben, damit Browser den benötigten Platz reservieren und das Dokument weiter aufbauen können, während die Bilddaten noch geladen werden. Fehlen diese Attribute, führt dies zu unschönen Effekten beim Seitenaufbau; die Seite wird zunächst ohne Bilder angezeigt und muss sich schließlich neu zurechtruckeln, sobald die Bilder geladen sind.

DEFINITION

*Das **Attribut alt** gibt alternativen Text für Benutzerprogramme an, die Bilder nicht darstellen können, beispielsweise Screenreader oder Textbrowser, aber auch Suchmaschinen.*

Das `alt`-Attribut muss für jedes `img`-Element angegeben werden.

Es ist nicht einfach, für jede Grafik stets den richtigen Alternativtext zu finden. Hilfreich ist es, zwischen drei Arten von Grafiken zu unterscheiden:

Drei Typen von Grafiken

- **Bilder, die echten Inhalt aufweisen**

  Dazu gehören alle Grafiken, die informationstragende Schriftzüge enthalten, in vielen Fällen das Logo des Webauftritts, als Grafiken realisierte Überschriften oder grafische Schaltflächen und Symbole. Der im Bild enthaltene Text muss im Alternativtext wiedergegeben werden; es ist am besten, wenn der im Bild enthaltene Text und der Alternativtext identisch sind oder nur leicht voneinander abweichen.

```
<img src="img/webkrauts-kopf.gif" width="684" height="202"
  alt="Webkrauts: Webstandards für ein besseres Web!" />
```

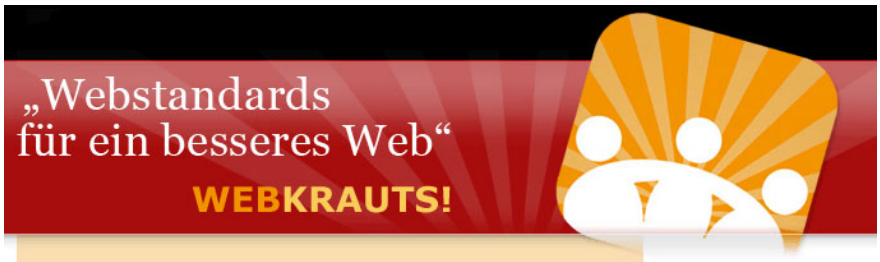


Abbildung 5.4: Webkrauts-Schriftzug

- **Fotos und grafische Darstellungen**

  Dazu gehören Diagramme und schematische Darstellungen von Zusammenhängen ebenso wie Fotos von der letzten Betriebsfeier. Hier erfüllt das alternative Attribut eher eine beschreibende Funktion.

```
<img src="/images/weihnachtsfeier/2004/42"
  width="400" height="300"
  alt="Maximilian, Alexander und Marie auf der Tanzfläche" />
```

- **Layoutgrafiken**

  Alle Grafiken, die zur Dekoration oder aus Gründen der Darstellung verwendet werden und keine Information transportieren, sind Layoutgrafiken. Dazu gehören Hintergrundgrafiken, Icons, Trennlinien und das in

düsteren Zeiten des Webdesigns beliebte transparente GIF zur Positionierung und Ausrichtung von Inhalten. Bei derartigen Bildern führen Sie das `alt`-Attribut natürlich auf, lassen es aber leer. Alle nicht visuellen Benutzerprogramme, also auch Suchmaschinen und Screenreader, beachten die Layoutgrafik daraufhin nicht.

```
<img src="pix.gif" width="2" height="500" alt="" />
```

Layoutgrafiken haben im Sinne der konsequenten Trennung von Markup und Layout im XHTML-Quelltext eines Dokuments nichts zu suchen und sollten mittels CSS in eine Seite eingebunden werden. Mehr dazu lesen Sie in Kapitel 7.2.

Tooltipps mit `title`

Im Zusammenhang mit dem `alt`-Attribut kommt es immer wieder zu einem schweren Missverständnis: Viele Webdesigner glauben, mithilfe des alternativen Attributs erzeuge man einen Tooltipp. Dieser Irrglaube resultiert aus der Tatsache, dass veraltete Browser den Inhalt des alternativen Attributs als Tooltipp darstellen – dieses Verhalten ist entsprechend den Empfehlungen zwar nicht falsch, aber sehr unglücklich, wie sich immer wieder zeigt. Tooltipps bei Bildern erreichen Webautoren wie bei jedem Element über das `title`-Attribut. Ein Tipp: Möchten Sie verhindern, dass das alternative Attribut auch als Tooltipp angezeigt wird, notieren Sie zusätzlich `title=""`.

```
<img src="/images/weihnachtsfeier/2004/42"
  width="400" height="300" title=""
  alt="Maximilian, Alexander und Marie auf der Tanzfläche" />
```

5.4 Objekte

5.4.1 Objekte einbetten: Der Elementtyp object

Ginge es nach dem Willen der Autoren der HTML-Empfehlungen, wäre das `img`-Element schon vor Jahren von der Bildfläche verschwunden. Unrecht haben sie damit nicht, denn es wirft einige unbeantwortete Fragen auf:

Kritik am `img`-Element

- Warum gibt es ein Element zur Einbettung von Bildern, aber keines zur Einbettung von Audio-, Video oder anderen Multimedia-Dateien? Oder anders gefragt: Weshalb gibt es kein Element, das beliebige Medien einbetten kann, also auch Bilder, und das `img`-Element damit überflüssig macht?
- Reicht der Wert des alternativen Attributs, eine einfache Zeichenkette, wirklich aus, um ein Bild zu ersetzen? Wie kann die Zugänglichkeit dieser Alternative gewährleistet werden, wenn Webautoren Abkürzungen, Akronyme oder Sprachwechsel nicht auszeichnen können?

Das Element `object` soll diese Probleme lösen.

DEFINITION

*Das **Element object** erlaubt Webautoren die Einbindung von Ressourcen, die Browser nicht nativ, also »von Haus aus«, unterstützen, zum Beispiel spezielle Grafiken, Videos, Musikdateien, XHTML-Dokumente, ein Word-Dokument, ein Java-Applet oder andere Programme. XHTML bietet über das `object`-Element eine einheitliche Syntax zur Einbettung beliebiger Objekte, anstatt für jeden Datentyp ein eigenes Element bereitzustellen.*

Verarbeitung von Objekten

Eine per `object`-Element eingebettete Ressource kann nur dann dargestellt werden, wenn das System des Nutzers mit der Art der Ressource etwas anfangen kann, also das Benutzerprogramm die Ressource verarbeiten kann oder ein entsprechendes Plug-in installiert ist. Ob das Objekt innerhalb der Webseite angezeigt werden kann, hängt von dem Programm ab, das zur Anzeige des Objekts herangezogen wird. Kann das System des Nutzers die eingebettete Ressource nicht verarbeiten, wird der Alternativinhalt herangezogen. Dieser beschränkt sich beim `object`-Element im Gegensatz zum Element `img` nicht nur auf den Wert eines alternativen Attributs. Das `object`-Element darf Block- und Inline-Elemente als Kinder enthalten und stellt so vielfältige Möglichkeiten zur Auszeichnung des alternativen Inhalts zur Verfügung. So ist es denkbar, einen kurzen Film durch eine Folge von statischen Bildern zu ersetzen, deren Bildunterschriften die Geschichte des Films nacherzählen.

Nicht jeder Browser interpretiert das Element `object` so, wie die Empfehlung es vorschreibt, leider auch nicht bei der Einbettung von Grafiken. Dies ist der Grund dafür, dass das `img`-Element sich bis heute außerordentlicher Beliebtheit erfreut und ein Großteil der Webdesigner sicherlich nicht einmal weiß, dass Bilder auch über das `object`-Element eingebettet werden können. Zum Teil liegt dies auch an der Vielzahl der proprietären Multi- und Hypermedia-Formate und ihren unterschiedlichen Schnittstellen. Für Standardgrafiken sollten Sie weiterhin auf das `img`-Element zurückgreifen.

`object` verfügt über eine ganze Reihe an Attributen, von denen jeweils eine unterschiedliche Menge zur Einbettung der verschiedenen Medientypen verwendet werden kann oder muss.

Einbettung von SVG

Es gibt mehrere Möglichkeiten, SVG-Dateien in Webseiten einzubinden; die Einbettung über das Element `object` ist die gebräuchlichste und die von den meisten Browsern am besten interpretierte. Kann die SVG-Datei nicht angezeigt werden, wird der Inhalt des `object`-Elements als Alternative herangezogen – in Listing 5.3 dieselbe Grafik im GIF-Format.

```
<div>
  <object type="image/svg+xml"
    data="../beispiele/example.svg"
    width="460" height="570">
    <param name="src" value="../beispiele/example.svg" />
    <img src="../images/example.gif" width="425"
      height="533" alt="" />
  </object>
</div>
```

Listing 5.3: SVG einbinden

DEFINITION

*SVG-Dateien sind Datenobjekte und werden über das **Attribut data** in Form eines URI referenziert.*

DEFINITION

*Das **Attribut type** gibt den MIME-Typ der Daten an, die vom `data`-Attribut referenziert werden, in diesem Fall `image/svg+xml`. Unterscheidet sich der Wert dieses Attributs vom HTTP-MIME-Typ, der vom Server zurückgegeben wird, wenn das Objekt empfangen wird, ist der HTTP-MIME-Typ vorrangig.*

DEFINITION

*Die **Attribute width** und `height` legen die Breite und Höhe des Objekts in Pixel oder Prozent fest.*

Stets Breite und Höhe angeben

Bei Daten wie beispielsweise Videos, die über feste Dimensionen verfügen, sollten Sie die Originalbreite und -höhe angeben. Darüber hinaus sollten Sie die Anzeige des Objekts in verschiedenen Browsern überprüfen; es kann passieren, dass Rollbalken angezeigt werden, wenn die angegebenen Dimensionen aus irgendwelchen Gründen nicht ausreichen. In diesem Fall müssen Sie so lange verschiedene Werte ausprobieren, bis die Darstellung browserübergreifend zufriedenstellend ist.

Nicht alle Browser zeigen die SVG-Datei an, wenn sie ausschließlich über das `data`-Attribut referenziert wird. Der Internet Explorer beispielsweise benötigt einen zusätzlichen Parameter.

DEFINITION

*Das **leere Element param** gibt einen Wert an, der von einem Objekt zur Laufzeit benötigt wird. Welche Parameter benötigt werden, ist abhängig vom einzubettenden Objekt. Die Reihenfolge ist beliebig. Das Attribut `name` definiert den Namen des Laufzeitparameters, das Attribut `value` dessen Wert.*

Einbettung von Flash

DEFINITION

Flash *ist eine Kombination aus einem Grafikprogramm und einer Animationssoftware zur Erzeugung von Multimedia-Dateien im auf Vektorgrafiken basierenden SWF-Format der amerikanischen Firma* Adobe *(vormals* Macromedia*). Flash ermöglicht es, Filme und interaktive Elemente wie beispielsweise Spiele in Webseiten zu integrieren. Zur Anzeige von Flash benötigt man ein Plug-in, den sogenannten Flash Player, den man sich beim Hersteller herunterladen kann.*

Auch zur Einbindung von Flash gibt es verschiedene Ansätze, wobei der gebräuchlichste dem zur Einbindung von SVG-Dateien gleicht. Der Unterschied liegt in der Anzahl und der Art der Parameter, die Sie üblicherweise wie in Listing 5.4 notieren. Bitte beachten Sie auch den Alternativinhalt.

```
<div>
  <object type="application/x-shockwave-flash"
    data="../images/example.swf" width="400" height="300">
    <param name="movie" value="../images/example.swf" />
    <param name="quality" value="high" />
    <param name="bgcolor" value="#FFFFFF" />
    <p>Leider kann Ihr Browser die Diashow nicht anzeigen.
      Womöglich benötigen Sie einen aktuellen <a
      href="http://www.macromedia.com/de/downloads/">
      Flash Player</a>.</p>
  </object>
</div>
```

Listing 5.4: Flash einbinden

Einbettung von Java-Applets

DEFINITION

Java *ist eine objektorientierte Programmiersprache, die unter anderem von Patrick Naughton, Mike Sheridan und James Gosling in den Jahren 1991 bis 1992 im Auftrag des amerikanischen Computerherstellers* Sun Microsystems *entworfen wurde. Das ursprüngliche Ziel bei der Entwicklung von Java war nicht nur die Entwicklung einer weiteren Programmiersprache, sondern die Entwicklung einer vollständigen Betriebssystemumgebung, inklusive virtueller CPU, für unterschiedlichste Einsatzzwecke. Java wird noch heute weiterentwickelt.*

Man unterscheidet

- Java-Anwendungen, die von einer virtuellen Maschine als eigenständige Programme ausgeführt werden,

- Java-Servlets, die auf einem Webserver ausgeführt werden, und
- Java-Applets, die in einem Webbrowser laufen.

Java-Applets werden entweder über das missbilligte (nicht in der strikten Variante von XHTML enthaltene) Element `applet` eingebunden oder über das Element `object`.

Einbettung mittels applet

DEFINITION

*Das **Element applet** gestattet es Webautoren, ein Java-Applet in ein XHTML-Dokument einzubetten. Der Inhalt des `applet`-Elements dient als Alternative für Benutzerprogramme, die dieses Element nicht unterstützen oder so konfiguriert sind, dass sie Java-Applets nicht ausführen.*

```
<div>
  <applet name="Newsticker" code="NewsTicker.class"
    width="400" height="50">
    <param name="bgcolor" value="000000" />
    <param name="fgcolor" value="ffffff" />
    <param name="speed" value="5" />
    <param name="text0" value="www.intermitto.net" />
    <param name="url0"
      value="http://www.intermitto.net/" />
    <param name="text1" value="Apache Tutorial" />
    <param name="url1"
      value="http://www.intermitto.net/apache/" />
    <param name="text2" value="PHP Tutorial" />
    <param name="url2"
      value="http://www.intermitto.net/php/" />
    <param name="text3" value="und vieles mehr ..." />
    <param name="url3"
      value="http://www.intermitto.net/" />
    <p>Leider kann Ihr Browser das Java-Applet nicht
      anzeigen.</p>
  </applet>
</div>
```

Listing 5.5: Das Element applet[4]

4 Das in Listing 5.5 und Listing 5.6 verwendete Applet wurde von Jens Becker programmiert. Es zeigt, wie ein Applet realisiert werden kann, das seine Anzeige zeitgesteuert ändert. Das Applet ist auf der Seite »Ein Thread-gesteuertes Java-Applet« (*http://www.intermitto.net/loesungen/artikel/appletthread/*) dokumentiert.

DEFINITION

*Das **Attribut code** gibt den Namen der auszuführenden Java-Klassendatei an, die normalerweise die Dateinamenserweiterung `.class` hat. Falls sich diese Datei in einem anderen Verzeichnis befindet als das XHTML-Dokument, das das Applet einbettet, können Webautoren dieses Verzeichnis mittels des Attributs `codebase` über einen URI referenzieren.*

DEFINITION

*Die **Attribute width und height** legen die Breite und Höhe des Applets in Pixel oder Prozent fest. Beides sind Pflichtattribute, müssen also gesetzt werden.*

DEFINITION

*Das **Attribut name** gibt einen Namen für die Instanz des Applets an, um zu ermöglichen, dass verschiedene Applets innerhalb desselben Dokuments miteinander kommunizieren können oder per JavaScript auf das Applet zugegriffen werden kann.*

Innerhalb des `applet`-Elements können Sie beliebig viele Parameter notieren, die dem Applet beim Aufruf übergeben werden. Welche Parameter das Applet erwartet, sollten Sie der Dokumentation des Applets – falls es eine solche gibt – entnehmen können.

Einbettung mittels object

Das Element `applet` wurde zugunsten der Einbindung mittels des Elements `object` für missbilligt erklärt. Wenn Sie die strikte Variante von HTML oder XHTML wählen, müssen Sie auf die Einbindung über `object` zurückgreifen. Sie ähnelt der herkömmlichen Einbindung stark und unterscheidet sich von ihr nur in einigen Attributen.

```
<div>
  <object classid="java:NewsTicker.class"
    codetype="application/java-vm" width="400" height="50">
    <param name="bgcolor" value="000000" />
    <param name="fgcolor" value="ffffff" />
    <param name="speed" value="5" />
    <param name="text0" value="www.intermitto.net" />
    <param name="url0"
      value="http://www.intermitto.net/" />
    <param name="text1" value="Apache Tutorial" />
    <param name="url1"
      value="http://www.intermitto.net/apache/" />
    <param name="text2" value="PHP Tutorial" />
    <param name="url2"
      value="http://www.intermitto.net/php/" />
    <param name="text3" value="und vieles mehr ..." />
```

```
      <param name="url3"
        value="http://www.intermitto.net/" />
      <p>Leider kann Ihr Browser das Java-Applet nicht
        anzeigen.</p>
    </object>
  </div>
```

Listing 5.6: Java-Applets einbinden

DEFINITION

*Das **Attribut classid** gibt den Namen der auszuführenden Java-Klassendatei in einer speziellen Form an: Der festen Zeichenkette `java:` folgt der Klassenname samt Dateinamenserweiterung. Falls sich diese Datei in einem anderen Verzeichnis befindet als das XHTML-Dokument, das das Java-Applet einbettet, können Webautoren dieses Verzeichnis mittels des Attributs `codebase` über einen URI referenzieren.*

DEFINITION

*Auch bei der Einbettung von Applets gibt das **Attribut codetype** den MIME-Typ der einzubettenden Ressource an. Bei Java-Applets ist dies entweder `application/java` oder `application/java-vm`.*

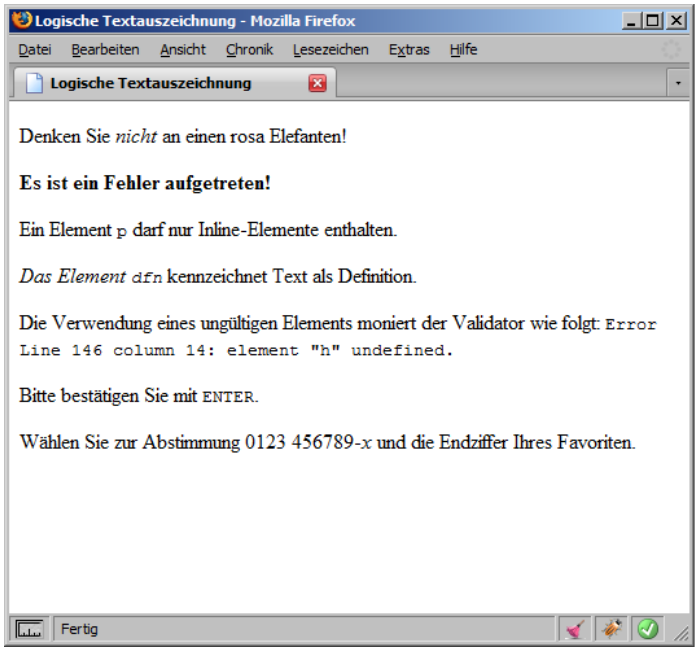
Die Übergabe der Parameter erfolgt analog zur Einbettung über das `applet`-Element.

5.5 Auszeichnung im Text

XHTML enthält sowohl Elementtypen zur logischen Auszeichnung von (inzeiligem) Text als auch Elementtypen zur physischen Textauszeichnung. Logische Auszeichnung versieht Inhalte mit einer bestimmten *Semantik*, während die physische Textauszeichnung eine Aussage über die *Darstellung* der Elemente trifft.

5.5.1 Logische Textauszeichnung: Die Elementtypen em, strong, dfn, code, samp, kbd und var

Elemente der logischen Textauszeichnung versehen Teile eines (inzeiligen) Texts mit einer inhaltlichen Bedeutung. Neben weiteren, die an anderen Stellen dieser Einführung erläutert werden, gehören die folgenden zu den Elementen der logischen Textauszeichnung. Abbildung 5.5 zeigt die Standarddarstellung in Firefox 3 und in Safari 3.



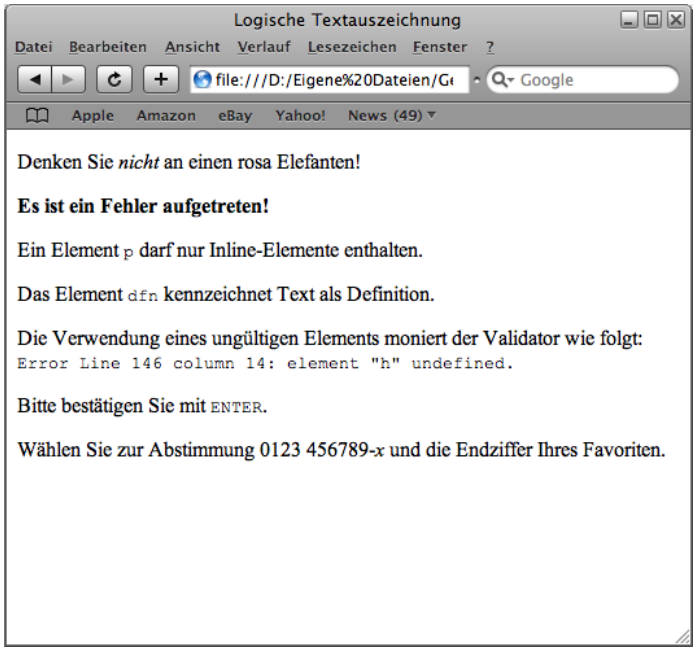


Abbildung 5.5: Firefox 3 und Safari 3 stellen die Elemente `em`, `strong`, `code`, `dfn`, `samp`, `kbd` und `var` ähnlich dar

DEFINITION

*Das **Element em** kennzeichnet Text als betont.*

```
<p>Denken Sie <em>nicht</em> an einen rosa Elefanten!</p>
```

DEFINITION

*Das **Element strong** kennzeichnet Text als stark betont.*

```
<p><strong>Es ist ein Fehler aufgetreten!</strong></p>
```

Da `strong`-Elemente in Browsern im Allgemeinen fett und `em`-Elemente kursiv dargestellt werden, glauben viele Webautoren, `strong` und `em` seien die besseren `b` und `i` (siehe Kapitel 5.5.4). Dieser Gedanke ist nicht von der Hand zu weisen, schließlich wird ein Text tatsächlich dadurch betont, dass er fett oder kursiv formatiert wird, jedoch bleibt diese Betonung rein visuell. Andere Arten von Benutzerprogrammen, beispielsweise Screenreader oder Suchmaschinen, bemerken von der Betonung nichts – anders als bei der semantischen Auszeichnung mit `strong` oder `em`.

DEFINITION

*Das **Element code** kennzeichnet Text als Quelltext.*

```
<p>Ein Element <code>p</code> darf nur Inline-Elemente enthalten.</p>
```

DEFINITION

*Das **Element dfn** kennzeichnet Text als Definition.*

```
<p><dfn>Das Element <code>dfn</code></dfn> kennzeichnet Text als
Definition.</p>
```

Beachten Sie, dass nicht die Definition, sondern der zu definierende Term ausgezeichnet wird.

DEFINITION

*Das **Element samp** kennzeichnet Text als Ausgabebeispiel unter anderem von Programmen oder Scripts.*

```
<p>Die Verwendung eines ungültigen Elements moniert der Validator wie
folgt: <samp>Error Line 146 column 14: element "h" undefined.</
samp></p>
```

DEFINITION

*Das **Element kbd** kennzeichnet Text als Benutzereingabe.*

```
<p>Bitte bestätigen Sie mit <kbd>ENTER</kbd>.</p>
```

DEFINITION

*Das **Element var** kennzeichnet Text als Variable oder Programmargument.*

```
<p>Wählen Sie zur Abstimmung 0123 456789-<var>x</var> und die
Endziffer Ihres Favoriten.</p>
```

5.5.2 Kennzeichnen von Dokumentänderungen: Die Elementtypen del und ins

DEFINITION

*Die **Elemente del** und `ins` kennzeichnen Text als nicht mehr gültig beziehungsweise als neu hinzugefügt.*

```
<p>Der Verein hat <del>42</del> <ins>68</ins> Mitglieder.</p>
```

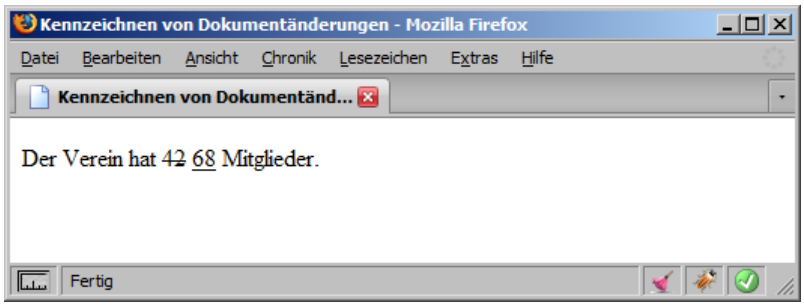


Abbildung 5.6: Alle Browser stellen `del` durch- und `ins` unterstrichen dar

Über die Attribute `cite` und `datetime` können Webautoren nähere Angaben zum Grund der Änderung machen.

DEFINITION

*Das **Attribut cite** dient der Angabe einer Ressource mittels eines URI, die als Grund für die Änderung gelten kann – beispielsweise eine W3C-Empfehlung oder eine Norm.*

```
<p>Bei sehr umfangreichen Dokumenten kann es sinnvoll sein,
Querverweise innerhalb des Dokuments zu setzen. In <acronym>HTML</
acronym> können Webautoren dies über ein Element <code>a</code> mit
dem Attribut <code>name</code> realisieren. <ins cite="http://
www.w3.org/TR/xhtml1/#h-4.10">In <acronym>XHTML</acronym> ist dieses
Attribut für Anker missbilligt.</ins></p>
```

Darstellung

Wenn Sie sich das Beispiel in einem Browser anschauen, werden Sie zwei Dinge feststellen (vgl. Abbildung 5.7):

- Ihr Browser stellt den hinzugefügten Text mit hoher Wahrscheinlichkeit unterstrichen dar. Dies ist sehr unglücklich, schließlich gilt im Web die Regel, dass es sich bei unterstrichenen Texten um Links handelt. Sie sollten die Unterstreichung daher per CSS durch Fettdruck ersetzen.
- Ganz gleich, auf welche Weise der Textabschnitt sich visuell vom Rest des Absatzes unterscheidet, es ist auf den ersten Blick nicht zu erkennen, *weshalb* er sich davon unterscheidet. Der Verweis auf die Quelle ist entweder überhaupt nicht oder nur über Umwege zugänglich, etwa über ein Eigenschaftenmenü.

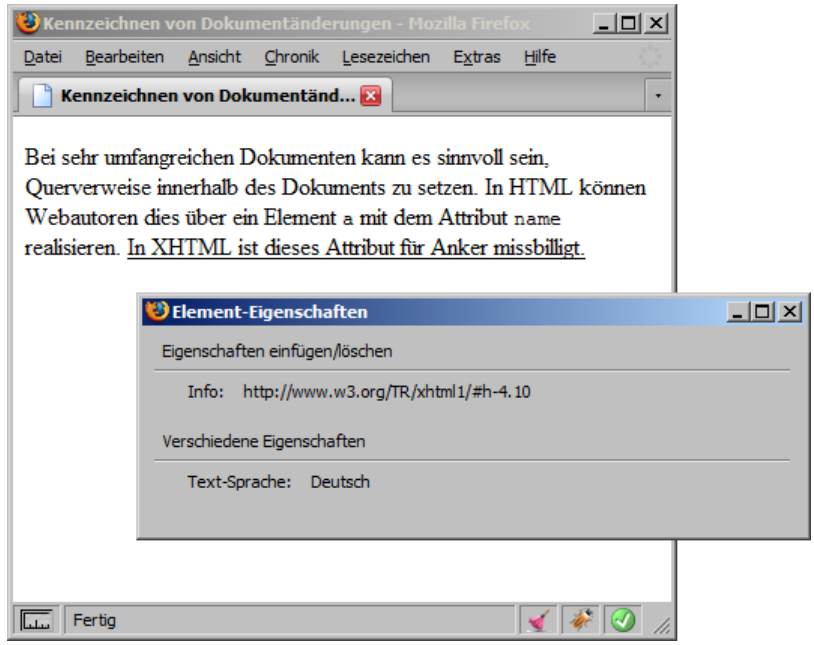


Abbildung 5.7: Der Wert des `cite`-Attributs ist in Firefox über das Eigenschaftenfenster zugänglich

Vor allem aus dem zweiten Grund wird das Element `ins` im Web nur zusammen mit `del` verwendet, zudem häufig so, dass auch am Kontext ersichtlich ist, dass es sich um eine Änderung handelt.

DEFINITION

*Das **Attribut datetime** spezifiziert den Zeitpunkt der Änderung als ISO-Datumsformat.*

Datumsformat

Das Datumsformat hat die Form `YYYY-MM-DDThh:mm:ssTZD`. `YYYY` ist eine vierstellige Jahresangabe, `MM` eine zweistellige Monatsangabe (01 bis 12), `DD` eine zweistellige Tagesangabe (01 bis 31). Der Buchstabe `T` trennt Datums- und Zeitangabe voneinander. `hh` gibt die Stunde an (00 bis 23), `mm` die Minute (00 bis 59), `ss` die Sekunde (00 bis 59) – jeweils zweistellig. `TZD` steht für die Angabe der Weltzeit UTC. Über eine Angabe in der Form `+hh:mm` oder `-hh:mm` geben Sie die Differenz zwischen Weltzeit und lokaler Zeit an. Der Wert `2006-03-12T15:22+01:00` steht beispielsweise für den 12. März 2006, 15 Uhr 22 nach UTC-Zeit plus eine Stunde, also 16:22 Uhr MEZ.

Uhrzeit und Zeitzonenangabe werden häufig weggelassen.

```
<p>Der Verein hat <del datetime="2005-01-01">42</del> <ins
datetime="2006-01-01">68</ins> Mitglieder.</p>
```

Auch der Wert von `datetime` ist entweder überhaupt nicht oder nur über ein Eigenschaftenmenü zugänglich. Daher kann es sinnvoll sein, den Grund der Änderung zusätzlich genauer (und menschenlesbar) über das Attribut `title` zu beschreiben.

```
<p>Der Verein hat <del datetime="2005-01-01" title="Stand: Januar
2005">42</del> <ins datetime="2006-01-01" title="Stand: Januar
2006">68</ins> Mitglieder.</p>
```

Abbildung 5.8: Im Vergleich zu Abbildung 5.6 ist der Zeitpunkt der Änderung nun zugänglich, aber wer kommt schon auf die Idee, sich den Tooltipp anzeigen zu lassen?

5.5.3 Abkürzungen und Akronyme: Die Elementtypen abbr und acronym

Westliche Sprachen machen intensiven Gebrauch von Abkürzungen und Akronymen. Andere Sprachen verwenden analoge Abkürzungsmechanismen, beispielsweise Chinesisch und Japanisch, in denen ein langer Name durch einen Teil der Han-Zeichen des vollständigen Namens repräsentiert wird.

DEFINITION

*Das **Element abbr** zeichnet eine Abkürzung aus. Eine Abkürzung ist die verkürzte Form eines Worts oder einer Wortgruppe wie »z. B.«, »usw.« oder »Abk.«.*

DEFINITION

*Das **Element acronym** zeichnet ein Akronym aus. Ein Akronym ist ein Sonderfall der Abkürzung. Laut Duden handelt es sich um ein Kunstwort, das aus den Anfangsbuchstaben mehrerer Wörter zusammengesetzt ist, beispielsweise »EDV«, »WWW« oder »CDU«.*

Erklärung

Laut Bedingung 4.2 der BITV müssen Abkürzungen und Akronyme bei ihrem ersten Auftreten kenntlich gemacht und erläutert werden[5]. Diese Erläuterung erfolgt über das Attribut `title`:

```
<p>Webautoren benötigen Kenntnisse in vielen Sprachen, <abbr
title="zum Beispiel">z.B.</abbr> in <acronym lang="en"
xml:lang="en" title="">HTML</acronym> und <acronym lang="en"
xml:lang="en" title="Cascading Style Sheets">CSS</acronym>.
<acronym>HTML</acronym> ist noch immer die populärste Sprache des
Webs, wird aber zunehmend von <acronym lang="en" xml:lang="en"
title="Extensible Hypertext Markup Language">XHTML</acronym>
verdrängt.</p>
```

Listing 5.7: Die Elemente `abbr` und `acronym`

Anwendung

Die Auszeichnung und Erläuterung von Abkürzungen und Akronymen ist sinnvoll, da Webautoren nicht davon ausgehen können, dass alle Nutzer deren Bedeutung kennen, entweder weil die Abkürzung nicht allgemein bekannt oder weil dem Nutzer die deutsche Sprache oder das Thema des Texts nicht so geläufig ist. Darüber hinaus liefert die Auszeichnung dieser Konstrukte nützliche Informationen für Werkzeuge wie Rechtschreibkontrollen, Übersetzungssysteme und Suchroboter. Nutzer von Screenreadern können veranlassen, dass Abkürzungen und Akronyme durch deren Erläuterung ersetzt werden, also nicht die abgekürzte, sondern die ausformulierte Form vorgelesen wird.

5 siehe auch ***[Hellbusch 2005]***

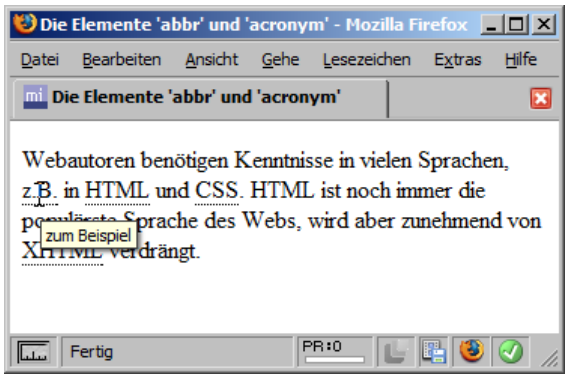


Abbildung 5.9: Erklärung der Abkürzung als Tooltipp

5.5.4 Physische Textauszeichnung: Die Elementtypen i, b, tt, big und small

Elemente der physischen Textauszeichnung versehen Teile eines (inzeiligen) Texts mit einer visuellen Formatierung. Sie treffen keine Aussage über die Semantik des enthaltenen Texts und gehören daher zu den Elementen, auf die Sie wenn möglich im Sinne der Trennung von Markup und Layout zugunsten von CSS verzichten sollten. Folgende Elemente sind auch in der strikten XHTML-Variante enthalten.

DEFINITION

*Das **Element i** formatiert einen Text kursiv.*

```
<p>Ein <i>Text in kursiver Schrift</i>.</p>
```

DEFINITION

*Das **Element b** formatiert einen Text fett.*

```
<p>Ein <b>Text in fetter Schrift</b>.</p>
```

Die Verwendung der Elemente `i` und `b` erübrigt sich häufig aus einem der beiden folgenden Gründe:

`b` und `i` nur sparsam verwenden

- Textpassagen sind aus einem bestimmten visuellen Kontext heraus fett, der häufig über weitere Formatierungsmerkmale verfügt. Denken Sie beispielsweise an ein Navigationsmenü, das in Fettschrift gesetzt ist, aber auch noch über weitere Layouteigenschaften verfügt, beispielsweise Rahmen oder

bestimmte Hintergrundeffekte. CSS ist dabei – im Sinne der konsequenten Trennung von Markup und Layout – die bessere Alternative.

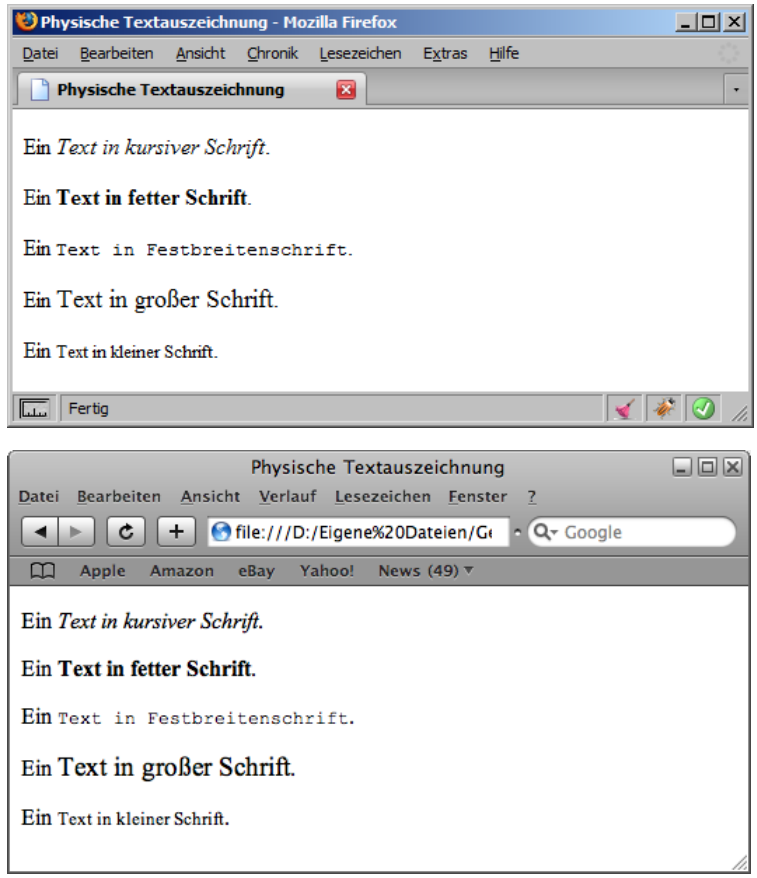


Abbildung 5.10: Darstellung der Elemente `i`, `b`, `tt`, `big` und `small` in Firefox 3 und in Safari 3

- Textpassagen sollen betont werden. Dafür bietet XHTML die deutlich besser geeigneten Elemente der logischen Textauszeichnung `em` und `strong` (siehe Kapitel 5.5.1).

DEFINITION

*Das **Element tt** formatiert einen Text in Festbreitenschrift (Schreibmaschinenschrift).*

```
<p>Ein <tt>Text in Festbreitenschrift</tt>.</p>
```

DEFINITION

*Das **Element big** formatiert einen Text in größerer Schrift.*

```
<p>Ein <big>Text in großer Schrift</big>.</p>
```

DEFINITION

*Das **Element small** formatiert einen Text in kleinerer Schrift.*

```
<p>Ein <small>Text in kleiner Schrift</small>.</p>
```

Es gibt Webautoren, die im Element `small` das Gegenstück zu `strong` sehen. Dieser Eindruck kann allerdings höchstens in visuellen Benutzerprogrammen entstehen; es gibt keinen Screenreader, der mit `small` ausgezeichnete Textpassagen leiser spricht oder gar flüstert.

5.6 Listen

Mengentheoretische Konzepte

In vielen naturwissenschaftlichen Modellen findet man drei grundlegende mengentheoretische Konzepte immer wieder:

- Eine Menge ist eine Sammlung verschiedener Elemente, in der es keine Reihenfolge gibt.
- Ein Tupel ist eine geordnete Zusammenstellung von Elementen, wobei jedes Element mehrfach vorkommen kann.
- Eine Relation ist eine Menge von Tupeln.

Wenn man danach sucht, findet man diese Konzepte auch in XHTML wieder, und zwar in

- ungeordneten Listen,
- geordneten Listen und
- Definitionslisten.

DEFINITION

***Listen** sind eine Darstellungsform für Daten in waagerechter oder (bevorzugt) senkrechter Form. Listen werden zur Organisation von Informationen verwendet.*

5.6.1 Ungeordnete Listen: Die Elementtypen ul und li

DEFINITION

*Das **Element ul** zeichnet eine ungeordnete Liste aus, das heißt, deren **Listenpunkte** `li` unterliegen keiner bestimmten Ordnung, sind also gleichwertig.*

```
<p>Wenn man danach sucht, findet man diese Konzepte auch
  in <acronym>XHTML</acronym> wieder, und zwar in</p>
<ul>
  <li>ungeordneten Listen,</li>
  <li>geordneten Listen und</li>
  <li>Definitionslisten.</li>
</ul>
```

Listing 5.8: Das Element `ul`

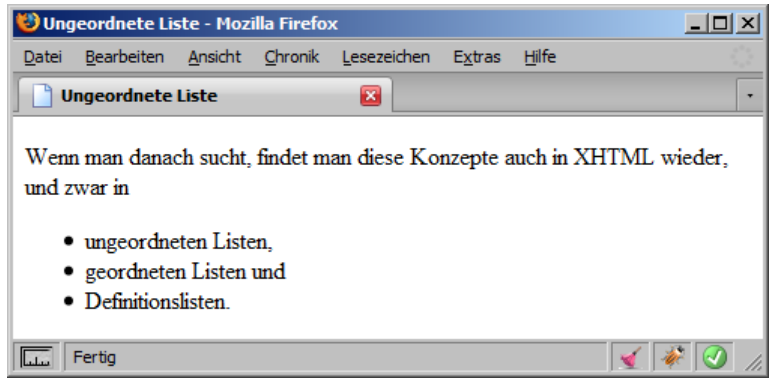


Abbildung 5.11: Ungeordnete Liste

5.6.2 Geordnete Listen: Die Elementtypen ol und li

DEFINITION

*Das **Element ol** zeichnet eine geordnete Liste aus, deren **Listenpunkte li** einer bestimmten Reihenfolge unterliegen.*

Listing 5.9 zeigt die Top 10 der 500 besten Alben aller Zeiten, ermittelt in einer Umfrage unter den Lesern des »Rolling Stone«-Magazins im Jahr 2003. Wie die meisten Listen kann auch diese Top-10-Liste als Tabelle ausgezeichnet werden, wie in Kapitel 5.7 in Listing 5.13 zu sehen ist.

```
<ol>
  <li>Sgt. Pepper's Lonely Hearts Club Band, The Beatles</li>
  <li>Pet Sounds, The Beach Boys</li>
  <li>Revolver, The Beatles</li>
  <li>Highway 61 Revisited, Bob Dylan</li>
  <li>Rubber Soul, The Beatles</li>
  <li>What's Going On, Marvin Gaye</li>
  <li>Exile on Main Street, The Rolling Stones</li>
  <li>London Calling, The Clash</li>
  <li>Blonde on Blonde, Bob Dylan</li>
  <li>The Beatles (»The White Album«), The Beatles</li>
</ol>
```

Listing 5.9: Das Element ol

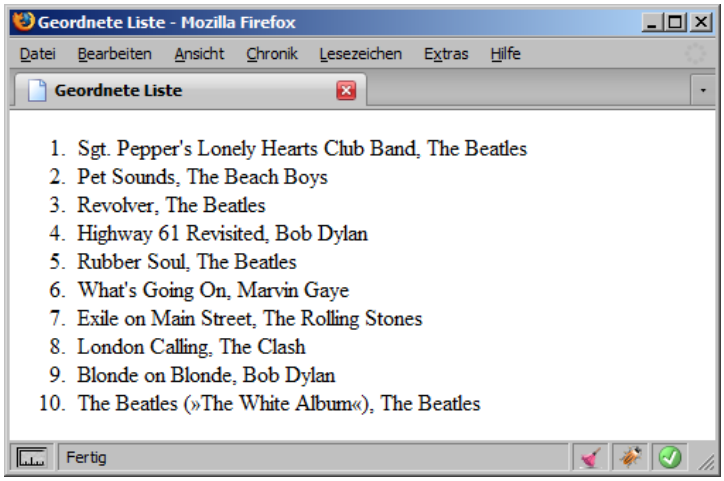


Abbildung 5.12: Geordnete Liste

Ariadne-Faden

Ein weiteres Beispiel für eine geordnete Liste ist ein **Ariadne-Faden**. Der griechischen Mythologie zufolge ist der Ariadne-Faden ein Geschenk der Prinzessin Ariadne an Theseus. Mithilfe des Fadens fand Theseus den Weg durch das Labyrinth, in dem sich der Minotaurus befand. Im Webdesign bezeichnet der Ariadne-Faden – auch Brotkrumen-Navigation genannt – eine bestimmte Art der Navigation, die die Hierarchie widerspiegelt und dem Benutzer so eine zusätzliche Orientierungshilfe bietet. Viele Webautoren halten eine ungeordnete Liste für die beste Auszeichnung. Listing 5.10 zeigt einen Ariadne-Faden, dessen Darstellung, wie in Abbildung 5.13 zu sehen, per CSS so angepasst wurde, dass sie kaum noch an eine Liste erinnert.

```
<ol>
  <li><a href="/">Startseite</a></li>
  <li><a href="/wsdev/">Webentwicklung</a></li>
  <li><a href="/wsdev/einfuehrung/">Einführung in
    <acronym>XHTML</acronym>, <acronym>CSS</acronym> und
    Webdesign</a></li>
  <li><a href="/wsdev/einfuehrung/xhtml/">
    <acronym>XHTML</acronym></a></li>
  <li title="Hier befinden Sie sich gerade.">Listen</li>
</ol>
```

Listing 5.10: Ariadne-Faden

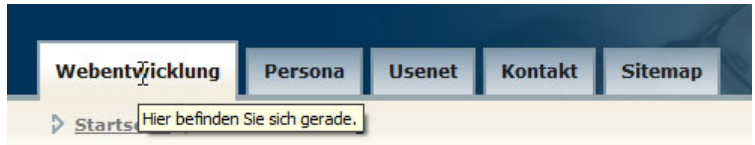


Abbildung 5.13: Ariadne-Faden

Verschachtelung

Anfänger haben häufig Schwierigkeiten, Listen korrekt ineinander zu verschachteln. Behalten Sie stets im Auge, dass die Elemente `ul` und `ol` ausschließlich Elemente vom Typ `li` als Kind beinhalten dürfen. Untergeordnete Listen müssen Sie daher korrekt innerhalb der einzelnen Listenpunkte aufführen, wie Listing 5.11 zeigt.

```
<ul>
  <li>Ebene eins, Listenpunkt eins</li>
  <li>Ebene eins, Listenpunkt zwei
    <ul>
      <li>Ebene zwei, Listenpunkt eins
        <ol>
          <li>Ebene drei, Listenpunkt eins</li>
          <li>Ebene drei, Listenpunkt zwei</li>
          <li>Ebene drei, Listenpunkt drei</li>
        </ol>
      </li>
      <li>Ebene zwei, Listenpunkt zwei</li>
      <li>Ebene zwei, Listenpunkt drei</li>
    </ul>
  </li>
  <li>Ebene eins, Listenpunkt drei</li>
</ul>
```

Listing 5.11: Verschachtelte Listen

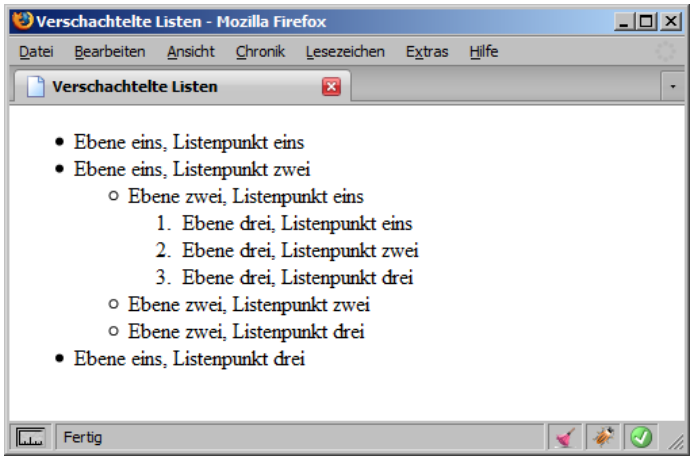


Abbildung 5.14: Browser visualisieren verschachtelte Listen standardmäßig durch Einrückung und unterschiedliche Listenelementmarkierungen

5.6.3 Definitionslisten: Die Elementtypen dl, dt und dd

DEFINITION

*Das **Element** `dl` zeichnet eine Definitionsliste aus. Diese unterscheidet sich von anderen Listenarten nur insoweit, als die Listeneinträge aus zwei Teilen bestehen: einem Definitionsterm und einer Beschreibung. Der Definitionsterm wird durch das **`dt`-Element** zugewiesen und ist auf Inline-Elemente beschränkt. Die Beschreibung wird mit dem **Element** `dd` angegeben und kann Blockelemente enthalten. Ein Listeneintrag kann mehrere `dd`-Elemente umfassen.*

Anwendung

Der Begriff Definitionsliste sollte Sie nicht davon abhalten, Definitionslisten überall dort einzusetzen, wo Paare aus jeweils einem Begriff oder Sachverhalt und mindestens einer Beschreibung aufgeführt werden. Ein klassisches Beispiel ist ein Glossar oder ein Literaturverzeichnis; die HTML-Empfehlung führt gar einen Dialog als Anwendungsbeispiel auf.

```
<dl class="literatur">
  <dt id="Balzert-2004">Balzert 2004</dt>
  <dd>H. Balzert; »Webdesign &amp; Web-Ergonomie«, Dortmund,
    W3L, 2004.</dd>

  <dt id="Behme-Mintert-2000">Behme, Mintert 2000</dt>
  <dd>H. Behme, S. Mintert; <acronym>»XML</acronym> in der
    Praxis &#8211; Professionelles Web-Publishing mit der
    Extensible Markup Language«, 2. Auflage, München, Addison-
    Wesley Verlag, 2000.</dd>
```

```
    <dt id="Berners-Lee-1999">Berners-Lee 1999</dt>
    <dd>T. Berners-Lee; »Der Web-Report &#8211; Der Schöpfer des
      World Wide Webs über das grenzenlose Potential des
      Internets«, München, Verlagshaus Goethestraße
      <acronym>GmbH</acronym> &amp; Co. <acronym>KG</acronym>,
      1999.</dd>

    <!-- Weitere Einträge -->
  </dl>
```

Listing 5.12: Literaturverzeichnis als Definitionsliste

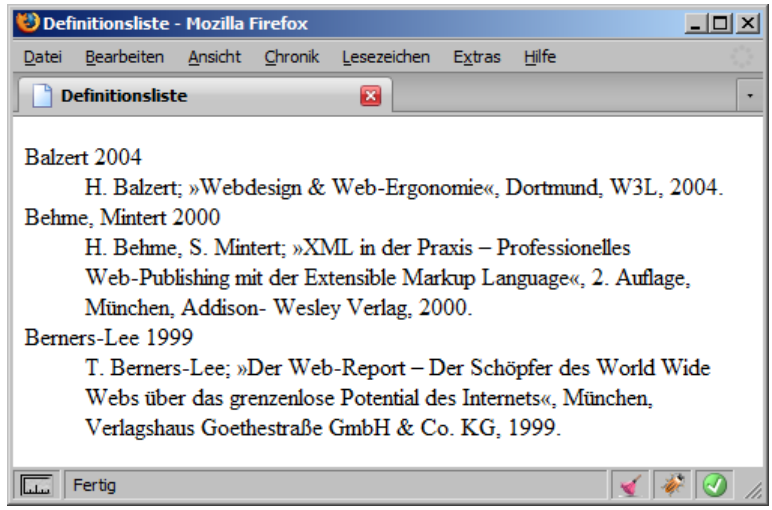


Abbildung 5.15: Die Beschreibungen rücken Browser ein wenig ein, ansonsten sind Definitionslisten standardmäßig eher unübersichtlich – CSS kann da Abhilfe schaffen

5.7 Tabellen

DEFINITION

Tabellen *sind geordnet angezeigte Daten, oft als eine Anzahl von Listen in parallelen Spalten oder rechtwinkligen Anordnungen, die sich nach einer bestimmten Regel aufeinander beziehen. Tabellen eignen sich dazu, Informationen in (visuell) sinnvollen Untermengen zu organisieren.*

Das Tabellenmodell von XHTML gestattet Webautoren die Anordnung von Daten – Text, vorformatiertem Text, Bildern, Links, Formularen, Formularfeldern oder anderen Tabellen – in Zellen, die in Zeilen und Spalten angeordnet sind.

5.7.1 Tabellen, Zeilen, Kopf- und Datenzellen: Die Elementtypen table, tr, th und td

DEFINITION

*Das **Element table** bildet den Container für die Auszeichnung von Tabellen.*

DEFINITION

*Das **Element tr** fungiert als Behälter für eine Zeile, die mehrere Zellen enthält.*

DEFINITION

*Tabellenzellen können zwei Informationsarten enthalten: Kopfinformationen und Daten. Diese Unterscheidung hilft Benutzerprogrammen, Kopf- und Datenzellen unterschiedlich darzustellen: Text in Kopfzellen könnte in fett gedruckter Schrift angezeigt oder in einem anderen Tonfall vorgelesen werden. Das **Element** `th` definiert eine Zelle, die Kopfinformationen enthält. Das **Element** `td` definiert eine Zelle, die Daten enthält.*

Beispiel

Wie bereits in Kapitel 5.6 »Listen« erwähnt, lassen sich viele Listen auch als Tabelle darstellen. Listing 5.13 zeigt die Auszeichnung der Top 10 der 500 besten Alben aller Zeiten als einfache Tabelle (vgl. die Auszeichnung als Liste in Listing 5.9).

```
<table>
  <tr>
    <th>Rang</th>
    <th>Titel</th>
    <th>Interpret</th>
  </tr>
  <tr>
    <td>1</td>
    <td>Sgt. Pepper's Lonely Hearts Club Band</td>
    <td>The Beatles</td>
  </tr>
  <tr>
    <td>2</td>
    <td>Pet Sounds</td>
    <td>The Beach Boys</td>
  </tr>
  <tr>
    <td>3</td>
    <td>Revolver</td>
    <td>The Beatles</td>
  </tr>
  <!-- weitere Zeilen -->
</table>
```

Listing 5.13: Top 10 der 500 besten Alben aller Zeiten

5.7.2 Tabellenüberschrift, Zusammenfassung und Abkürzungen: Der Elementtyp caption und die Attribute summary und abbr

DEFINITION

*Das **Element** `caption` erzeugt eine Tabellenüberschrift.*

HINWEIS

Oftmals ist es erwünscht, aus einer Tabellenüberschrift eine Tabellenunterschrift zu machen. Die Ausrichtung bestimmen Sie mit CSS (siehe Kapitel 7.11.7).

BITV

Die BITV fordert in Bedingung 5.5:

> *»Für Tabellen sind unter Verwendung der hierfür vorgesehenen Elemente der genutzten Markup-Sprache Zusammenfassungen bereitzustellen.«*

In Bedingung 5.6 ist zu lesen:

> *»Für Überschriftenzellen sind unter Verwendung der hierfür vorgesehenen Elemente der genutzten Markup-Sprache Abkürzungen bereitzustellen.«*

Erfüllen lassen sich diese Forderungen über die Attribute `summary` für Elemente vom Typ `table` und `abbr` für Elemente vom Typ `th` beziehungsweise `td`.

DEFINITION

*Das **Attribut** `summary` bietet eine Zusammenfassung der Intention und der Struktur einer Tabelle für Benutzerprogramme, die den Inhalt für nicht visuelle Medien wie Sprach- oder Braillebrowser ausgeben.*

Das `summary`-Attribut wird in visuellen Browsern nicht angezeigt, dient Nutzern von Screenreadern jedoch als gute Orientierungshilfe.

DEFINITION

*Das **Attribut** `abbr` kann dazu verwendet werden, eine abgekürzte Form von Zeilen- und Spaltenüberschriften zur Verfügung zu stellen.*

Anwendung

Diese Maßnahme hat zur Folge, dass in Sprachausgaben lediglich beim ersten Lesen der Überschriftenzelle der komplette Text vorgelesen wird, während bei weiteren Bewegungen in der Tabelle nur die Kurzbezeichnung angesagt wird. Verwenden Sie das Attribut `abbr` nur bei sehr langen Überschriftenzellen.

5.7.3 Zeilengruppen: Die Elementtypen thead, tfoot und tbody

DEFINITION

*Tabellenzeilen können mit den **Elementen** `thead`, `tfoot` und `tbody` in einen Tabellenkopf, einen Tabellenfuß und in einen oder mehrere Rumpfbereiche unterteilt werden. Diese Teilung ermöglicht es Benutzerprogrammen, das Scrollen von Tabellenrümpfen unabhängig von Tabellenkopf und -fuß zu unterstützen. Werden lange Tabellen gedruckt, können Informationen des Tabellenkopfs und -fußes auf jeder Seite mit Tabellendaten wiederholt werden. Tabellenkopf und -fuß sollten Informationen über die Tabellenspalten enthalten. Der Tabellenrumpf sollte Zeilen mit Tabellendaten enthalten.*

Reihenfolge

Vermutlich wären Sie nicht darauf gekommen, dass Sie die drei Bereiche in der Reihenfolge `thead`, `tfoot`, `tbody` notieren müssen; der Tabellenfuß steht vor dem Tabellenrumpf, damit Benutzerprogramme den Fuß darstellen können, bevor alle (der eventuell zahlreichen) Datenzeilen empfangen werden.

Listing 5.14 ergänzt die Tabelle aus Listing 5.13 um die entsprechenden Auszeichnungen für Tabellenkopf und -rumpf. Spannend wird das Dokument erst dann, wenn Sie sich die Druckvorschau ansehen oder das Dokument ausdrucken; die Spaltenüberschriften werden auf jeder Seite wiederholt (siehe Abbildung 5.16).

Beispiel

```
<table>
  <caption>Die Top 10 der 500 besten Alben aller
    Zeiten</caption>
  <thead>
    <tr>
      <th>Rang</th>
      <th>Interpret</th>
      <th>Titel</th>
    </tr>
  </thead>
  <tbody>
    <tr>
      <td>1</td>
      <td>Sgt. Pepper's Lonely Hearts Club Band</td>
      <td>The Beatles</td>
    </tr>
    <tr>
      <td>2</td>
      <td>Pet Sounds</td>
      <td>The Beach Boys</td>
    </tr>
    <tr>
```

```
        <td>3</td>
        <td>Revolver</td>
        <td>The Beatles</td>
      </tr>
      <!-- weitere Zeilen -->
    </tbody>
  </table>
```

Listing 5.14: Top 100 der 500 besten Alben aller Zeiten, vervollständigt

Abbildung 5.16: Wiederholung der Spaltenüberschriften

5.7.4 Spaltengruppen: Die Elementtypen colgroup und col

Die Darstellung einer Tabelle in visuellen Browsern ergibt sich aus den definierten Zeilen und Spalten (und natürlich aus den CSS-Regeln, die auf eine Tabelle angewendet werden), allerdings haben einige vor allem ältere Browser Schwierigkeiten beim Aufbau einer besonders umfangreichen Tabelle, wenn diese noch nicht vollständig geladen wurde. Die Definition von Spaltengruppen kann die Darstellung beschleunigen.

DEFINITION

*Das **Element** `colgroup` erzeugt eine Spaltengruppe. Für die Angabe der Anzahl der Spalten in der Spaltengruppe gibt es zwei sich gegenseitig ausschließende Möglichkeiten:*

1. *Das `span`-**Attribut** des Elements `colgroup` gibt die Anzahl der Spalten in der Gruppe an.*
2. *Das `colgroup`-**Element** enthält `col`-**Elemente**, von denen jedes eine Spalte der Gruppe repräsentiert. Auch an `col`-Elementen lässt sich ein `span`-Attribut notieren. Es gibt die Anzahl der Spalten an, die das Element umfasst.*

Es ist erlaubt, mehrere Spaltengruppen innerhalb einer Tabelle zu definieren.

DEFINITION

*Das **Attribut** `width` lässt sich sowohl für `colgroup`- als auch für `col`-Elemente notieren. Es gibt die Standardbreite für jede Spalte der aktuellen Spaltengruppe (`colgroup`) an beziehungsweise die Standardbreite für jede Spalte, über die sich das aktuelle `col`-Element erstreckt.*

Das `width`-Attribut kann drei Arten von Werten annehmen:

- Einen Pixelwert (z. B. `width="50"`): eine feste Breitenangabe.
- Einen Prozentwert (z. B. `width="25%"`): eine relative Breitenangabe, die auf dem Prozentsatz des horizontalen Raums basiert, der für die Tabelle zur Verfügung steht. Die Angabe `25%` bedeutet folglich ein Viertel des verfügbaren Platzes.
- Einen Proportionalwert (z. B. `width="3*"`): eine relative Breitenangabe der Spalten untereinander. Wenn Benutzerprogramme Spalten Platz zuweisen, weisen sie zuerst Pixel- und Prozentwerte zu, dann verteilen sie den verbleibenden verfügbaren Platz unter den relativen Maßen. Folglich werden, wenn 100 Pixel an Platz verfügbar sind, nachdem das Benutzerprogramm Pixel- und Prozentwerte zugewiesen hat, und die relativen Maße `2*`, `3*` und `5*` sind, der `2*` 20 Pixel, der `3*` 30 Pixel und der `5*` 50 Pixel zugewiesen.

Wie genau das aussehen kann, wird klarer, wenn Sie sich Listing 5.15 bis Listing 5.18 nebst den zugehörigen Abbildungen anschauen. Damit Sie die Darstellung der Tabellen besser vergleichen können, führe ich das Attribut `border` ein und weise der dritten und vierten Tabelle eine Gesamtbreite von 100 Prozent zu.

DEFINITION

*Das **Attribut** `border` gibt die Breite des Rahmens um eine Tabelle und deren Zellen in Pixeln an. Zugunsten entsprechender CSS-Eigenschaften (siehe die Kapitel 7.5.4 bis 7.5.7) können Sie auf dieses Attribut verzichten.*

```
<table border="1">
  <caption>Die Top 3 der 500 besten Alben aller
    Zeiten</caption>
  <colgroup width="50" />
  <colgroup width="250" span="2" />
  <thead>
    <tr>
      <th>Rang</th>
      <th>Titel</th>
      <th>Interpret</th>
    </tr>
  </thead>
  <tbody>
    <tr>
      <td>1</td>
      <td>Sgt. Pepper's Lonely Hearts Club Band</td>
      <td>The Beatles</td>
    </tr>
    <!-- weitere Zeilen -->
  </tbody>
</table>
```

Listing 5.15: Zwei Spaltengruppen mit fester Breite; die beiden letzten Spalten sind in einer Gruppe zusammengefasst

```
<table border="1">
  <caption>Die Top 3 der 500 besten Alben aller
    Zeiten</caption>
  <colgroup>
    <col width="50" />
    <col width="250" />
    <col width="200" />
  </colgroup>
  <thead>
    <tr>
      <th>Rang</th>
      <th>Titel</th>
      <th>Interpret</th>
    </tr>
  </thead>
  <tbody>
    <tr>
      <td>1</td>
      <td>Sgt. Pepper's Lonely Hearts Club Band</td>
      <td>The Beatles</td>
    </tr>
    <!-- weitere Zeilen -->
  </tbody>
</table>
```

Listing 5.16: Eine Spaltengruppe mit drei definierten Spalten; feste Breitenangaben

```
<table border="1" width="100%">
  <caption>Die Top 3 der 500 besten Alben aller
    Zeiten</caption>
  <colgroup>
    <col width="10%" />
    <col width="50%" />
    <col width="200" />
  </colgroup>
  <thead>
    <tr>
      <th>Rang</th>
      <th>Titel</th>
      <th>Interpret</th>
    </tr>
  </thead>
  <tbody>
    <tr>
      <td>1</td>
      <td>Sgt. Pepper's Lonely Hearts Club Band</td>
      <td>The Beatles</td>
    </tr>
    <!-- weitere Zeilen -->
  </tbody>
</table>
```

Listing 5.17: Eine Spaltengruppe mit drei definierten Spalten; Breitenangaben prozentual und fest

```
<table border="1" width="100%">
  <caption>Die Top 3 der 500 besten Alben aller
    Zeiten</caption>
  <colgroup>
    <col width="1*" />
    <col width="5*" />
    <col width="200" />
  </colgroup>
  <thead>
    <tr>
      <th>Rang</th>
      <th>Titel</th>
      <th>Interpret</th>
    </tr>
  </thead>
  <tbody>
    <tr>
      <td>1</td>
      <td>Sgt. Pepper's Lonely Hearts Club Band</td>
      <td>The Beatles</td>
    </tr>
```

```
      <!-- weitere Zeilen -->
    </tbody>
  </table>
```

Listing 5.18: Eine Spaltengruppe mit drei definierten Spalten; Breitenangaben proportional und fest

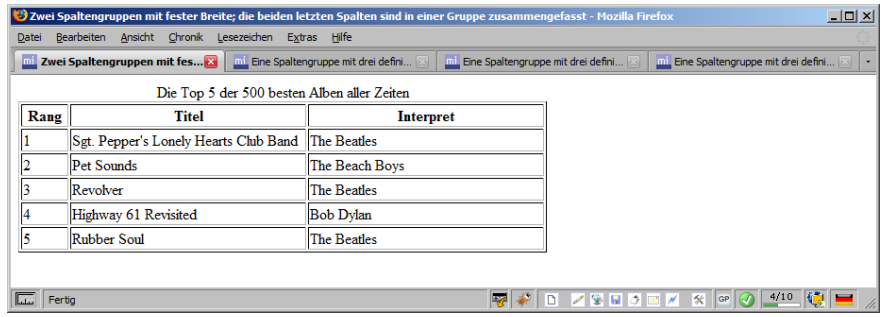


Abbildung 5.17: Darstellung von Listing 5.15 in Firefox 3

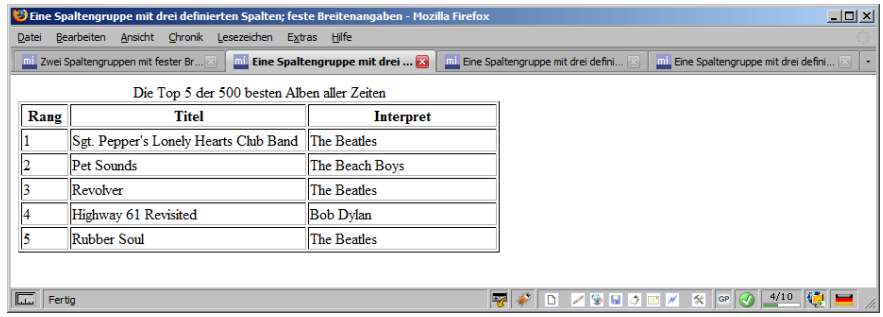


Abbildung 5.18: Darstellung von Listing 5.16 in Firefox 3

Abbildung 5.19: Darstellung von Listing 5.17 in Firefox 3

Abbildung 5.20: Darstellung von Listing 5.18 in Firefox 3

Gerade bei der Darstellung der vierten Tabelle werden Sie in unterschiedlichen Browsern teilweise völlig abweichende Darstellungen vorfinden. Je älter der Browser, desto größer die Abweichung. Daher rate ich von der Verwendung von Proportionalwerten ab. Ziehen Sie stattdessen Prozentwerte vor, entsprechend BITV, Bedingung 3.4: BITV

> *»Es sind relative anstelle von absoluten Einheiten in den Attributwerten der verwendeten Markup-Sprache und den Style-Sheet-Property-Werten zu verwenden.«*

5.7.5 Zellen miteinander verbinden: Die Attribute colspan und rowspan

DEFINITION

*Das **Attribut** `rowspan` gibt die Anzahl der Zeilen an, über die sich die aktuelle Zelle erstreckt, das **Attribut** `colspan` entsprechend die Anzahl der Spalten. Der Standardwert beider Attribute ist 1. Der Wert 0 bedeutet, dass sich die Zelle von der aktuellen Zeile oder Spalte bis zur letzten Zeile oder Spalte des Tabellenabschnitts (Kopf, Fuß oder Rumpf) erstreckt, in dem die Zelle definiert ist.*

```
<table>
  <tr>
    <td>Zelle 1, Zeile 1</td>
    <td colspan="2">Zellen 2 und 3, Zeile 1</td>
  </tr>
  <tr>
    <td>Zelle 1, Zeile 2</td>
    <td>Zelle 2, Zeile 2</td>
    <td>Zelle 3, Zeile 2</td>
  </tr>
</table>
```

Natürlich lassen sich `colspan` und `rowspan` miteinander kombinieren, um eine Zelle über mehrere Zeilen und Spalten auszudehnen.

```
<table>
  <tr>
    <td colspan="2" rowspan="2">Zelle 1 und 2, Zeilen 1
      und 2</td>
    <td>Zelle 3, Zeile 1</td>
  </tr>
  <tr>
    <td>Zelle 3, Zeile 2</td>
  </tr>
  <tr>
    <td>Zelle 1, Zeile 3</td>
    <td>Zelle 2, Zeile 3</td>
    <td>Zelle 3, Zeile 3</td>
  </tr>
</table>
```

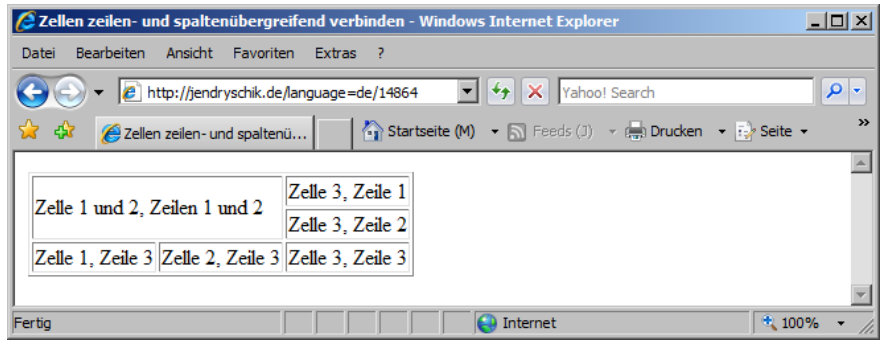


Abbildung 5.21: Zellen zeilen- und spaltenübergreifend verbinden

5.8 Zitate

Grundsätzlich unterscheidet man zwei Arten von Zitaten: wörtliche Zitate und sinngemäße Zitate.

DEFINITION

Wörtliche Zitate *lassen sich in* kurze Zitate *(inzeiliger Inhalt) und* lange Zitate *(Zitatblöcke) einteilen. Wenn es aus dem Zusammenhang nicht klar wird, folgt in beiden Fällen die Angabe der Quelle beziehungsweise des Urhebers, erkennbar vom Rest des Texts abgetrennt, meistens durch Klammern. Wird aus einem bestimmten Werk zitiert, ist es üblich, eine Jahreszahl anzufügen.*

Setzen Sie ein kurzes Zitat in Anführungszeichen, beispielsweise: Man sollte Zitate »nur gebrauchen, wo man fremder Autorität wirklich bedarf« (Schopenhauer). Lange Zitate rücken Sie – abhängig von der zugrunde liegenden Zitierregel mal mit, mal ohne Anführungszeichen – in den laufenden Text um etwa fünf Millimeter (links oder beidseitig) ein.

Anführungszeichen

• • • DEFINITION

Stellen, die aus einem anderen Text zwar nicht wörtlich, aber doch vom Sinn oder Gehalt her wiedergegeben werden, die also nicht Ihre eigenen Gedanken, sondern die eines anderen Urhebers formulieren, nennt man ***sinngemäße Zitate****.*

Sie müssen hier die Quelle ebenfalls angeben. Für gewöhnlich steht der Quellenangabe ein vgl. (vergleiche) voran.

Allgemeingültige Regeln dazu, wie die Quellenangabe genau auszusehen hat, gibt es nicht – zumal diese Angaben schwieriger wurden, seit außer aus Büchern und Zeitschriften auch aus Webseiten zitiert wird, die sich jederzeit ändern können. Von Verlag zu Verlag, Universität zu Universität und Fakultät zu Fakultät gibt es andere Bestimmungen. Vor allem wenn Sie an universitären Texten wie Seminar-, Magister- oder Diplomarbeiten schreiben, sollten Sie sich im Vorfeld genau informieren, welche Zitierweise von Ihnen erwartet wird.

Zitierregeln

5.8.1 Inzeilige Zitate: Der Elementtyp q

• • • DEFINITION

Das ***Element*** *q zeichnet ein kurzes Zitat aus, das keinen Zeilenwechsel erfordert (inzeiliger Inhalt).*

Das oben angeführte kurze Zitat sollten Webautoren in XHTML entsprechend den Empfehlungen eigentlich wie folgt auszeichnen:

```
<p>Ein kurzes Zitat wird in Anführungszeichen gesetzt. Beispiel: Man
sollte Zitate <q>nur gebrauchen, wo man fremder Autorität wirklich
bedarf</q> (<cite>Schopenhauer</cite>).</p>
```

Fragwürdiges Konzept und fehlerhafte Implementierungen

Die HTML 4.01-Spezifikation des W3C legt in Abschnitt 9.2.2 eindeutig fest, dass der Inhalt des `q`-Elements mit begrenzenden Anführungszeichen dargestellt werden muss, und weist zugleich darauf hin, dass Autoren selbst keine Anführungszeichen setzen sollten. Dieses Konzept ist aus zwei Gründen fragwürdig:

Anführungszeichen

- Die Vorschrift einer bestimmten Darstellung hat in einer Auszeichnungssprache nichts zu suchen, vor allem nicht in HTML, das von seiner Rückwärtskompatibilität lebt. Beim Einsatz des `q`-Elements müssen sich Webautoren auf dessen vorgeschriebene Darstellung verlassen, damit die Bedeutung übertragen wird. Was soll ein Browser tun, der `q` nicht implementiert hat? Er weiß nicht, dass er Anführungszeichen setzen muss, tut es auch nicht, und der Benutzer kann den Text folglich nicht als Zitat erkennen. Diesen Fehler hat in dieser Form kein anderes Element.
- Die korrekte Kennzeichnung von Zitaten ist Semantik auf Zeichenebene und gehört damit zu den Aufgaben der Typografie, nicht der Textauszeichnung. Fragen werden ja auch mit einem Fragezeichen gekennzeichnet und nicht wie folgt: `<question>Warum hat Noah die zwei Stechmücken nicht erschlagen</question>`

Browserunterstützung

Abgesehen von den theoretischen Bedenken gibt es derzeit auch keine Implementierung, die man ruhigen Gewissens als akzeptabel bezeichnen könnte. Schauen Sie sich folgendes XHTML-Dokument in einem Browser Ihrer Wahl an:

```
<p><cite>Egon Friedell</cite> illustriert den Hang der Rokoko-Kultur
zur Ausnahme mit dem wunderbaren Beispiel: <q cite="isbn:3-406-02510-
2">Als eine vornehme italienische Dame einmal ein künstliches
Fruchteis verzehrte, sagte sie: <q>Wie schade, dass es keine Sünde
ist.</q></q></p>
```

Viele Browser, darunter der Internet Explorer für Windows, verzichten auf die Darstellung von Anführungszeichen um zitierten Text, sodass Zitate nicht von gewöhnlichem Fließtext zu unterscheiden und dadurch nicht als solche zu identifizieren sind. Die Verwirrung der Leser können Sie sich sicherlich vorstellen, ganz zu schweigen von den rechtlichen Problemen, die eine nicht vorhandene Auszeichnung von Zitaten mit sich bringen kann. Aktuelle Gecko- und Opera-Browser verteilen zwar typografische Anführungszeichen um Zitate, jedoch noch immer nicht die richtigen[6].

Lösungsansätze

Mit CSS ist es möglich, eine richtige Darstellung in einigen Browsern zu erreichen, dies ist jedoch keine Lösung, denn Dokumente müssen auch dann funktionieren, wenn das verwendete Benutzerprogramm kein CSS unterstützt[7].

6 Vgl. Kapitel 2.11.3.

7 Vgl. Bedingung 6.1 der BITV: »Es muss sichergestellt sein, dass mittels Markup-Sprachen geschaffene Dokumente verwendbar sind, wenn die zugeordneten Style Sheets deaktiviert sind.«

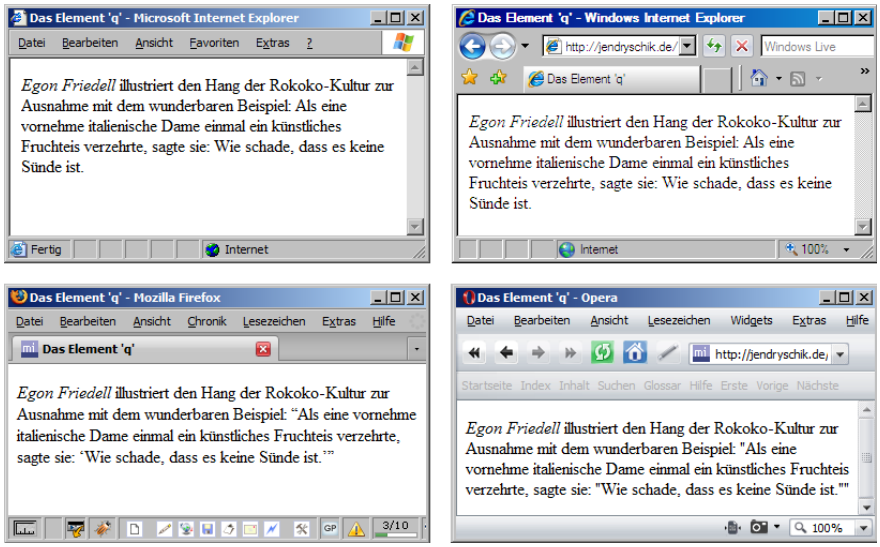


Abbildung 5.22: Darstellung des Elements `q` in IE 6, IE 7, Firefox 3 und Opera 9.5

Ein oft empfohlener Workaround, zitierten Text vom umgebenden Fließtext abzuheben, besteht in der Änderung des Schriftschnitts, zum Beispiel kursiv, oder der Schriftfarbe. Da auch hier keine Lösung ausschließlich über CSS-Formatierungen infrage kommt, könnten Webautoren auf das Element `i` zurückzugreifen:

```
<p>Ein kurzes Zitat wird in Anführungszeichen gesetzt. Beispiel: Man
sollte Zitate <q><i>nur gebrauchen, wo man fremder Autorität
wirklich bedarf</i></q> (<cite>Schopenhauer</cite>).</p>
```

Das Element `i` gehört zu den Elementen der physischen Textauszeichnung (siehe Kapitel 5.5.4) und fügt keinerlei Semantik hinzu, sondern sorgt für die kursive Darstellung seines Textinhalts. Es ist nicht als deprecated (missbilligt) gebrandmarkt, kann also auch in der strikten Variante von XHTML verwendet werden. Es ist fester Bestandteil der Sprache und wird von jedem aktuellen Browser erkannt und interpretiert, womit eine vom Fließtext unterscheidbare Darstellung von Zitaten übergreifend erreicht wäre. Allerdings liegt der Zusammenhang zwischen kursiver Schrift und zitiertem Text nicht gerade auf der Hand (zumal mit `em` ausgezeichneter Text normalerweise auch kursiv dargestellt wird), und zudem erfordern alle Zitierregeln eindeutig Anführungszeichen um zitierten Text. Und wie sollen Webautoren mit verschachtelten Zitaten umgehen? Eine Änderung des Schriftschnitts oder der Farbe ist als Kennzeichnung von Zitaten somit nicht akzeptabel.

q nicht verwenden!

Viele Webautoren halten die Verwendung von `q` für einen handwerklichen Fehler. Auch ich rate Ihnen davon ab. Aus den aufgeführten Gründen ist es unmöglich, dieses Element derzeit zu verwenden. Ich empfehle, Zitate wie gewöhnlichen Fließtext auszuzeichnen und die richtigen Anführungszeichen manuell zu setzen. So umschiffen Sie alle Browser-Bugs und stellen eine den Zitierregeln entsprechende Darstellung sicher. Den Verlust an Markup sollten Sie in diesem Fall verschmerzen.

5.8.2 Zitatblöcke: Der Elementtyp blockquote

DEFINITION

*Das **Element** `blockquote` zeichnet ein langes Zitat aus. Es erzeugt eine eigene Zeile und muss Blockelemente beinhalten.*

Anführungszeichen

Visuelle Benutzerprogramme stellen `blockquote` im Allgemeinen als eingerückten Block dar; der Screenreader JAWS kündigt das Folgende als Zitat an. Es werden keine automatischen Anführungszeichen gesetzt, weil diese in manchen Zitierregeln für Zitatblöcke nicht vorgesehen sind und auch weil es in den düsteren Zeiten vor CSS gängige Praxis war, `blockquote` als Element zur visuellen Einrückung beliebiger Inhalte zu verwenden. Wenn Sie Anführungszeichen setzen wollen oder müssen, sollten Sie diese auch hier per Hand einfügen. Darüber hinaus müssen Sie beachten, dass das Zitat nicht direkt im `blockquote`-Element stehen darf, sondern erst andere Blockelemente wie etwa `p` eingefügt werden müssen.

Listing 5.19 enthält einen Zitatblock, der wiederum zwei inzeilige Zitate enthält. Es demonstriert ebenfalls die Verwendung des Attributs und des Elements `cite`, die ich anschließend im nächsten Abschnitt erläutere.

```
<p>Buchautor und -kritiker <cite>Dieter Wunderlich</cite>
  urteilt über Ondaatjes Roman »Der englische Patient« wie
  folgt:</p>
<blockquote
  cite="http://www.dieterwunderlich.de/
    Ondaatje_patient.htm#com">
  <p>»›Manche der Geschichten, die der Mann ruhig in das
    Zimmer hinein erzählt, gleiten wie Falken von Schicht
    zu Schicht.‹ Das prägt den gesamten Roman. Der in Sri
    Lanka geborene holländische Schriftsteller Michael
    Ondaatje erzählt teilweise authentisch vom Schicksal
    des Wüstenforschers Graf Ladislaus (auch: László)
    Almásy (1895-1951). [...]</p>
  <p>›Dieser Roman trägt eine dreifache Krone: er ist
    tiefgründig, schön und lässt das Herz schneller
```

```
      schlagen<, urteilte die Nobelpreisträgerin <cite>Toni
      Morrison</cite>.«</p>
  </blockquote>
```

Listing 5.19: Die Elemente `blockquote` und `cite`

5.8.3 Referenz auf die Quelle eines Zitats: Der Elementtyp und das Attribut cite

DEFINITION

*Das **Element** `cite` dient der Auszeichnung von Referenzen auf die Quelle beziehungsweise den Urheber eines Zitats.*

Für gewöhnlich stellen Browser den Elementinhalt kursiv dar. In wissenschaftlichen Schriften, etwa in Bibliografien, werden Autorennamen oft in Kapitälchen gesetzt (siehe die CSS-Eigenschaft `text-transform`, Abschnitt 7.4.6).

DEFINITION

*Das **Attribut** `cite` wird dazu verwendet, eine Quelle direkt über ihren URI zu adressieren. Das kann auch ein URN sein, der die ISBN eines Buchs angibt, aus dem zitiert wird. Webautoren können das Attribut `cite` sowohl beim Element `q` als auch beim Element `blockquote` verwenden.*

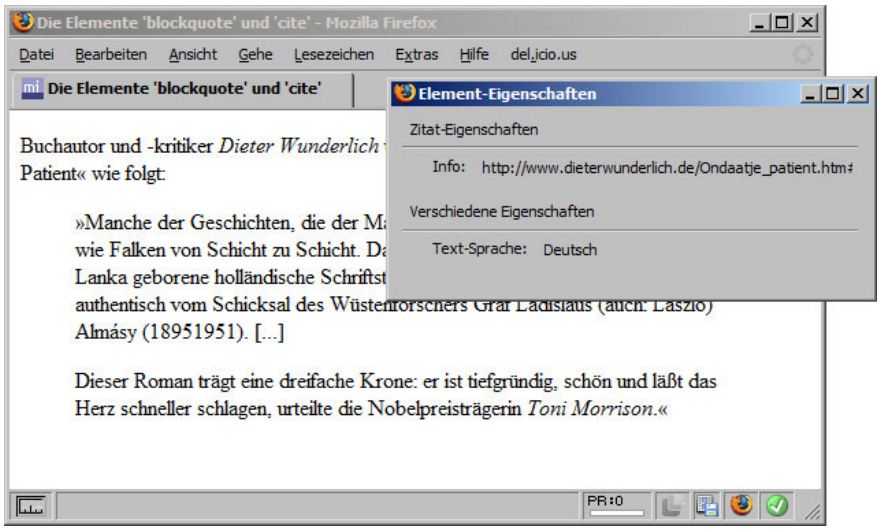


Abbildung 5.23: Firefox zeigt den Wert des `cite`-Attributs in den Elementeigenschaften an

Browserunterstützung

Leider ist auch die Unterstützung des `cite`-Attributs mangelhaft. Lediglich iCab und Mozilla zeigen die Referenz entweder direkt im Kontextmenü oder im Unterpunkt EIGENSCHAFTEN an. Mit CSS oder externen Tools lassen sich die im `cite`-Attribut angeführten Verweise zugänglich machen, allerdings nicht für alle Browser. Daher gilt auch hier: Wenn Sie auf die Anzeige der Zitatquelle angewiesen sind, müssen Sie diese direkt im Text anführen.

5.9 Struktur und Gruppierung

5.9.1 Gruppierung: Die Elementtypen div und span

DEFINITION

*Die **Elemente** `div` und `span` ermöglichen es, ein Dokument zusätzlich zu strukturieren. Mit dem Blockelement `div` können Webautoren Elemente zusammenfassen, mit dem Inline-Element `span` können sie (textuelle) Elementinhalte besonders kennzeichnen.*

Anwendung

Beide Elemente enthalten keinerlei Semantik, das heißt, sie sind lediglich Container ohne inhaltliche Bedeutung und beeinflussen auch nicht die Darstellung eines Elements. Sie sollten daher nur dann verwendet werden, wenn kein anderes, geeigneteres Element existiert.

Zusammen mit den Attributen `id` und `class` (siehe Kapitel 4.6.1) sind beide Elemente gut dafür geeignet,

- eine Art semantischer Feinstrukturierung vorzunehmen, wenngleich diese im Allgemeinen nicht oder zumindest nicht standardisiert maschinenlesbar ist, und
- entsprechend ausgezeichnete Bereiche speziell zu formatieren, vorzugsweise per CSS über ID- beziehungsweise Klassenselektoren.

Beispiel

Listing 5.20 zeigt ein XHTML-Dokument, das auf Datenbankinformationen beruht und eine DVD-Sammlung präsentiert. Da es in XHTML keine Elemente für DVD-Titel, Erscheinungsjahr oder Darsteller gibt, könnte eine Lösung wie folgt aussehen:

```
<div class="dvd" id="citizen-kane">
  <h2>Citizen Kane</h2>
  <table>
    <tr>
      <th>Originaltitel:</th>
      <td>Citizen Kane</td>
    </tr>
    <tr>
```

```
        <th>Produktionsjahr:</th>
        <td>1941</td>
      </tr>
      <tr>
        <th>Darsteller:</th>
        <td><span class="hauptdarsteller">Orson
          Welles</span>, Joseph Cotten, Agnes Moorehead</td>
      </tr>
    </table>
  </div>

  <div class="dvd" id="sieben">
    <h2>Sieben</h2>
    <table>
      <tr>
        <th>Originaltitel:</th>
        <td>Seven</td>
      </tr>
      <tr>
        <th>Produktionsjahr:</th>
        <td>1998</td>
      </tr>
      <tr>
        <th>Darsteller:</th>
        <td><span class="hauptdarsteller">Brad Pitt, Morgan
          Freeman</span>, Gwyneth Paltrow, Kevin Spacey</td>
      </tr>
    </table>
  </div>
```

Listing 5.20: Die Elemente `div` und `span`

Gerade `div`-Elemente sind hervorragend dazu geeignet, ein Dokument »vorzustrukturieren« und so einen Rahmen für die eigentliche Auszeichnung der Inhalte aufzubauen. Wichtig ist hierbei, keine reine `div`- und `span`-Wüste zu erzeugen – die Gefahr besteht durchaus. Eine Handvoll dieser Elemente an »strategisch« günstigen Stellen kann die innere Struktur eines Dokuments erheblich verbessern und bietet zusätzliche Steuerungsmöglichkeiten für die Gestaltung und die logische Trennung von Abschnitten. Ein `div`-Element sollte jedoch niemals dort eingesetzt werden, wo ein `p` oder ein `h3` sinnvoll wäre, ein `span` niemals dort, wo Sie das Element `label` verwenden sollten. Für `div`-Elemente gilt im Grunde dasselbe, was die Webentwicklerin Steffi Abel vor einiger Zeit in der Newsgroup *de.comm.infosystems.www.authoring.misc* über verschachtelte Tabellen geschrieben hat:

> *»›Der Mann, der die Brücke, die über den Fluss, der im Winter Hochwasser, das ganz schlammig ist, führt, führt, baute, lebt nicht mehr.‹ ist eine valide (grammatisch, orthografisch und syntaktisch korrekt) formulierte Aussage. Dennoch ist es unsinnig, solche Sätze zu formulieren.«*

CSS Zen Garden

Das Projekt *CSS Zen Garden*[8] von Dave Shea hat diese Vorgehensweise zum Äußersten getrieben und zeigt zugleich, wie man es macht und wie man es nicht macht. Dave Shea und Molly Holzschlag beschreiben in ihrem Buch »Zen und die Kunst des CSS-Designs« ***[Shea, Holschlag 2005]***, was das Projekt seit seinem Start Ende 2002 vermittelt:

> *»Gäbe es nur eine zentrale Sammlung großartiger CSS-Designarbeiten! Das Vorhandene lediglich zu sammeln würde nicht funktionieren, denn es gab nicht viel. Der Schlüssel zur Förderung dieser Idee bestand darin, Designer zu neuen, hervorragenden CSS-Arbeiten zu ermutigen.«*

Und an anderer Stelle:

> *»Der Hauptzweck des Zen-Gartens besteht darin, die Möglichkeiten standardbasierten Designs zu veranschaulichen, und die verschiedenen Designs zeigen eindrucksvoll, welche Flexibilität CSS bietet.«*

Die größtmögliche Flexibilität wurde leider durch eine Vielzahl an `div`- und `span`-Elementen erkauft, die es ermöglichen, viele zusätzliche Ansatzpunkte für die Gestaltung der Seite bereitzustellen, die ohne diese Forderung allerdings nicht notwendig wären. Keine Webseite sollte vollkommen inhaltsleere Auszeichnungselemente wie

```
<div id="extraDiv1"><span></span></div>
<div id="extraDiv2"><span></span></div>
<div id="extraDiv3"><span></span></div>
<div id="extraDiv4"><span></span></div>
<div id="extraDiv5"><span></span></div>
<div id="extraDiv6"><span></span></div>
```

oder nichtssagende Klassennamen wie `p1`, `p2`, `p3` im Überfluss enthalten.

5.10 Formulare

DEFINITION

*Ein **Formular** ist ein standardisiertes Mittel zur Erfassung von Daten. Formulare vereinfachen die Erhebung von (Massen-)Daten, sorgen für Vollständigkeit sowie Datenintegrität und schützen vor Mehrdeutigkeiten, die bei freier Wortwahl oder einem formlosen Antrag auftreten können.*

8 *http://www.csszengarden.com/*

DEFINITION

*Ein **XHTML-Formular** ist ein Abschnitt eines Dokuments, der neben gewöhnlichem Inhalt spezielle Steuerelemente – beispielsweise Optionsfelder, Schaltflächen, (Mehrfach-) Auswahllisten oder Kontrollkästchen – und deren Beschriftungen enthält.*

Formulare sind ein Instrument der Mensch-Maschine-Interaktion, das heißt eine Schnittstelle zwischen einem Nutzer, der das Formular ausfüllt, und einem Programm oder Script, das die Angaben entgegennimmt und auswertet oder verarbeitet. Benutzer füllen ein Formular im Allgemeinen durch die Bedienung seiner Steuerelemente aus und senden es anschließend ab, zum Beispiel zu einem Webserver oder einem Mailserver. Im Web häufig anzutreffende Formulartypen sind Anmelde-, Bestell-, Kontakt- oder Suchformulare.

Mensch-Maschine-Interaktion

Das Auslesen und Verarbeiten der Formulardaten kann XHTML allein jedoch nicht leisten. Hier kommen server- und clientseitige Programmier- oder Skriptsprachen zum Einsatz.

Verarbeitung

HINWEIS

*Auch wenn Zugänglichkeit und **Gebrauchstauglichkeit** wichtige Teile der Webentwicklung sind (gerade bei Formularen, bei denen es darauf ankommt, dass sie verständlich, übersichtlich und auch per Tastatur vollständig bedienbar sind), wird in dieser Einführung nur auf die technischen Grundlagen eingegangen. Web-ergonomische Forderungen, Eingabehilfen, programmatische Verarbeitungsabläufe und Fehlerbehandlung würden den Rahmen dieser Einführung sprengen. Stattdessen empfehle ich die Lektüre der Artikel **[Caspers 2007]** und **[Hellbusch]**.*

5.10.1 Formularcontainer: Der Elementtyp form

DEFINITION

*Das **Element** `form` ist ein Container für Steuerelemente. Alles, was zwischen Start- und End-Tag steht, gehört zum Formular. In strikten Dokumenten darf das `form`-Element nur Blockelemente als Kinder enthalten.*

`form`-Elemente verfügen üblicherweise über die Attribute `action`, `method` und `enctype`.

Attribute

DEFINITION

*Das **Attribut** `action` gibt in Form eines URI an, wohin die Formulardaten geschickt werden sollen, wenn das Formular abgesendet wird.*

DEFINITION

*Das **Attribut** `method` gibt die HTTP-Methode zur Übertragung der Formulardaten an. Die Methode `get` ist die Standardeinstellung und legt fest, dass die Daten als Parameter an die im Attribut `action` angegebene Adresse angehängt werden. Bei der Methode `post` werden die Daten über den sogenannten Standardeingabekanal zur Verfügung gestellt.*

Die W3C-Empfehlungen legen nahe, `post` immer dann zu verwenden, wenn das auswertende Programm »Nebeneffekte« verursacht, die Daten also über das auswertende Programm hinaus weiterverwendet werden. Dies ist zum Beispiel dann der Fall, wenn eine Datenbank verändert werden soll. Wird das Formular lediglich zum Informationsabruf verwendet, beispielsweise bei einer Suchabfrage, reicht `get` aus.

DEFINITION

*Das **Attribut** `enctype` gibt den MIME-Typ für die Übertragung des Formulars zum Server an, sofern als Übertragungsmethode `post` gewählt ist. Der Standardwert ist `application/x-www-form-urlencoded`; ermöglicht das Formular Dateiuploads, sollte `multipart/form-data` verwendet werden.*

```
<form action="http://www.example.org/kontakt">
  <!-- Auszeichnungs- und Steuerelemente -->
</form>
<form action="http://www.example.org/bildupload"
  method="post" enctype="multipart/form-data">
  <!-- Auszeichnungs- und Steuerelemente -->
</form>
```

5.10.2 Steuerelemente

Ein Formular besteht aus speziellen Elementen, genannt Steuerelemente, die eine Interaktion eines Nutzers mit einer Webseite über einfache Hyperlinks hinaus ermöglichen. Jedes Steuerelement hat einen über das Attribut `name` definierten Namen und einen Anfangswert, der seitens des Nutzers oder durch ein Script verändert werden kann. Wird ein Formular abgesendet, fassen einige Steuerelemente ihre Namen mit den aktuellen Werten zu einem Paar zusammen. Diese Paare werden mit dem HTTP-Request übertragen.

Der Elementtyp input

DEFINITION

*Das **leere Element** `input` definiert unterschiedliche Steuerelemente. Das Attribut `type` entscheidet, um welche Art von Steuerelement es sich konkret handelt.*

Unter anderem folgende Werte kann das Attribut `type` annehmen:

type="text"

Texteingabefeld

Erzeugt ein einzeiliges Texteingabefeld.

```
<input type="text" name="text" />
```

Anfangsbreite und Längenbeschränkung

Das Attribut `size` gibt dem Browser die Anfangsbreite des Texteingabefelds an. Der Wert bezieht sich auf die Anzahl der Zeichen. Das Attribut `maxlength` begrenzt die Zahl der Zeichen, die ein Nutzer eingeben kann. Eine derartige Einschränkung kann beispielsweise bei der Erfassung deutscher Postleitzahlen sinnvoll sein oder zur Eingrenzung auf einen festgelegten Wertebereich. `size` und `maxlength` können ungleich sein.

```
<input type="text" name="plz" size="5" maxlength="5" />
```

Vorbelegung

Oftmals ist es eine gute Idee, Formularfelder mit bestimmten Werten vorzubelegen. Das kann hinsichtlich der Gebrauchstauglichkeit das Mittel der Wahl sein, um das Ausfüllen des Formulars zu vereinfachen. Das Attribut `value` weist einem Steuerelement einen Anfangswert zu. Bei Texteingabefeldern hat es jedoch den Nachteil, dass Nutzer die Vorbelegung manuell löschen müssen, wenn sie einen anderen Wert eintragen wollen. Über ein einfaches JavaScript lässt sich dies automatisieren. Fokussiert (**Fokus**) der Nutzer das Eingabefeld, etwa indem er hineinklickt, wird die Vorbelegung entfernt; lässt er es leer und entfernt den Fokus aus dem Eingabefeld, wird die Vorbelegung wieder eingefügt.

```
<input type="text" name="suche" value="Vorbelegung"
  onblur="if(this.value == '') this.value='Vorbelegung'"
  onfocus="if(this.value == 'Vorbelegung') this.value=''" />
```

type="password"

Passwortfeld

Erzeugt ein einzeiliges Texteingabefeld, dessen Zeichen im Gegensatz zu Standardtexteingabefeldern versteckt werden, in den meisten Browsern in Form von Asterisken (`*`).

```
<input type="password" name="password" />
```

Auch Passwortfelder erlauben Längenbeschränkungen und eine Vorbelegung.

```
<input type="password" name="password"
  size="15" maxlength="32" value="Vorbelegung"
  onblur="if(this.value == '') this.value='Vorbelegung'"
  onfocus="if(this.value == 'Vorbelegung') this.value=''" />
```

Vorbelegung

Abbildung 5.24: Texteingabefeld mit Vorbelegung

Kontrollkästchen

type="checkbox"

Mit `type="checkbox"` wird ein Kontrollkästchen erzeugt. Kontrollkästchen sind Ein-/Ausschalter, von denen der Benutzer keinen, einen oder mehrere ankreuzen kann. Das Attribut `value` ist notwendig und gibt den Wert des Kontrollkästchens an, der mit dem Absenden des Formulars an das verarbeitende Programm übertragen wird.

```
<label><input type="checkbox" name="contact" value="phone" />
  Ich freue mich auf den Anruf eines
  Vertriebsmitarbeiters.</label><br />
<label><input type="checkbox" name="contact" value="mail" />
  Bitte senden Sie mir zukünftig weitere Informationen per
  E-Mail.</label>
```

Vorselektierung

Mithilfe des Attributs `checked` lassen sich Kontrollkästchen vorselektieren.

```
<label><input type="checkbox" name="contact" value="phone"
  checked="checked" /> Ich freue mich auf den Anruf eines
  Vertriebsmitarbeiters.</label><br />
<label><input type="checkbox" name="contact" value="mail" />
  Bitte senden Sie mir zukünftig weitere Informationen per
  E-Mail.</label>
```

☑ Ich freue mich auf den Anruf eines Vertriebsmitarbeiters.
☐ Bitte senden Sie mir zukünftig weitere Informationen per E-Mail.

Abbildung 5.25: Kontrollkästchen

Optionsfeld

type="radio"

`type="radio"` erzeugt ein Optionsfeld. Optionsfelder sind wie Kontrollkästchen Ein-/Ausschalter, schließen sich jedoch gegenseitig aus, wenn sie sich den gleichen Namen teilen. Autoren sollten sicherstellen, dass ein Optionsfeld als aktiv vorbelegt ist.

```
Anrede:
<label><input type="radio" name="anrede" value="herr"
  checked="checked" /> Herr</label>
<label><input type="radio" name="anrede" value="frau" />
  Frau</label>
```

Anrede: Herr Frau

Abbildung 5.26: Optionsfelder

type="submit"

Absenden-Schaltfläche

`type="submit"` erzeugt eine einfache Absenden-Schaltfläche. Aktiviert man sie, wird das Formular abgeschickt. Benutzerprogramme verwenden den Wert des Attributs `value` als Beschriftung für die Schaltfläche.

```
<input type="submit" value="Absenden" />
```

Absenden

Abbildung 5.27: Absenden-Schaltfläche

type="reset"

Zurücksetzen-Schaltfläche

`type="reset"` erzeugt eine Schaltfläche zum Zurücksetzen des Formulars in seinen Anfangszustand. Benutzerprogramme verwenden den Wert des Attributs `value` als Beschriftung für die Schaltfläche.

```
<input type="reset" value="Abbrechen" />
```

Zurücksetzen-Schaltflächen sehen genau so aus wie Absenden-Schaltflächen, nur deren Beschriftung unterscheidet sich. Daher geschieht es recht häufig, dass Nutzer nach dem Ausfüllen eines Formulars versehentlich auf die Zurücksetzen- anstatt auf die Absenden-Schaltfläche klicken. Um dies zu vermeiden, verzichten viele Webautoren auf Zurücksetzen-Schaltflächen. Ein Nachteil für Nutzer ergibt sich daraus nicht. Oder wie oft hatten Sie bereits das Bedürfnis, Eingaben, die Sie in einem Formular vorgenommen haben, aktiv zu verwerfen?

type="hidden"

Verstecktes Steuerelement

`type="hidden"` erzeugt ein verstecktes Steuerelement, das nicht dargestellt wird, dessen Wert aber mit dem Formular übertragen wird. Webautoren verwenden versteckte Felder im Allgemeinen, um Informationen von Client/Server-Operationen zu speichern, die ansonsten dadurch, dass HTTP ein zustandsloses Protokoll ist, verloren gingen.

```
<input type="hidden" name="parameter" value="wert" />
```

Dateiauswahl

type="file"

`type="file"` erzeugt ein Dateiauswahl-Steuerelement, das Benutzern die Auswahl einer Datei gestattet, sodass ihr Inhalt mit dem Formular übertragen werden kann. Benutzerprogramme können den Wert des `value`-Attributs als Anfangdateinamen verwenden.

```
Datei hochladen: <input type="file" name="upload" />
```

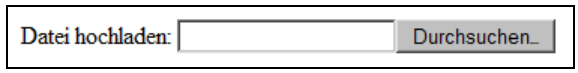


Abbildung 5.28: Dateiauswahl-Steuerelement

Auswahlmenüs: Der Elementtyp select

DEFINITION

*Das **Element** `select` erzeugt in Verbindung mit `option`-**Elementen** ein Menü, das Benutzern Alternativen anbietet, aus denen sie wählen können. Normalerweise wird beim Absenden des Formulars der Elementinhalt des ausgewählten Listeneintrags übertragen. Webautoren können jedoch über das Attribut `value` bestimmen, dass ein anderer Wert übertragen wird. Einträge lassen sich über die Angabe `selected="selected"` vorwählen.*

```
<select name="lacrimosa">
  <option>Angst (CD)</option>
  <option>Einsamkeit (CD)</option>
  <option>Alles Lüge (CDS)</option>
  <option>Satura (CD)</option>
  <option>Schakal (CDS)</option>
  <option>Inferno (CD)</option>
  <option>Stolzes Herz (CDS)</option>
  <option>Stille (CD)</option>
  <option>Live (DCD)</option>
  <option>Allein zu zweit (CDS)</option>
  <option selected="selected">Elodia (CD)</option>
  <option>Der Morgen danach (CDS)</option>
  <option>Fassade (CD)</option>
  <option>Durch Nacht und Flut (CDS)</option>
  <option>Echos (CD)</option>
  <option>Lichtgestalt (CD)</option>
  <option>Lichtgestalten (EP)</option>
  <option>Lichtjahre (DCD)</option>
</select>
```

Abbildung 5.29: Auswahlmenü mit Vorselektierung

Sichtbare Einträge

Das Attribut `size` gibt an, wie viele Einträge beziehungsweise Zeilen eines Auswahlmenüs sichtbar sind. Wird `size` nicht angegeben oder ist sein Wert gleich `1`, stellen Browser das Auswahlmenü im Allgemeinen als Klappliste dar.

Mehrfachselektion

Über die Angabe `multiple="multiple"` geben Sie an, dass auch mehrere Einträge selektiert werden können. Dann sollten Sie die Anzahl der Zeilen, die angezeigt werden sollen, angeben, da Browser sich in diesem Fall unterschiedlich verhalten. Einige Browser zeigen einen Teil der Liste an, andere klappen sie vollständig auf. Für Nutzer ist eine Mehrfachauswahl im Allgemeinen nicht unmittelbar erkennbar, auch wissen viele nicht, wie sie mehrere Einträge selektieren können (`Strg` beziehungsweise `Befehlstaste` bei gleichzeitigem Selektieren der gewünschten Einträge). Ein kleiner Hinweis kann da nicht schaden, beispielsweise in Form eines `title`-Attributs.

```
<select name="lacrimosa" multiple="multiple" size="10"
  title="Mehrfachauswahl bei gedrückter Strg-Taste möglich">
  <option>Angst (CD)</option>
  <option>Einsamkeit (CD)</option>
  <!-- ... -->
  <option>Lichtgestalten (EP)</option>
  <option>Lichtjahre (DCD)</option>
</select>
```

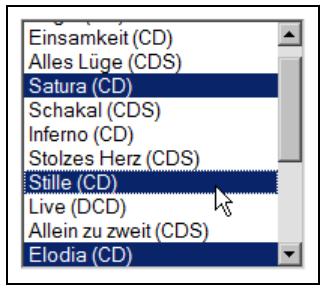


Abbildung 5.30: Auswahlmenü mit Mehrfachselektion

DEFINITION

*Das **Element** `optgroup` gestattet es Autoren, die Auswahlmöglichkeiten logisch zu gruppieren. Dies ist besonders dann hilfreich, wenn der Nutzer aus einer langen Liste wählen muss. Gruppen dürfen nicht verschachtelt werden. Über das Attribut `label` vergeben Sie eine Menübeschriftung für die Gruppe von Einträgen, die im Auswahlmenü als selbst nicht auswählbare Überschrift angezeigt wird.*

```
<select name="lacrimosa" multiple="multiple" size="10">
  <optgroup label="CD">
    <option>Angst</option>
    <option>Einsamkeit</option>
    <option>Satura</option>
    <option>Inferno</option>
    <option>Stille</option>
    <option>Elodia</option>
    <option>Fassade</option>
    <option>Echos</option>
    <option>Lichtgestalt</option>
  </optgroup>
  <optgroup label="CDS">
    <option>Alles Lüge</option>
    <option>Schakal</option>
    <option>Stolzes Herz</option>
    <option>Allein zu zweit</option>
    <option>Der Morgen danach</option>
    <option>Durch Nacht und Flut</option>
  </optgroup>
  <optgroup label="DCD">
    <option>Live</option>
    <option>Lichtjahre (DCD)</option>
  </optgroup>
  <optgroup label="EP">
```

```
      <option>Lichtgestalten</option>
    </optgroup>
  </select>
```

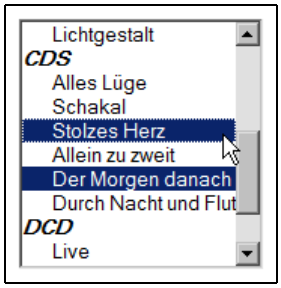


Abbildung 5.31: Mehrfachselektion bei einem gruppierten Auswahlmenü

Mehrzeilige Texteingabefelder: Der Elementtyp textarea

DEFINITION

*Das **Element** `textarea` erzeugt ein mehrzeiliges Texteingabefeld. Benutzerprogramme sollten den Elementinhalt als Anfangswert des Steuerelements verwenden und diesen Text zu Beginn darstellen. Über die Attribute `rows` und `cols` geben Sie die Anzahl der sichtbaren Textzeilen beziehungsweise die sichtbare Breite in durchschnittlicher Zeichenbreite an.*

Darstellung

Zeilen, die länger sind als die sichtbare Breite, werden umbrochen. Darüber hinaus stellen Browser im Allgemeinen Rollbalken dar, falls der Inhalt länger wird als die Anzahl der sichtbaren Textzeilen.

```
<textarea name="statement" cols="40" rows="10">Inhalt</textarea>
```

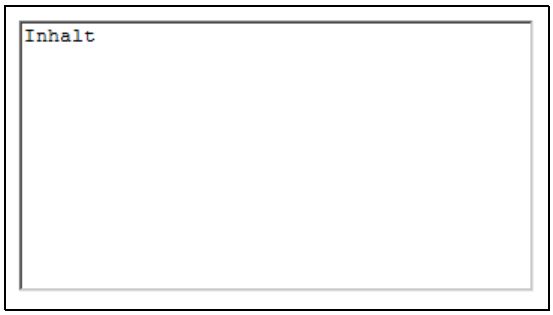


Abbildung 5.32: Mehrzeiliges Texteingabefeld mit Vorbelegung

5.10.3 Beschriftungen: Der Elementtyp label

Einige Steuerelemente haben bereits automatisch eine zugehörige Beschriftung oder erhalten sie mittels eines Attributs `value`, beispielsweise Schaltflächen, während andere keine Beschriftung haben.

DEFINITION

*Das **Element** `label` dient der Beschriftung von Steuerelementen, die keine implizite Beschriftung haben.*

Eine Verwendung des `label`-Elements demonstriert das Beispiel zu Kontrollkästchen auf Seite 256. Dadurch wird ein direkter Bezug zwischen dem Kontrollkästchen und der dazugehörigen Beschriftung »Ich freue mich auf den Anruf eines Vertriebsmitarbeiters.« hergestellt. In vielen Browsern ist es dadurch möglich, das Kontrollkästchen auch durch Klick auf die Beschriftung an- oder abzuwählen. Dient das `label`-Element der Beschriftung eines Eingabefelds, würde dieses bei Klick auf die Beschriftung fokussiert werden. Auf die visuelle Gestaltung eines Formulars hat das `label`-Element standardmäßig keinen direkten Einfluss, wenngleich sich dies mit CSS natürlich ändern ließe.

Bezug zwischen Beschriftung und Steuerelement

Der Bezug zwischen Beschriftung und Steuerelement lässt sich auch dann herstellen, wenn diese im Quelltext durch Markup voneinander getrennt sind, beispielsweise durch eine Layouttabelle zur übersichtlichen Anordnung der Formularelemente. In diesem Fall vergeben Sie dem Steuerelement eine eindeutige ID und referenzieren es über das Attribut `for` im `label`-Element.

```
<table>
  <tr>
    <td><label for="name">Name</label></td>
    <td><input type="text" name="name" id="name" /></td>
  </tr>
  <tr>
    <td><label for="e-mail">E-Mail</label></td>
    <td><input type="text" name="e-mail" id="e-mail" /></td>
  </tr>
  <tr>
    <td><label for="nachricht">Ihre Nachricht</label></td>
    <td><textarea name="nachricht" id="nachricht" cols="30"
      rows="7"></textarea></td>
  </tr>
</table>
```

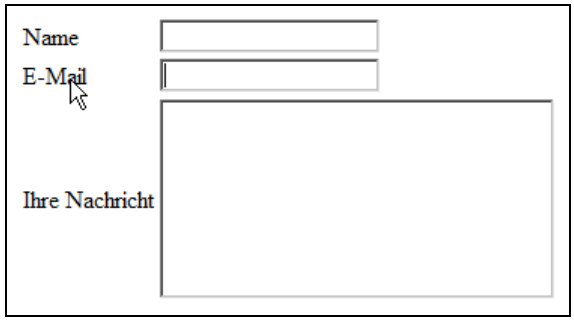


Abbildung 5.33: `label`-Elemente erhöhen die Benutzerfreundlichkeit Ihrer Formulare

5.10.4 Struktur: Die Elementtypen fieldset und legend

DEFINITION

*Das **Element** `fieldset` gestattet es Webautoren, thematisch zusammengehörige Formularinhalte zu gruppieren. Über das **Element** `legend` wird der Gruppe ein Name verliehen, der die Funktion einer Überschrift erfüllen kann.*

Bedingung 12.3 der BITV fordert: BITV

> *»Große Informationsblöcke sind mittels Elementen der verwendeten Markup-Sprache in leichter handhabbare Gruppen zu unterteilen.«*

Eine Gruppierung vereinfacht Nutzern die Orientierung innerhalb eines Formular und dessen Bedienung, vor allem dann, wenn dieses besonders umfangreich ist; die Beschriftung verbessert darüber hinaus die Zugänglichkeit, wenn das Formular nicht visuell ausgegeben wird. Die meisten Browser visualisieren das `fieldset`-Element als Rahmen um die darin enthaltenen Formularelemente.

```
<form action="http://www.example.org/kontakt">
  <fieldset>
    <legend>Persönliche Angaben</legend>
    <!-- Auszeichnungs- und Steuerelemente -->
  </fieldset>
  <!-- Weitere Gruppen -->
</form>
```

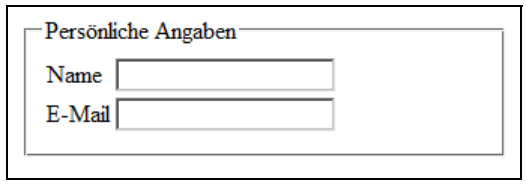


Abbildung 5.34: Gruppierung von Steuerelementen mittels `fieldset`

5.11 Weitere Elemente

In diesem Abschnitt erhalten Sie einen Überblick über weitere interessante XHTML-Elementtypen, die sich nicht in einen der vorherigen Abschnitte einordnen ließen.

5.11.1 Kontaktinformationen: Der Elementtyp address

DEFINITION

*Das **Element** `address` dient Webautoren dazu, Kontaktinformationen für das aktuelle Dokument bereitzustellen.*

Es ist nicht dafür gedacht, generelle Kontaktinformationen auszuzeichnen, also etwa Einträge in einem Adressbuch. `address` ist ein Blockelement und darf nur Inline-Elemente enthalten.

```
<address>Verantwortlich für den Inhalt:
  <a href="mailto:michael@jendryschik.de">Jendryschik</a>
</address>
```

5.11.2 Präformatierter Text: Der Elementtyp pre

DEFINITION

*Das **Element** `pre` zeichnet seinen Inhalt als präformatiert (vorformatiert) aus. Der Browser lässt Leerraumzeichen (in HTML und XHTML die Zeichen Tabulator, Leerzeichen, Wagenrücklauf und Zeilenvorschub) intakt, verhindert einen automatischen Umbruch und stellt Text im Allgemeinen in einer Nichtproportionalschrift (Festbreitenschrift) dar.*

Dieses Verhalten ist vorteilhaft, wenn die flexible Darstellung von XHTML die gewünschte Darstellung Ihres Texts verhindert, zum Beispiel bei Gedichten, ASCII-Zeichnungen oder Quelltextbeispielen.

Programmcode

Beachten Sie, dass Quelltext zusätzlich mit dem Element `code` ausgezeichnet wird (siehe Kapitel 5.5.1). Da das Element `pre` Markup enthalten kann, müssen Sonderzeichen durch die entsprechenden Zeichenreferenzen umschrieben werden.

```
<pre><code>&lt;address>Verantwortlich für den Inhalt:
  &lt;a href="mailto:michael@jendryschik.de">Jendryschik&lt;/a>
&lt;/address></code></pre>
```

5.11.3 Programmcode eines Scripts: Die Elementtypen script und noscript

JavaScript

Ein clientseitiges Script ist ein Programm, das in ein (X)HTML-Dokument eingebettet ist und auf dem Rechner des Nutzers ausgeführt wird, wenn das Dokument geladen oder ein bestimmtes Ereignis ausgelöst wird, etwa wenn der Nutzer auf einen Link klickt oder mit der Maus über einen bestimmten Bereich der Seite fährt. Die am weitesten verbreitete Skriptsprache ist JavaScript.

DEFINITION

JavaScript *ist eine hauptsächlich clientseitig eingesetzte Skriptsprache, die dazu entwickelt wurde, Webseiten dynamisch zu gestalten. JavaScript ist unabhängig vom Betriebssystem, aber nicht vom verwendeten Browser, der das Script interpretiert. Die Grundlage von JavaScript bildet ECMAScript, das als ECMA-262 standardisiert wurde; JavaScript in der Version 1.5 entspricht ECMA-262, Version 3* ***[ECMA-262 1999]****. Aktuell ist Version 1.8. Über das* Document Object Model *(siehe Kapitel 4.3) kann JavaScript auf alle Elemente der Webseite zugreifen und ermöglicht, diese Elemente auch dann zu ändern oder sogar neu zu erzeugen, nachdem die Seite vom Browser geladen und dargestellt wurde.*

Einsatzmöglichkeiten von JavaScript

Mit JavaScript ist es unter anderem möglich,

- eine rudimentäre Plausibilitätsprüfung von Formularen vor dem Absenden vorzunehmen,
- Formularfelder vorzubelegen,
- auf Benutzereingaben mit speziellen Aktionen zu reagieren, zum Beispiel Elemente ein- oder auszublenden oder Grafiken auszutauschen,
- Werte von CSS-Eigenschaften und damit die Darstellung von Elementen auf der Seite zu ändern,
- Daten zu empfangen und zu senden, ohne dass der Browser die Seite neu laden muss (Stichwort: AJAX).

DEFINITION

*Das **Element** `script` bindet ein Script innerhalb eines Dokuments ein. Es darf beliebig oft im `head` und `body` eines Dokuments stehen. Das Script kann im Inhalt des `script`-Elements definiert oder in eine externe Datei ausgelagert werden. Das Attribut `type` gibt in Form eines MIME-Typs an, um welche Skriptsprache es sich handelt.*

Beispiel

Das bewusst sehr einfach gehaltene Listing 5.21 (schließlich lesen Sie gerade eine Einführung in XHTML, nicht in JavaScript) zeigt die Verwendung des `script`-Elements. Es enthält eine Funktion, über die Vordergrund-, Hintergrund- und Rahmenfarbe eines Elements einer bestimmten ID geändert werden können. Das Script wird über das Universalattribut `onmouseover` der im Dokumentkörper aufgeführten Links aufgerufen. Ein `style`-Element (mehr dazu in Kapitel 6.3.2) enthält die notwendigen CSS-Eigenschaften, deren Werte durch das Script verändert werden.

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

<html xmlns="http://www.w3.org/1999/xhtml" lang="de" xml:lang="de">
  <head>
    <meta http-equiv="Content-Type"
      content="text/html; charset=utf-8" />
    <title>Das Element 'script'</title>

    <script type="text/javascript">
      /* <![CDATA[ */
      function changeColors(id, color, backgroundColor,
        borderColor)
      {
        var current = document.getElementById(id);

        if(current)
        {
            current.style.color = color;
            current.style.backgroundColor = backgroundColor;
            current.style.borderColor = borderColor;
            return false;
        }
      }
      /* ]]> */
    </script>

    <style type="text/css">
      /* <![CDATA[ */
      #block {
        color: #5F5E5E;
```

```
        background-color: #E9EEF1;
        border: 1px solid #A9B7C0;
        width: 400px;
        padding: 5px 10px;
        font-weight: bold;
        text-align: center;
      }
      /* ]]> */
    </style>
  </head>

  <body>
    <p>Bitte wählen Sie ein Farbschema:</p>

    <ul class="chooser">
      <li>
        <a href="#"
           onmouseover="return changeColors('block',
             '#1A1A1A', '#EBE4C3', '#9AA7AE');"
        >Erste Farbkombination</a>
      </li>
      <li>
        <a href="#"
           onmouseover="return changeColors('block',
             '#494A8D', '#CCCCFF', '#494A8D');"
        >Zweite Farbkombination</a>
      </li>
      <li>
        <a href="#"
           onmouseover="return changeColors('block',
             '#5F5E5E', '#E9EEF1', '#A9B7C0');"
        >Standardwerte</a>
      </li>
    </ul>

    <p id="block">Lorem ipsum eos ignota fierent officiis at,
      vero rebum adolescens eum te. Id erant fabellas eam, ne
      populo luptatum eleifend est. Pri eu insolens
      persecuti.</p>
  </body>
</html>
```

Listing 5.21: Das Element `script`

Externes Script

Eine bessere Lösung ist in den meisten Fällen, das Script in eine externe Datei auszulagern. Dabei wird der Inhalt des `script`-Elements in ein eigenes Textdokument geschrieben und mit der Dateinamenserweiterung `js` gespeichert. Die

Einbindung erfolgt ebenfalls über das `script`-Element, wobei die Datei im Attribut `src` über einen URI referenziert wird:

```
<script type="text/javascript" src="changeColors.js"></script>
```

Alternative Inhalte

Scripts werden häufig für zentrale Funktionen einer Website eingesetzt, etwa zur Realisierung eines dynamischen Navigationsmenüs (ohne an dieser Stelle diskutieren zu wollen, ob ein solches JavaScript-Menü sinnvoll ist oder nicht). Wenn ein Nutzer diese Website mit einem Browser besucht, der die verwendete Skriptsprache nicht kennt, oder wenn der Nutzer die Scriptunterstützung seines Browsers deaktiviert hat, wird er womöglich ein Problem bekommen: Gibt es keine alternative Navigation, ist die Website für ihn nicht vollständig zugänglich, im schlimmsten Fall überhaupt nicht navigierbar. Die Verwendung des Elements `noscript` kann Abhilfe schaffen.

DEFINITION

*Das **Element** `noscript` erlaubt Autoren, alternativen Inhalt anzubieten, der nur angezeigt wird, wenn ein Script nicht ausführbar ist.*

Beispiel

So könnte Listing 5.21 dahingehend erweitert werden, dass über ein `noscript`-Element eine alternative Navigation bereitgestellt wird, die auf Unterseiten verweist, bei denen das entsprechende Farbschema standardmäßig eingestellt ist. Entwickler sollten dann natürlich darauf achten, dass der JavaScript-Farbwechsler nur dann erscheint, wenn JavaScript aktiviert ist, das heißt diesen also auch per Script ausgeben.

```
<noscript>
  <ul class="chooser">
    <li>
      <a href="farbschema1.html">Erste Farbkombination</a>
    </li>
    <li>
      <a href="farbschema2.html">Zweite Farbkombination</a>
    </li>
  </ul>
</noscript>
```

6 Einführung in CSS

6.1 Design und Styling

Die beiden Bezeichnungen »Design« und »Styling« kennen wir sicherlich alle aus dem täglichen Gebrauch. Intuitiv verwenden wir das richtige Vokabular: Wir kaufen Möbel oder Geräte aus dem Bereich der Unterhaltungselektronik nicht nur deshalb, weil sie hochwertig verarbeitet und funktional sind, sondern auch, weil uns ihr Design gefällt. Jugendliche Discogänger müssen sich erst »aufstylen«, bevor sie auch nur einen Fuß vor die Tür setzen. Stars und Sternchen besorgen sich ihre Abendgarderobe beim Modedesigner, gehen aber zum Stylisten, der Frisur, Make-up und Fingernägel für ihren großen Auftritt auf dem roten Teppich richtig in Szene setzen soll.

Schwerer als ihr intuitiver Gebrauch fällt die exakte Definition der Bezeichnungen Design und Styling. Im Folgenden versuche ich, der genauen Bedeutung beider Bezeichnungen auf den Grund zu gehen.

6.1.1 Design

Die etymologischen Wurzeln des Worts Design liegen in mehreren Begriffen. Das Substantiv »Design« entstammt sowohl dem französischen Wort »dessein« als auch dem italieni-

schen Wort »disegno«, deren Wurzeln in dem lateinischen Verb »designare« (beabsichtigen, bezeichnen) sowie in dem lateinischen Nomen »signum« (Zeichen, Abzeichen, Kennzeichen, Signal, Bild, Siegel) liegen. Ins Deutsche ging der Begriff Anfang des 19. Jahrhunderts über, jedoch in der älteren Form »Dessein«. Seit den 1960ern verwenden wir die englische Form »Design«.

Definitionen

Es gibt eine Vielzahl unterschiedlicher Definitionen des Designbegriffs. Hier eine kleine Auswahl:

> *»Der Begriff Design bezeichnet die Gestaltung von Gegenständen aller Art nach den Kriterien von Funktionalität (zum Beispiel Ergonomie) und Ästhetik. Nicht zuletzt in Hinblick auf die Marktchancen eines Produkts zielt der Designer auf eine möglichst optimale Verschmelzung beider Kategorien. Je nach inhaltlicher Ausrichtung unterscheidet man in Display-(Verpackung), Foto-, Grafik-, Industrie-, Kraftfahrzeug-, Mode-, Produkt-, Textil-, Schmuckdesign etc.« (Designlexikon International[1])*
>
> *»›Design‹ kann ›Entwicklung‹, ›Berechnung‹, ›Konstruktion‹ bzw. deren Ergebnis, ›Entwurf‹, ›Gestaltung‹ oder ›Konzept‹ usw. einschließen und entsprechend benannt werden.« (DIN EN ISO 9001 Qualitätsmanagementsysteme. Modell zur Qualitätssicherung/QM-Darlegung in Design, Entwicklung, Produktion, Montage und Wartung. Ausgabe 1994–08, Seite 4.)*

Die Bezeichnung »Design« verweist also auf einen Vorgang wie auch auf das Ergebnis dieses Vorgangs.

> *»Design bedeutet meist Entwurf oder Formgebung. (...) Im Deutschen ist die Bezeichnung ›Design‹ als Bezeichnung für den Prozess des bewussten Gestaltens vor allem einer Fachszene geläufig. Im Verlauf der jüngeren Designgeschichte wurde [der Begriff] Bestandteil des allgemeinen Sprachgebrauchs. Hier dient er häufig als Sammelbegriff für alle bewusst gestalteten Aspekte eines realen oder virtuellen Objektes, einer Dienstleistung oder Marke. (...) Seine Ergebnisse sind stets vorläufig: andere Designprozesse können verbessern, überholen oder in Frage stellen.« (Wikipedia[2])*

Besonders interessant ist die Definition des *Oxford English Dictionary*:

> *»(Design umfasst) zunächst nicht viel mehr als die einer Arbeit zugrunde liegende Konzeption. Sobald also eine planerische Absicht oder ein Entwurf einem Produkt zugrunde liegt, das arbeitsteilig hergestellt wird, handelt es sich um Design. Somit kann es im industriellen Kontext kein Nicht-Design geben, was bedeutet, dass auch das noch so schlecht gestaltete, wenig funktionsfähige Produkt Design ist. Design ist daher keineswegs eine Eigenschaft, die einem Produkt als ästhetisches Extra hinzugefügt werden kann. Design ist auch kein*

1 *http://www.designlexikon.net/Fachbegriffe/S/styling.html*
2 *http://de.wikipedia.org/wiki/Design*

Attribut, das ein Produkt hat und ein anderes nicht. Design ist vielmehr ein geplanter und umfassender Gestaltungsprozess, der durch mehrere Personen und in mehreren Arbeitsschritten realisiert wird.«

Design bedeutet folglich nicht einfach nur visuelle, Form- und Materialgestaltung mit dem Ziel, ein ästhetisches Produkt zu entwerfen, sondern hat ein ergonomisches Produkt zum Ziel unter Beachtung der Funktionalität, Gebrauchstauglichkeit und (handwerklichen) Qualität. Jedes (gegenständliche) Objekt hat ein Design beziehungsweise ist Design; diskutieren könnte man lediglich darüber, ob es ein gutes oder ein schlechtes Design ist.

6.1.2 Styling

Die Bezeichnung »Styling« entwickelte sich in den späten 1920er- und frühen 1930er-Jahren in den USA, als durch die Wirtschaftskrise der Automobilabsatz stockte. Geringfügiger äußerlicher Feinschliff sollte den Absatz wieder ankurbeln. Diese Maßnahmen hatten Erfolg, und seitdem breitete sich das Styling auf alle Produktgruppen aus.

Definitionen

Da die Bezeichnung »Styling« verhältnismäßig jung ist, gehen alle Definition in folgende Richtung:

»Styling bedeutet die formalästhetische Überarbeitung eines Produkts im Sinne einer Verschönerung mit dem Ziel besserer Verkäuflichkeit. Funktionale Verbesserungen werden dabei nicht angestrebt.« (Designlexikon International[3])

Am besten gefällt mir folgende Definition:

»Styling ist, wenn meine Teenie-Tochter sich vor dem Spiegel anmalt, bevor sie in den Club geht. Zusammenhänge zwischen Verpackung und Inhalt werden sorgfältig verwischt. Ähnlich ist es bei dem Prospekt, der kurz vor der Drucklegung einem Grafiker geschickt wird; der soll ihn noch was aufmotzen. Da ist es zum Design zu spät, das wird nur noch Styling.« (Oliver Baer in »Was passiert, wenn wir Design und Styling verwechseln?«[4])

Styling hat folglich nicht zum Ziel, schöne und gebrauchstaugliche Dinge zu erschaffen, sondern beschränkt sich lediglich darauf, schöne Ergebnisse zu erzielen. Durchsucht man Suchdienste nach der Definition von Styling, erhält man viele Treffer auf Webseiten, die sich mit Bodystyling oder Kosmetik- und Haarpflegeprodukten befassen. Der Zusammenhang zwischen Styling und äußerer Erscheinung wird dadurch sehr deutlich.

3 *http://www.designlexikon.net/Fachbegriffe/S/styling.html*
4 *http://www.baer-coach.de/deutsch/99/design_13.htm*

6.1.3 Design und Styling in der Webentwicklung

Legt man die oben aufgeführten Definitionen der Bezeichnung Design zugrunde, lässt sich die Bezeichnung »Webdesign« wie folgt definieren:

DEFINITION

Webdesign *ist die Gestaltung von Webseiten nach den Kriterien von Funktionalität und Ästhetik, mit anderen Worten: die Gestaltung einer Website derart, dass sie Benutzern in so hohem Maße wie möglich zu Verfügung steht und darüber hinaus auch noch gut aussieht.*

Bestandteile guten Webdesigns

Web-Ergonomie, Zugänglichkeit und Gebrauchstauglichkeit sind wichtige Bestandteile guten Webdesigns. Dazu gehören unter anderem

- sinnvolle Gruppierung, also eine Einteilung der Seitenelemente auf dem Bildschirm als visuell eigenständige Einheiten,
- der Erwartungshaltung der Nutzer entsprechende Platzierung der Gruppen,
- ansprechende Dichte der Information,
- intuitive und konsequente Nutzerführung und nicht zuletzt natürlich
- optisch ansprechende Gestaltung.

Das Mittel der Wahl, um Elemente zu gruppieren, zu positionieren und visuell zu gestalten, sind **Cascading Style Sheets** (*CSS*). Während eine durchdachte Informationsarchitektur den Nutzer durch die Website und logische, strukturierte Textauszeichnung per (X)HTML durch die einzelnen Webseiten führt, sorgen entsprechende CSS-Regeln für ein angenehmes Erscheinungsbild und unterstützen die Informationsaufnahme nicht zuletzt dadurch, dass sie diese visuell in Form bringen und aufwerten. Nicht zu unterschätzen ist in diesem Zusammenhang, dass Nutzer sich eher mit den Inhalten einer Website befassen, wenn sie sich wohlfühlen, also von ihr auch optisch angesprochen werden. Sorgfältiges Styling mittels CSS ist Teil des Designprozesses einer Website.

6.2 Entwurfskonzepte und Vorteile

CSS basiert auf bestimmten Entwurfskonzepten, die eine ganze Reihe von Vorteilen bieten.

- **CSS vereinfacht das Schreiben von Webseiten**: Da Sie die komplette (visuelle) Gestaltung einer Website – im Sinne der Trennung von Inhalt und Formatierung – mit CSS vornehmen können, ist es möglich, schlanke,

semantisch reichhaltige XHTML-Dokumente zu schreiben, die keinerlei Informationen zur Darstellung mehr enthalten. Dadurch sparen Sie nicht nur eine Menge Zeit, sondern auch Bandbreite. So konnten die Entwickler der US-amerikanischen Website *Multimap.com* nach einem Relaunch die Dateigröße pro Seite von 65 auf 35 KByte verringern; das entspricht bei vier Millionen Zugriffen täglich einer Ersparnis von 40.000 GByte jährlich!

- **Auf- und Abwärtskompatibilität**: CSS 1 ist im Wesentlichen eine Teilmenge von CSS 2.1. Daraus resultiert, dass Benutzerprogramme, die CSS 2.1 verstehen, auch mit CSS 1 umgehen können. Dies gilt ebenfalls umgekehrt: CSS-konforme Benutzerprogramme müssen Eigenschaften und Werte gemäß der Empfehlung interpretieren und alle ungültigen Deklarationen oder unerlaubte und unbekannte Werte ignorieren und die Abarbeitung des Stylesheets bei der nächsten Deklaration oder Regel fortsetzen.
- **Hersteller-, Plattform- und Geräteunabhängigkeit**: CSS ist ein offener Standard, der von allen Entwicklern von Browsern, Editoren und anderen Benutzerprogrammen befolgt wird.
- **Verbesserung der Wartbarkeit**: Moderne Websites bestehen aus einer Vielzahl einzelner Dokumente, deren Layout im Allgemeinen von nur einer Handvoll Stylesheets gesteuert wird. Häufig verweisen alle Dokumente auf ein einziges Stylesheet, das gegebenenfalls weitere importiert. So ist es möglich, mit der Änderung nur einer einzigen Datei das Aussehen der gesamten Website zu ändern. Sie können sich sicher vorstellen, wie viel Zeit selbst bei umfangreichen Layoutänderungen im Vergleich zu älteren Sites ohne Einsatz von CSS gespart werden kann.
- **Einfachheit**: CSS ist eine sehr einfache Stylesprache, die verständlich und schnell erlernbar ist. Die einzelnen Eigenschaften sind so weit wie möglich unabhängig voneinander, und es gibt zumeist nur einen Weg, einen gewünschten Effekt zu erreichen.
- **Flexibilität**: Jedes strukturierte Dokument können Sie mit CSS gestalten. Dabei können Sie sich die Konzepte der Vererbung und Kaskade zunutze machen, die es ermöglichen, Stylesheets Schritt für Schritt aufeinander aufbauend vom Allgemeinen hin zum Speziellen zu entwickeln.
- **Alternative Sprachbindungen**: Die Menge der in der CSS-Empfehlung beschriebenen Eigenschaften und Werte bildet ein konsistentes Formatierungsmodell, auf das Sie nicht nur per CSS zugreifen können. Es sind auch Bindungen zu anderen Sprachen möglich, etwa zu JavaScript, über das Sie sehr einfach Eigenschaftswerte dynamisch verändern können.

- **Erhöhung der Zugänglichkeit**: Die konsequente Trennung von Struktur und Darstellung bietet eine deutliche Verbesserung der Zugänglichkeit einer Website. Gerade Nutzer alternativer Benutzerprogramme wie beispielsweise Screenreader sind auf sauber strukturierte, linearisierbare, semantisch reichhaltige Dokumente angewiesen.

6.3 Einbindung von CSS in ein XHTML-Dokument

Es gibt drei gängige Methoden, CSS in ein XHTML-Dokument zu integrieren.

6.3.1 Einbindung per style-Attribut

Das `style`-Attribut ist ein Universalattribut (siehe Kapitel 4.6) und darf an fast allen XHTML-Elementen notiert werden. Folgendes Beispiel zeigt, wie Sie einen Absatz formatieren können.

```
<p style="text-align: center; font-weight: bold;">Absatz</p>
```

Vor- und Nachteile

Diese sogenannten Inline-Styles werden häufig verwendet, wenn Angaben nur ein einziges Mal für ein bestimmtes Element vorgenommen werden sollen oder die Vergabe einer Klasse oder ID nicht infrage kommt oder nicht möglich ist. Dies kann der Fall sein, wenn auf den XHTML-Output kein Einfluss genommen werden kann, zum Beispiel bei der Ausgabe eines **Content-Management-Systems**. Inline-Styles haben jedoch zwei entscheidende Nachteile:

- Die Trennung von Struktur und Präsentation ist damit hinüber.
- Inline-Styles können autorenseitig nur sehr schwer, nämlich ausschließlich mit `!important`, überschrieben werden (dazu mehr in Kapitel 6.7 »Die Kaskade«).

Browserunterstützung

Alle Browser – Inline-Styles werden von allen modernen Browsern unterstützt.

6.3.2 Einbindung per style-Element

Das `style`-Element wird innerhalb des Dokumentkopfs `head` notiert. Es kann ein vollständiges Stylesheet enthalten. Die Regel im `style`-Element in Listing 6.1 formatiert alle Elemente vom Typ `p`, also beide Absätze.

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

```
<html xmlns="http://www.w3.org/1999/xhtml"
  lang="de" xml:lang="de">
  <head>
    <meta http-equiv="Content-Type"
      content="text/html; charset=utf-8" />
    <title>CSS-Einbindung per 'style'-Element</title>

    <style type="text/css">
      p {
          text-align: center;
          font-weight: bold;
      }
    </style>
  </head>

  <body>
    <p>Erster Absatz</p>
    <p>Zweiter Absatz</p>
  </body>
</html>
```

Listing 6.1: CSS-Einbindung per `style`-Element

DEFINITION

*Das **Attribut** `type` ist ein Hinweis für Browser darauf, um welche Art Stylesheet es sich handelt – schließlich gibt es noch andere Stylesheetsprachen. Die Sprache wird in Form eines MIME-Typs angegeben.*

Gecko, Safari und der Internet Explorer behandeln diese Angabe als verbindlich; wenn Sie also etwas anderes als `text/css` angeben, wird das Stylesheet selbst dann nicht verarbeitet, wenn es doch CSS ist.

Vor- und Nachteile

`style`-Elemente werden dazu verwendet, CSS zu notieren, das dokumentweit gelten soll, in anderen Dokumenten jedoch nicht verwendet wird. Darüber hinaus dienen sie häufig dazu, Stylesheets mittels der `@import`-Regel einzubinden, was aufgrund diverser Bugs in mehreren Browsern nicht zu empfehlen ist (siehe Kapitel 6.4.4 »At-Regeln«). Die Trennung zwischen Markup und Styling ist besser als beim `style`-Attribut, jedoch noch nicht perfekt; noch immer müssten Sie alle Dokumente ändern, wenn Sie Änderungen am Layout der gesamten Website vornehmen wollten.

Browserunterstützung

Alle Browser – Die Einbindung per `style`-Element wird von allen modernen Browsern unterstützt.

6.3.3 Einbindung per link-Element

Dies ist beste Methode. Das Element `link` ist Kind des `head`-Elements (siehe Kapitel 5.2.2) und verweist auf ein externes Stylesheet.

```
<link rel="stylesheet" media="screen,projection"
  title="Standardstylesheet" href="/stylesheets/default.css"
  type="text/css" />
```

DEFINITION

*Das **Attribut** `rel` gibt die Beziehung des Elements an; daran erkennen alle Browser, dass es ein Stylesheet gibt, das sie auf das Dokument anzuwenden haben.*

DEFINITION

*Das **Attribut** `media` gibt an, für welche Ausgabemedien das referenzierte Stylesheet bestimmt ist. Es kann eine einzelne Medienbeschreibung sein oder eine durch Kommata getrennte Liste.*

Eine Liste alle Medientypen finden Sie in Tabelle 6.1 auf Seite 279.

Angabe des Ausgabemediums

Es gehört zum guten Stil, immer das Ausgabemedium anzugeben. Damit lassen sich ungewollte Nebeneffekte in den »falschen« Medien verringern. Benutzerprogramme können Zeit sparen, indem sie nur diejenigen Stylesheets aus dem Netz laden, die für das aktuelle Gerät passend sind. In diesem Fall sind die Medien `screen` (Computerbildschirme) und `projection` (Projektoren) angegeben, Letzteres, um Opera-Browser auch im Vollbildmodus mit einem Stylesheet zu versorgen.

DEFINITION

*Das **Attribut** `href` teilt dem Browser über einen URI mit, wo er das Stylesheet finden kann.*

DEFINITION

*Das **Attribut** `title` gibt dem Stylesheet einen Namen. Webautoren können auch mehrere Stylesheets mit demselben Titel versehen und sie damit als zusammengehörig kennzeichnen. Diese Stylesheets lassen sich dann in neueren Browsern gemeinsam abschalten. Bei mehreren Stylesheets mit unterschiedlichen Titeln werden nur die mit dem ersten Titel als bevorzugt behandelt und angewandt, die übrigen sind – zumindest in neueren Browsern – automatisch alternative Stylesheets.*

Vor- und Nachteile

Die Verwendung des `link`-Elements zur Einbindung von CSS stellt die optimale Trennung von Markup und Layout dar und bietet unter anderem folgende Vorteile:

- Webautoren können Stylesheets über viele Dokumente (und Websites) hinweg verwenden.
- Webautoren können das Stylesheet auswechseln und dadurch das Layout einer Website ändern, ohne Änderungen an den XHTML-Dokumenten vornehmen zu müssen.

Alternative Stylesheets

Stylesheet können persistent (beständig), bevorzugt oder alternativ eingebunden werden.

Persistente Einbindung

Das Stylesheet ist immer aktiv, es sei denn, Nutzer schalten die CSS-Unterstützung ihres Browsers komplett ab. Wenn Sie alternative Stylesheets anbieten, kann das persistente Stylesheet diejenigen Regeln enthalten, die allen Stylesheets gemeinsam sind.

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

Bevorzugte Einbindung

Dieses Stylesheet ist das Standardstylesheet. Es ist aktiv, sobald die Seite geladen ist. Es wird inaktiv, wenn stattdessen ein alternatives Stylesheet gewählt wird.

```
<link rel="stylesheet" type="text/css" title="Bevorzugt"
  media="screen,projection" href="bevorzugt.css" />
```

Mit dem Attribut `title` geben Sie dem Stylesheet eine Bezeichnung, über die es ausgewählt werden kann – dadurch unterscheidet es sich vom persistenten Stylesheet.

Alternative Einbindung

Dieses Stylesheet ist beim Laden der Seite inaktiv. Es kann vom Besucher als Alternative zum Standardstylesheet ausgewählt werden. Beachten Sie den zusätzlichen `rel`-Attributwert `alternate`.

```
<link rel="alternate stylesheet" type="text/css"
  title="Alternativ" media="screen,projection"
  href="alternativ.css" />
```

W3C Core Styles

Auf der *Web Style Sheet Home Page*[5] des W3C wird eine große Auswahl unterschiedlicher Stylesheets angeboten. Darunter sind die sogenannten W3C Core Styles[6], die jedem Webautor frei zur Verfügung stehen. Es gibt insgesamt acht unterschiedliche Stylesheets: *Oldstyle*, *Modernist*, *Midnight*, *Ultramarine*, *Swiss*, *Chocolate*, *Traditional* und *Steely*. Sie können sie jeweils auf folgende Weise in Ihre Website einbinden:

```
<link rel="alternate stylesheet" title="Oldstyle"
  type="text/css" media="screen,projection"
  href="http://www.w3.org/StyleSheets/Core/Oldstyle" />
```

Diese Stylesheets sind mittlerweile veraltet und haben den einen oder anderen Mangel (so verfügt beispielsweise keines über unterstrichene Anker), aber sie eignen sich hervorragend dazu, mal eben ein Dokument zu layouten, wenn die Zeit fehlt, ein eigenes Stylesheet zu schreiben.

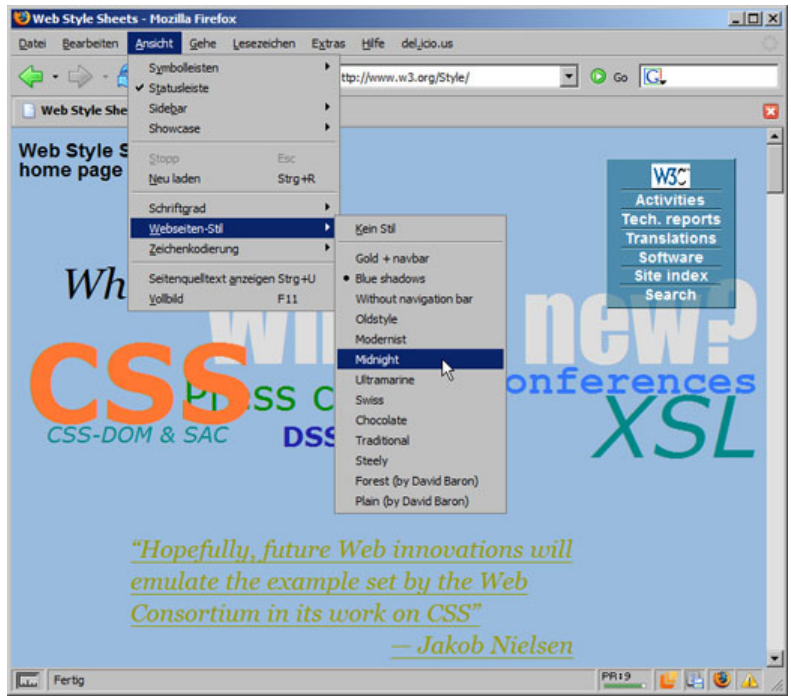


Abbildung 6.1: Auswahl alternativer Stylesheets

5 *http://www.w3.org/Style/*
6 *http://www.w3.org/StyleSheets/Core/*

Auswahl des Stylesheets

Abbildung 6.1 zeigt, wie Sie das Stylesheet im Browser Firefox wechseln können. In Mozilla- und Opera-Browsern können Sie ähnlich vorgehen. Nutzer des Internet Explorers schauen mal wieder in die Röhre. Dazu kommt leider, dass diese Funktion auch in Browsern, die sie unterstützen, nicht praxistauglich ist. Nach dem Neuladen der Seite wird wieder das Standardstylesheet angezeigt, ebenso dann, wenn Sie von einer Seite der Website zu einer anderen wechseln. Es bleibt in dem Fall nichts anderes übrig, als sich mit dem Standardstylesheet zufriedenzugeben oder die Auswahl erneut zu treffen.

Browserunterstützung

Alle Browser – Die Einbindung per `link`-Element wird im Wesentlichen von allen modernen Browsern unterstützt.

6.3.4 Ausgabemedien

CSS bietet die Möglichkeit, Stylesheets für unterschiedliche Ausgabemedien festzulegen. So können Sie für Ihr Dokument eigene Stylesheets zum Beispiel für die Präsentation am Bildschirm und für die Ausgabe über einen Drucker schreiben. Das Benutzerprogramm beziehungsweise das Gerät, das Ihre Webseite darstellt, sollte das geeignete Stylesheet auswählen.

Folgende Medientypen stehen zur Verfügung:

Typ	Medium
`all`	Für alle Geräte.
`aural`	Für Sprachbrowser, die den Inhalt der Webseite vorlesen.
`braille`	Für Ausgabegeräte mit sogenannter Braillezeile. Der Text der Webseite wird in die veränderte Oberflächenstruktur der Braillezeile umgewandelt und ist so für blinde Menschen tastbar.
`embossed`	Für »Brailledrucker«. Der Text der Website wird dabei gestanzt und ist dann für blinde Menschen tastbar.
`handheld`	Für kleine Handcomputer wie Palms oder Mobiltelefone, zumeist mit kleinen monochromen Bildschirmen und geringer Bandbreite.
`print`	Für die Ausgabe durch Drucker.
`projection`	Für Projektoren und ähnliche Geräte.
`screen`	Für die Ausgabe auf dem Bildschirm, also auf einem Computermonitor.
`tty`	Für die Ausgabe auf nicht grafischen Ausgabemedien mit Festbreitenschrift, wie zum Beispiel Terminals oder auch Textbrowser wie Lynx.
`tv`	Für die Ausgabe auf Fernsehern oder ähnlichen Geräten.

Tabelle 6.1: CSS-Medientypen

6.4 CSS: Syntax und Vokabular

6.4.1 Regeln

Ein Stylesheet besteht vor allem aus einer Menge von Regeln.

DEFINITION

*Eine **Regel** (auch: Regelmenge, Regelsatz) ist die Grundkonstruktion eines Stylesheets. Regeln bestehen aus mindestens einem **Selektor** und einem Paar geschweifter Klammern, Deklarationsblock genannt, die eine Liste mit null oder mehr durch Semikola voneinander abgetrennten Deklarationen enthalten.*

DEFINITION

*Eine Deklaration ist Bestandteil eines CSS-Deklarationsblocks. Deklarationen bestehen aus einer **Eigenschaft**, gefolgt von einem Doppelpunkt, gefolgt von einem **Wert**. Vor und hinter diesen Bestandteilen kann sich Leerraum befinden. Deklarationen werden durch Semikola voneinander getrennt.*

Teilen sich mehrere Selektoren denselben Deklarationsblock, können sie durch Kommata getrennt nacheinander aufgeführt werden. Beachten Sie auch die übersichtliche Blockschreibweise bei der Notation mehrerer Deklarationen innerhalb eines Blocks.

```
p { margin: 0; }

h1, h2, h3, h4, h5, h6 {
  color: #889399;
  background-color: transparent;
  margin: 2em 0 1em 0;
}
```

Tabelle 6.2 verdeutlicht die Bezeichnungen.

Bezeichnung	Beispiel
Regel	`p { margin: 0; }`
Selektor	`p { margin: 0; }`
Deklarationsblock	`p { margin: 0; }`
Deklaration	`p { margin: 0; }`
Eigenschaft	`p { margin: 0; }`
Wert	`p { margin: 0; }`

Tabelle 6.2: Regel

• • • DEFINITION

*Einige Eigenschaften sind **zusammenfassende Eigenschaften** (*shorthand properties*), die es Autoren erlauben, die Werte mehrere Eigenschaften innerhalb einer einzigen Eigenschaft anzugeben.*

So lassen sich diese Eigenschaften

```
p {
  border-width: 1px;
  border-style: solid;
  border-color: red;
}
```

in der folgenden Eigenschaft zusammenfassen:

```
p { border: 1px solid red; }
```

! ! ! ACHTUNG

Werden in einer zusammenfassenden Form Werte weggelassen, wird diesen »fehlenden« Eigenschaften ihr Ausgangswert zugewiesen (lesen Sie dazu Kapitel 6.6 »Vererbung und Initialwerte«).

6.4.2 Maskierung

Für alle formalen Sprachen gelten sehr strenge Regeln, die festlegen, welche Zeichen an welcher Stelle erlaubt sind. Möchten Sie ein Zeichen verwenden, das syntaktisch an dieser Stelle nicht zulässig ist oder eine besondere Bedeutung hat, die an dieser Stelle nicht gewünscht ist, müssen Sie es *maskieren*, das heißt, Sie müssen ihm seine besondere Bedeutung an dieser Stelle nehmen. In CSS maskieren Sie ein Zeichen, indem Sie ihm einen Backslash (\) voranstellen. Die beiden folgenden Regeln sind somit äquivalent:

```
bo\
dy { margin: 0; }

body { margin: 0; }
```

Anwendung

Dass einige veraltete Browser mit der Zeichenmaskierung nicht klarkommen, lässt sich für CSS-Hacks ausnutzen (siehe Kapitel 6.8.4 »CSS-Hacks für den Internet Explorer«).

Webautoren benötigen Zeichenmaskierung auch dann, wenn sie Zeichen benutzen wollen, die außerhalb des Zeichenvorrats der gewählten Zeichenkodierung liegen. In diesen Fällen folgt dem Backslash die hexadezimale Referenz der Unicode-Position des Zeichens.

```
a[href^="http://"]:before,
a[href^="https://"]:before {
  /* Nordostpfeil und Leerzeichen vor externen Links */
  content: "\2197\00A0";
}
```

Browserunterstützung

Alle Browser – Maskierung wird von allen modernen Browsern unterstützt.

6.4.3 Kommentare

Wie in XHTML gibt es auch in CSS die Möglichkeit, Kommentare zu notieren. Deren Syntax ist an die Programmiersprache C angelehnt und sieht wie folgt aus:

```
/* Einzeiliger Kommentar */

/* Mehrzeiliger
  Kommentar */
```

Kommentare dürfen nicht verschachtelt werden.

CSSDOC

Umfangreiche Websites, an denen mehrere Entwickler über einen längeren Zeitraum gemeinsam arbeiten, erfordern klare Konventionen, Tabus und umfassende Dokumentationen. Dabei sind Kommentare besonders wichtig. Jeder gewissenhafte Webautor kommentiert seine Stylesheets sauber, ausführlich und präzise. Kommentare erhöhen die Übersichtlichkeit und Verständlichkeit und helfen dabei, sich in einem fremden Stylesheet zurechtzufinden – oder in einem eigenen, an dem man schon seit einigen Wochen nicht mehr gearbeitet hat.

Leider ist die ausführliche Dokumentation eines Stylesheets eher die Seltenheit. Wenn Webautoren ihre Stylesheets kommentieren, dann tun sie es häufig zur Strukturierung, weniger zur Dokumentation, und jeder so, wie er es sich im Laufe der Jahre angewöhnt hat. Natürlich einigen auch Webautoren sich schnell auf einen gemeinsamen Kommentarstil, wenn sie zusammen an denselben Stylesheets arbeiten, aber sie sind derzeit auf sich allein gestellt, da kein einheitlicher Dokumentationsstandard zur Verfügung steht. Hier soll CSSDOC[7] Abhilfe schaffen.

7 *http://cssdoc.net*

CSSDOC ist ein sinnvoller Arbeitsentwurf eines längst überfälligen Dokumentationsstandard für CSS-Stylesheets. Derzeit gibt es noch keine fertige CSSDOC-Spezifikation und daher auch noch keine Werkzeuge, die direkt aus den Kommentaren eine Dokumentation erzeugen oder Webautoren bei der Kommentierung ihrer Stylesheets unterstützen. Sobald die erste stabile Dokumentation vorliegt, ist auch schnell mit entsprechenden Implementierungen zu rechnen.

CSSDOC-Syntax

Webautoren notieren CSSDOC-Kommentare in Form spezieller Kommentarblöcke, DocBlock genannt. DocBlöcke werden mit `/**` eingeleitet und mit `*/` beendet. Alle Zeilen innerhalb eines Blocks beginnen mit `*`. Jeder Block besteht aus bis zu drei Teilen: Titel, Beschreibung und CSSDOC-Tags. Diese Teile werden durch eine leere Kommentarzeile voneinander abgegrenzt.

```
/**
 * Ein einfacher DocBlock ohne Tags
 *
 * Hier steht die Beschreibung. Wenn sie besonders lang
 * ist, kann sie sich über mehrere Zeilen erstrecken.
 */
```

CSSDOC unterscheidet zwischen Dateikommentaren (*File Comments*), die den Inhalt einer Datei beschreiben, und Kommentarblöcken für einzelne Abschnitte (*Section Comments*). Der Dateikommentar dient der einleitenden Beschreibung eines CSS-Stylesheets. Er steht direkt zu Beginn des Stylesheets beziehungsweise direkt nach der Angabe der Zeichenkodierung (`@charset`). Abschnittskommentare unterteilen ein umfangreiches Stylesheet in mehrere Teile. Welche Teile das sind, gibt CSSDOC nicht vor – hier gilt es, eigene Konventionen aufzustellen. Abbildung 6.2 zeigt die grobe Aufteilung eines Stylesheets schematisch.

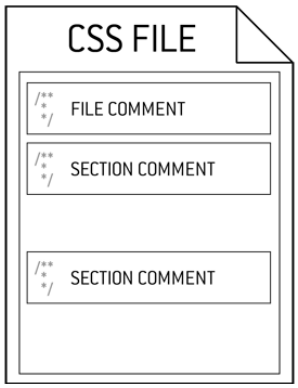


Abbildung 6.2: Die Aufteilung eines Stylesheets mittels CSSDOC

Tags

Tags sind die (maschinenlesbaren) Metainformationen eines CSSDOC-Kommentarblocks. Sie enthalten unter anderem Strukturinformationen, Autoren- und Copyrightinformationen oder beschreiben die Browserkompatibilität des kommentierten CSS. Tags stehen am Ende eines Kommentarblocks nach dem Titel und der Beschreibung.

Ein Tag besteht aus Name und Wert, wobei der Name mit einem `@` beginnt. Tabelle 6.3 führt einige der derzeit verfügbaren Tags auf und gibt an, wo und wie sie angewendet werden. Eine vollständige Liste aller Tags mit Anwendungsbeispielen bietet der aktuelle CSSDOC-Arbeitsentwurf[8].

Tag	Verwendung	Wert	Bedeutung
`@affected`	Dateikommentar, Abschnittskommentare	Browser und Version	Angabe der Browser, für die ein bestimmter Bugfix oder Workaround vorgenommen wird
`@author`	Dateikommentar	Name und gegebenenfalls E-Mail-Adresse	Angabe des verantwortlichen Stylesheetautors; ein Autor pro @autor-Tag
`@bugfix`	Abschnittskommentare	nichts oder eine Kurzbeschreibung	Dokumentation eines Bugfix
`@colordef`	Dateikommentar	Farbwert, Farbname, Kurzbeschreibung; die drei Angaben werden mit Semikola voneinander getrennt	Definition und Beschreibung einer im Stylesheet verwendeten Farbe
`@copyright`	Dateikommentar	Zeichenkette	Copyrightangabe
`@css-for`	Dateikommentar, Abschnittskommentare	Browser und Version	Angabe der Browser, die das nachfolgende CSS interpretieren können
`@date`	Dateikommentar	Datumsangabe nach ISO 8601	Erstellungs- bzw. Veröffentlichungsdatum des Stylesheets
`@lastmodified`	Dateikommentar	Datumsangabe nach ISO 8601	Datum der letzten Änderung
`@license`	Dateikommentar	Zeichenkette	Angabe der Lizenzform des Stylesheets
`@link`	Dateikommentar	URI	Angabe einer Webseite, in deren Kontext das Stylesheet steht

Tabelle 6.3: CSSDOC-Tags

8 *http://cssdoc.net/wiki/CssdocDraft*

Tag	Verwendung	Wert	Bedeutung
`@note`	Dateikommentar, Abschnittskommentare	Zeichenkette	Notiz
`@package`	Dateikommentar	Bezeichner	Angabe des Pakets, zu dem das Stylesheet gehört
`@section`	Abschnittskommentare	Zeichenkette	Beschreibung des folgenden Abschnitts
`@see`	Dateikommentar, Abschnittskommentare	URI	Angabe einer Webseite, die weitere Informationen über das nachfolgende CSS enthält
`@since`	Dateikommentar, Abschnittskommentare	Datumsangabe nach ISO 8601	Angabe des Datums, wann das nachfolgende CSS eingefügt wurde
`@todo`	Dateikommentar, Abschnittskommentare	Zeichenkette	Notiz einer noch zu erledigenden Aufgabe
`@valid`	Abschnittskommentare	»yes« oder »no«	Angabe der Gültigkeit eines verwendeten Bugfix oder Workarounds
`@version`	Dateikommentar	Zeichenkette	Angabe der Versionsnummer des Stylesheets
`@workaround`	Abschnittskommentare	nichts oder eine Kurzbeschreibung	Dokumentation eines Workarounds

Tabelle 6.3: CSSDOC-Tags (Fortsetzung)

Zwei Beispiele sollen die Verwendung von CSSDOC demonstrieren.

Beispiele

Listing 6.2 zeigt einen typischen Dateikommentar, der einige Informationen über den Autor des Stylesheets sowie Copyright- und Lizenzinformationen enthält. Natürlich kann der Dateikommentar noch weitere der in Tabelle 6.3 aufgeführten Tags enthalten.

```
/**
 * Stylesheet für http://jendryschik.de/
 *
 * Standardstylesheet
 *
 * @author    Michael Jendryschik
 * @link      http://jendryschik.de/
 * @copyright Copyright 2000-2008, Michael Jendryschik
 * @license   CC-A 2.0
 */
```

Listing 6.2: CSSDOC-Dateikommentar

In Listing 6.3 ist zu sehen, wie ein Bugfix kommentiert wird. Das Tag `@bugfix` stellt klar, dass es sich bei dem nachfolgenden CSS um einen Bugfix handelt. `@affected` führt die Browser auf, deren Bug behoben werden soll, `@css-for` wiederum die Browser, die das Bugfix-CSS lesen können. `@valid` gibt an, ob der Bugfix gültiges CSS verwendet, und `@see`, wo es weitere Informationen über den Bugfix gibt.

```
/**
 * Korrektur des Italics-Bugs
 *
 * @bugfix
 * @affected IE 5/Win, IE6
 * @css-for  IE 5/Win, IE6
 * @valid    yes
 * @see      http://www.highresolution.info/weblog/entry/
 *           dem_italics_bug_die_zaehne_zeigen/
 */
* html body * { overflow: visible; }
* html iframe, * html frame { overflow: auto; }
* html frameset { overflow: hidden; }
```

Listing 6.3: CSSDOC-Abschnittskommentar

In Kapitel 8 bauen wir gemeinsam eine kleine Website auf und schreiben ein vollständiges Stylesheet, das wir mit CSSDOC kommentieren und dokumentieren werden.

6.4.4 At-Regeln

DEFINITION

Neben gewöhnlichen Regelsätzen gibt es mit At-Regeln noch eine zweite Art von Regeln. Sie dienen besonderen Zwecken, die durch andere Regeln nicht erfüllt werden. So können mit At-Regeln unter anderem die Wirksamkeit von darin enthaltenen Regeln auf bestimmte Medien eingeschränkt, Stylesheets importiert und die Zeichenkodierung vorgegeben werden.

Eine At-Regel beginnt mit `@` (daher der Name), gefolgt von einem Bezeichner, und reicht bis zum schließenden Semikolon (`;`) oder dem Ende des folgenden Deklarationsblocks.

Ausgabe für bestimmte Medien

Mithilfe der At-Regel `@media` lassen sich »gewöhnliche« Regeln umschließen und deren Geltungsbereich auf bestimmte Ausgabemedien einschränken:

```
@media screen, handheld {
  body {
    color: #eee;
    background: #333;
  }
}

@media print {
  body {
    color: #000;
    background: #fff;
  }
}
```

Die erste `@media`-Regel schränkt den Geltungsbereich der darin enthaltenen Regel auf die Ausgabemedien `screen` und `handheld` ein; die zweite tut dasselbe für die Druckausgabe. Natürlich könnten die `@media`-Regeln noch weitere Regeln enthalten und theoretisch beliebig oft innerhalb eines Stylesheets vorkommen. Dadurch ist es zwar möglich, Deklarationen für die unterschiedlichen Medientypen innerhalb einer einzigen Datei zusammenzufassen, jedoch wird das Stylesheet so schnell unübersichtlich. Eher empfiehlt sich die Einbindung eines eigenen Stylesheets pro Medium per `link`-Element.

Browserunterstützung

Alle Browser – Die At-Regel `@media` wird von allen modernen Browsern unterstützt.

Stylesheets importieren

Die At-Regel `@import` muss vor allen anderen Regeln eines Stylesheets stehen und dient dazu, Regeln aus externen Stylesheets in eine CSS-Datei oder in das `style`-Element eines XHTML-Dokuments zu importieren. Dabei gibt es zwei Schreibweisen:

```
@import "fonts.css";

@import url("fonts.css");
```

Auf vielen Webseiten ist dies gängige Praxis. Aufgrund kleinerer, ärgerlicher Fehler in einigen Browsern sollten Sie in Stylesheets nicht importieren.

Browserunterstützung

Die At-Regel `@import` wird von allen modernen Browsern unterstützt mit folgenden Einschränkungen:

Ärgerlich ist ein Fehler im IE 6, der als FOUC[9] bekannt ist. Das Dokument wird kurz ohne Stylesheet angezeigt und ruckelt sich dann zurecht.

Gecko-Browser verlieren beim Abspeichern eines Dokuments alle Stylesheets, die mit `@import` eingebunden wurden.

Angabe der Zeichenkodierung

Die At-Regel `@charset` darf nur jeweils ein Mal in Stylesheets verwendet werden. Sie muss direkt zu Beginn der Datei notiert werden, auch Kommentare und Leerraum sind nicht erlaubt. Sie dient dazu, die Zeichenkodierung des Dokuments anzugeben, beispielsweise UTF-8:

```
@charset "UTF-8";
```

Browserunterstützung

Alle Browser – Die Unterstützung der At-Regel `@charset` entspricht der Unterstützung der verschiedenen Zeichensätze durch die Browser.

6.5 Selektoren

DEFINITION

In CSS legen Regeln für den Mustervergleich fest, welche Elemente im Dokumentbaum angesprochen werden. Diese Muster, die sogenannten ***Selektoren****, können von einfachen Elementnamen bis hin zu umfassenden Kontextmustern reichen. Wenn alle Bedingungen eines Musters für ein bestimmtes Element zutreffen, stimmt der Selektor mit dem Element überein, und die Regeln können auf das Element angewandt werden.*

Auf syntaktischer Ebene ist ein Selektor eine Folge einfacher Selektoren, die durch sogenannte **Kombinatoren** getrennt werden.

DEFINITION

Ein ***einfacher Selektor*** *ist entweder ein Typselektor, der Universalselektor, ein Attributselektor, ein Klassenselektor, ein ID-Selektor, ein Namensraumselektor oder eine Pseudoklasse. Ein Pseudoelement kann an den letzten einfachen Selektor in einer Kette angehängt werden.*

9 Flash Of Unstyled Content, siehe *http://www.bluerobot.com/web/css/fouc.asp*

DEFINITION

Kombinatoren *sind Leerraum, das Größer-als-Zeichen (>), das Pluszeichen (+) und die Tilde (~). Leerraum darf nur aus Spatium (Leerzeichen,* space*), Tabulator (*tab*), Zeilenvorschub (*line feed*) und Wagenrücklauf (*carriage return*) bestehen.*

Ein Selektor, der aus einem einzigen einfachen Selektor besteht, stimmt mit jedem beliebigen Element überein, das seine Anforderungen erfüllt. Werden eine Folge einfacher Selektoren und entsprechende Kombinatoren vorangestellt, kommen zusätzliche Übereinstimmungsbeschränkungen hinzu. Man spricht dann von *Kontextselektoren*. Auf diese Art und Weise können immer speziellere Teile des Dokumentbaums angesprochen werden.

In den nächsten Abschnitten führe ich viele (einfache) Selektoren und Kombinatoren auf und erläutere ihre Bedeutung. Ich erkläre, welche Elemente im Dokumentbaum der Selektor anspricht. Die Tabellen enthalten Selektoren aus CSS 1 bis CSS 3, die ich zur besseren Übersicht thematisch, nicht chronologisch, sortiert habe. Verstehen Sie die verwendeten Elementnamen *E* und *F*, den Attributnamen *foo* sowie den Attributwert *bar* als Variablen (Platzhalter) für tatsächliche Elementtypen und Attribute.

Zum Verständnis der Bedeutungen ist es wichtig, dass Sie sich folgende Begriffe innerhalb des Dokumentbaums noch einmal in Erinnerung rufen: »Vorfahrenelement«, »Nachfahrenelement«, »Elternelement«, »Kindelement« und »Geschwisterelement« (vgl. Kapitel 4.3).

Die Liste der möglichen Selektoren in CSS 3 ist zu lang, um sie hier ausführlich zu behandeln. Deshalb beschränke ich mich auf jene, die bereits eine nennenswerte Unterstützung in aktuellen Browsern erfahren.

6.5.1 Universalselektor

Selektor	Bezeichnung	Bedeutung
*	Universalselektor	alle Elemente

Tabelle 6.4: Universalselektor

Der Universalselektor trifft auf alle Elemente zu und kann daher zumeist einfach weggelassen werden. Die beiden folgenden Regeln sind bedeutungsgleich.

```
* { margin: 0; }

{ margin: 0; }
```

Browserunterstützung

Der Universalselektor wird von allen modernen Browsern erkannt, mit folgender Besonderheit:

Der Internet Explorer bis Version 6 entfernt alle führenden `*` eines Selektors, bevor er die passenden Elemente sucht. Im Ergebnis behandelt er `* html` daher genauso wie ein alleinstehendes `html`. Dieses Verhalten lässt sich für CSS-Hacks ausnutzen, siehe Kapitel 6.8.4.

6.5.2 Typselektor

Selektor	Bezeichnung	Bedeutung
`E`	Typselektor	Element `E`

Tabelle 6.5: Typselektor

Der Typselektor spricht Elemente anhand des Elementnamens an. Die Regel wird auf alle Elemente des entsprechenden Typs angewandt, egal an welcher Stelle des Dokuments sie notiert sind und welcher Klasse sie angehören oder welchen Bezeichner sie tragen.

```
p { margin: 0; }
```

Browserunterstützung

Alle Browser – Der Typselektor wird von allen modernen Browsern erkannt.

6.5.3 Klassen- und ID-Selektor

Selektor	Bezeichnung	Bedeutung
`E.klasse`	Klassenselektor	Element `E` der Klasse `klasse`
`E#bezeichner`	ID-Selektor	Element `E` mit der ID `bezeichner`

Tabelle 6.6: Klassen- und ID-Selektor

Klassenselektor

Das Universalattribut `class` können Sie für fast alle Elemente vergeben (siehe Kapitel 4.6). Sie sprechen es mit CSS über den Klassenselektor an, indem Sie den Klassennamen mit einem führenden Punkt ohne Leerzeichen direkt hinter den einfachen Selektor setzten. Die Regel

```
p.hinweis { margin: 0; }
```

spricht alle Elemente vom Typ `p` an, deren Attribut `class` den Wert `hinweis` enthält, beispielsweise folgende:

```
<p class="hinweis">Bitte füllen Sie alle mit einem
 Asterisk (*) markierten Felder aus.</p>

<p class="fehler hinweis">Es ist ein Fehler aufgetreten.<p>
```

Der Selektor `.hinweis` spricht alle Elemente der Klasse `hinweis` an, egal welchen Typs, und ist äquivalent zu `*.hinweis`.

Kombinierter Klassenselektor

Der folgende kombinierte Klassenselektor spricht Elemente an, die sowohl zur Klasse `fehler` als auch zur Klasse `hinweis` gehören:

```
.fehler.hinweis { margin: 0; }
```

ID-Selektor

Das Universalattribut `id` hat in XHTML-Dokumenten eine besondere Bedeutung, weil sein Wert pro Dokument nur ein Mal auftreten darf und Elemente mit diesem Attribut als Zielanker eines Links dienen können. Aus der zweiten Eigenschaft ergibt sich auch die Notation des ID-Selektors in CSS: Es wird eine Raute (#) vorangestellt:

```
ul#navigation { margin: 0; }
```

Browserunterstützung

Der Klassen- und der ID-Selektor werden von allen modernen Browsern erkannt, mit folgender Ausnahme:

Der Internet Explorer für Windows bis Version 6 beachtet bei kombinierten Klassenselektoren nur den letzten. Er wendet die Regel aus Beispiel 6 auf alle Elemente an, die eine Klasse `hinweis` haben, auch wenn die Klasse `fehler` fehlt. Aktuellere Versionen des IE verhalten sich korrekt.

6.5.4 Attributselektoren

Selektor	Bezeichnung	Bedeutung
`E[foo]`	Attributselektor	Element `E` mit Attribut `foo`
`E[foo="bar"]`	Attributselektor	Element `E` mit Attribut `foo` und Wert `bar`
`E[foo~="bar"]`	Attributselektor	Element `E` mit Attribut `foo`, dessen durch Leerzeichen getrennte Liste von Werten `bar` enthält (`p[class~="bar"]` entspricht `p.bar`)
`E[foo|="bar"]`	Attributselektor	Element `E` mit Attribut `foo`, dessen durch Trennstriche (`-`) getrennte Liste von Werten mit `bar` beginnt

Tabelle 6.7: Attributselektoren

Selektor	Bezeichnung	Bedeutung
`E[foo^="bar"]`	Attributselektor	Element `E` mit Attribut `foo`, dessen Wert mit `bar` beginnt
`E[foo$="bar"]`	Attributselektor	Element `E` mit Attribut `foo`, dessen Wert mit `bar` endet
`E[foo*="bar"]`	Attributselektor	Element `E` mit Attribut `foo`, das die Zeichenkette `bar` enthält

Tabelle 6.7: Attributselektoren (Fortsetzung)

Seit CSS 2 können Webautoren mithilfe der ersten vier Attributselektoren aus Tabelle 6.7 Elemente auch über ihre Attribute ansprechen. CSS 3 komplettiert die Liste mit den drei übrigen Attributselektoren. Der Link

```
<a href="http://example.com/" lang="de-DE" xml:lang="de-DE"
  title="Ein Link!">Beispiel</a>
```

kann unter anderem über folgende Attributselektoren angesprochen werden:

Anwendung

- `a[href]`: Das Element hat ein Attribut `href`.
- `a[href="http://example.com/"]`: Das Element hat ein Attribut `href` mit dem Wert `http://example.com/`.
- `a[title~="Ein"]`: Das Element hat ein Attribut `title`, in dem der Ausdruck »Ein« vorkommt. Dieser Ausdruck muss von weiteren durch mindestens ein Leerzeichen getrennt sein oder allein stehen – das ist hier der Fall.
- `a[lang|="de"]`: Das Element hat ein Attribut `lang`, dessen Wert mit »de-« beginnt.
- `a[href^="http"]`: Das Element hat ein Attribut `href`, das mit der Zeichenkette »http« beginnt.
- `a[title$="!"]`: Das Element hat ein Attribut `title`, das mit der Zeichenkette »!« endet.
- `a[href*="example"]`: Das Element hat ein Attribut `href`, dessen Wert die Zeichenkette »example« enthält.

Browserunterstützung

Aktuelle Versionen von Safari, Opera, Mozilla und iCab interpretieren zumindest die ersten beiden Attributselektoren und zum Teil auch alle weiteren.

Der IE ist systemübergreifend bis zur Version 6 nicht in der Lage, den Attributselektor zu erkennen.

6.5.5 Pseudoklassen

Mit Pseudoklassen können Webautoren Elemente anhand von Informationen auswählen, die entweder außerhalb der Dokumentstruktur liegen oder mit anderen Selektoren nicht ausgedrückt werden können. Die Notation beginnt immer mit genau einem Doppelpunkt (`:`).

Link-Pseudoklassen

Selektor	Bezeichnung	Bedeutung
`E:link`	Link-Pseudoklasse	Element `E`, wenn `E` der Quellanker eines *unbesuchten* Hyperlinks ist (üblicherweise das Element `a`)
`E:visited`	Link-Pseudoklasse	Element `E`, wenn `E` der Quellanker eines *besuchten* Hyperlinks ist

Tabelle 6.8: Link-Pseudoklassen

Die beiden Linkzustände `:link` und `:visited` treten ein, wenn der Zielanker eines Hyperlinks unbesucht oder besucht ist. Eine Ausnahme bildet der Internet Explorer bis Version 6, der seiteninterne Links grundsätzlich als besucht, also als `:visited`, kennzeichnet, unabhängig davon, ob sie tatsächlich schon angeklickt wurden.

Dynamische Pseudoklassen

Selektor	Bezeichnung	Bedeutung
`E:hover`	dynamische Pseudoklasse	Element `E`, wenn `E` mit dem (Maus-)Zeiger berührt wird
`E:focus`	dynamische Pseudoklasse	Element `E`, wenn `E` den Fokus hat
`E:active`	dynamische Pseudoklasse	Element `E`, wenn `E` aktiviert ist (beispielsweise der aktivierte Quellanker eines Hyperlinks)

Tabelle 6.9: Dynamische Pseudoklassen

Dynamische Pseudoklassen werden von alten Browsern nur für Links unterstützt; moderne Browser wenden sie aber grundsätzlich auf jedes Element an.

Besonders `:focus` wird immer häufiger zum Gestalten von Formularen eingesetzt. Mit

```
input[type="text"]:focus,
textarea:focus {
  color: black;
  background-color: #FFFF80;
}
```

können Webautoren das gerade fokussierte Texteingabefeld gelb hinterlegen, also das Feld, in das der Nutzer gerade schreiben kann. Besonders nützlich ist

dies bei längeren Formularen, bei denen Besucher zwischendurch vielleicht kurz abgelenkt sind, um beispielsweise etwas nachzuschlagen.

Der Internet Explorer unterstützt bis Version 6 schon `:active`, wenn ein Link erst den Fokus hat. `:focus` selbst versteht er nicht.

! ! ! ACHTUNG

Es ist sehr wichtig, dass Sie bei der Deklaration der Link- und dynamischen Pseudoklassen für Anker unbedingt die Reihenfolge `:link`, `:visited`, `:hover`, `:focus`, `:active` einhalten. Der Zielanker eines Links ist entweder noch nicht besucht worden (`:link`), oder er wurde bereits besucht (`:visited`) – beide Zustände schließen sich gegenseitig aus. Zusätzlich kann der Link mit dem Mauszeiger berührt werden (`:hover`) oder den Fokus haben (`:focus`). Zudem kann er aktiviert sein (`:active`). Wenn Sie beispielsweise `a:hover` vor `a:visited` aufführen, überschreiben (siehe Kapitel 6.7 »Die Kaskade«) die Deklarationen für `a:visited` die Deklarationen derselben Eigenschaften für `a:hover`; die Benutzerinteraktion ist somit nicht zu sehen. Aus demselben Grund sollte `a:active` stets nach `a:hover` und `a:focus` deklariert werden.

Struktur-Pseudoklassen

Selektor	Bezeichnung	Bedeutung
`E:first-child`	Struktur-Pseudoklasse	Element `E`, das erstes Kind seines Elternelements ist
`E:last-child`	Struktur-Pseudoklasse	Element `E`, das letztes Kind seines Elternelements ist
`E:first-of-type`	Struktur-Pseudoklasse	erstes Element `E`, das Kind seines Elternelements ist
`E:last-of-type`	Struktur-Pseudoklasse	letztes Element `E`, das Kind seines Elternelements ist
`E:only-child`	Struktur-Pseudoklasse	Element `E`, das einziges Kind seines Elternelements ist
`E:only-of-type`	Struktur-Pseudoklasse	einziges Element `E` in einer Reihe von Geschwisterelementen
`E:nth-child(n)`	Struktur-Pseudoklasse	Element `E`, das von vorn gezählt *n*-tes Kind seines Elternelements ist (der Index beginnt bei 1)
`E:nth-last-child(n)`	Struktur-Pseudoklasse	Element `E`, das von hinten gezählt *n*-tes Kind seines Elternelements ist
`E:nth-of-type(n)`	Struktur-Pseudoklasse	von vorn gezählt *n*-tes Element `E` in einer Reihe von Geschwisterelementen
`E:nth-last-of-type(n)`	Struktur-Pseudoklasse	von hinten gezählt *n*-tes Element `E` in einer Reihe von Geschwisterelementen

Tabelle 6.10: Struktur-Pseudoklassen

Selektor	Bezeichnung	Bedeutung
`E:empty`	Struktur-Pseudoklasse	Element `E` ohne Nachfahren (auch keine Textknoten, also auch kein textueller Inhalt)
`E:root`	Struktur-Pseudoklasse	Wurzelelement `E` eines Dokuments (in (X)HTML-Dokumenten äquivalent zum Selektor `html`)

Tabelle 6.10: Struktur-Pseudoklassen (Fortsetzung)

Über Struktur-Pseudoklassen können Webautoren Elemente anhand ihrer Position in der Dokumentstruktur ansprechen. `:first-child` gibt es schon in CSS 2, es spricht ein Element an, wenn es das erste Kind seines Elternelements ist.

Mit Ausnahme des Internet Explorers können alle aktuellen Browser diese Pseudoklasse verstehen. Alle anderen Struktur-Pseudoklassen sind Bestandteil von CSS 3 und werden bislang nur vereinzelt unterstützt. Ich führe sie an dieser Stelle hauptsächlich deshalb auf, weil sie so interessante Möglichkeiten bieten und über kurz oder lang sicherlich auch eine bessere Browserunterstützung erfahren werden.

`:nth-child(n)` ist eine Art mathematischer Selektor: Wenn man das *n* durch einen Zähler ersetzt, kann man alle Kinder an ungerader (`E:nth-child(2n+1)`) beziehungsweise gerader (`E:nth-child(2n)`) Position herausfinden – die Schlüsselwörter `odd` und `even` leisten dasselbe. Endlich könnten Webautoren auf einfache Weise Tabellenzeilen unterschiedlich einfärben:

```
/* Zeilen mit ungerader Nummer */
tr:nth-child(odd) {
  color: black;
  background-color: white;
}
/* Zeilen mit gerader Nummer */
tr:nth-child(even) {
  color: black;
  background-color: #ddd;
}
```

Es kann nicht schaden, sein Stylesheet bereits heute entsprechend aufzubereiten, um sich irgendwann über die zunehmende Anzahl von Browsern zu freuen, die den gewünschten Effekt anzeigen können.

Verweisziel-Pseudoklasse

Selektor	Bezeichnung	Bedeutung
`E:target`	Verweisziel-Pseudoklasse	Element `E`, das aktiver (aktuell angesprungener) Zielanker eines Hyperlinks ist

Tabelle 6.11: Verweisziel-Pseudoklasse

Die Verweisziel-Pseudoklasse `:target` wird auf Elemente angewandt, die das Ziel eines angesprungenen Hyperlinks sind. Besonders nützlich ist dies, wenn das Linkziel am Ende eines Dokuments liegt und durch die Position allein nicht sofort klar wird, was gerade angesprungen wurde. Unterstützt wird dieser Selektor bislang nur von aktuellen KHTML- und Gecko-Browsern.

```
h2:target { border: 2px solid #003359; }
```

Sprach-Pseudoklasse

Selektor	Bezeichnung	Bedeutung
`E:lang(x)`	Sprach-Pseudoklasse	Element `E` der (natürlichen) Sprache *x*

Tabelle 6.12: Sprach-Pseudoklasse

Wenn Sie die natürliche Sprache eines Elements mit dem Attribut `lang` gekennzeichnet haben, können Sie dieses und seine Nachfahren über die Sprach-Pseudoklasse gesondert ansprechen.

Folgende Regel stellt alle Elemente mit kursiver Schrift dar, die mit `lang="en"` gekennzeichnet wurden oder solch eine Auszeichnung erben.

```
:lang(en) { font-style: italic; }
```

Negations-Pseuoklasse

Selektor	Bezeichnung	Bedeutung
`E:not(s)`	Negations-Pseudoklasse	Element `E`, auf das der Selektor *s* nicht zutrifft

Tabelle 6.13: Negations-Pseudoklasse

Mit der Negations-Pseudoklasse `:not()` können Sie Elemente auswählen, auf die ein bestimmter Selektor *nicht* zutrifft. Diese Elemente werden mit einfachen Selektoren innerhalb der Klammern angegeben:

```
/* Wählt alle Elemente aus, die nicht das erste Kind sind. */
:not(:first-child) { font-size: .9em; }
```

Auch für diesen Selektor ist die Unterstützung bisher sehr schmal: Nur aktuelle KHTML- und Gecko-Browser können etwas damit anfangen.

Browserunterstützung

Kein Browser versteht derzeit jede der vorgestellten Pseudoklassen.

6.5.6 Pseudoelemente

Pseudoelemente stehen nicht im Quelltext eines Dokuments, werden von CSS aber so behandelt, als stünden sie darin. Ihre Namen beginnen in CSS 2 analog zu Pseudoklassen mit einem Doppelpunkt (`:`), in CSS 3 mit zwei Doppelpunkten (`::`). Die neuere Notation hilft, Pseudoelemente von Pseudoklassen zu

unterscheiden. Leider verstehen noch nicht alle aktuellen Browser die Schreibweise mit zwei Doppelpunkten. Sie sollten Pseudoelemente daher weiterhin mit nur einem Doppelpunkt notieren.

Selektor	Bezeichnung	Bedeutung
`E:first-letter`	Pseudoelement	erstes Zeichen eines Element `E`
`E:first-line`	Pseudoelement	erste Zeile eines Elements `E`

Tabelle 6.14: Selektion des ersten Zeichens und der ersten Zeile eines Elements

Unterschiedliche Interpretationen

`:first-letter` und `:first-line` gibt es schon seit CSS 1, und nahezu jeder Browser kann sie interpretieren, wenngleich unterschiedlich gut. Gerade `:first-letter` wirft einige Fragen auf. Führende Interpunktionszeichen werden im Allgemeinen mit dem ersten Buchstaben gemeinsam als eine Box behandelt. Für einige Fälle ist es allerdings unklar, schwer zu implementieren oder undefiniert, wo genau der erste Buchstabe zu suchen ist: Listenelementmarkierungen etwa für eine geordnete Liste (`ol`) können wahl- und teilweise dieser erste Buchstabe sein oder eben nicht. Führende Klammern, Ligaturen und Bilder (beziehungsweise deren Alternativtext) verursachen oft Probleme. Webautoren müssen die Inhalte daher sehr genau kennen, ehe sie `:first-letter` einsetzen können.

Werden sowohl `:first-line` als auch `:first-letter` eines Elements angesprochen, überschreibt `:first-letter` die Eigenschaften für `:first-line` erwartungsgemäß.

Selektor	Bezeichnung	Bedeutung
`E:before`	Pseudoelement	generierter Inhalt vor Element `E`
`E:after`	Pseudoelement	generierter Inhalt nach Element `E`

Tabelle 6.15: Generierter Inhalt

Generierter Inhalt

Die Pseudoelemente `:before` und `:after` sprechen keinen bestehenden Inhalt an, sondern erzeugen welchen. Sie werden in den meisten Fällen im Zusammenhang mit der Eigenschaft `content` verwendet (siehe Kapitel 7.12.1), um Inhalt vor und/oder nach dem tatsächlichen Inhalt zu generieren. So kann man in Browsern, die den Selektor verstehen, sehr einfach die über das Attribut `cite` angegebene Quelle eines Zitats anzeigen:

```
<blockquote
  cite="http://gutenberg.spiegel.de/knigge/umgang/umg3a01.htm">
  <p>Man soll es also dem Schriftsteller nicht übel ausdeuten,
    wenn er, verführt von ein wenig Geschwätzigkeit, von der
    Begierde, über irgendeine Materie allerlei Arten von
    Menschen seine Gedanken mitzuteilen, etwas drucken lässt,
```

```
    das nicht grade die Quintessenz von Weisheit, Witz,
    Scharfsinn und Gelehrsamkeit enthält.</p>
</blockquote>
```

Folgende Regel sorgt nun dafür, dass ein ausreichend CSS 2-fähiger Browser im Anschluss an das Zitat die Quelle anzeigt:

```
blockquote:after {
  content: "(Quelle: " att(cite) ")";
}
```

Kennzeichnung externer Links

In Verbindung mit einem der CSS 3-Attributselektoren lassen sich externe Links besonders einfach kennzeichnen:

```
a[href^="http://"]:before,
a[href^="https://"]:before {
  content: "\2197\00A0";
}
```

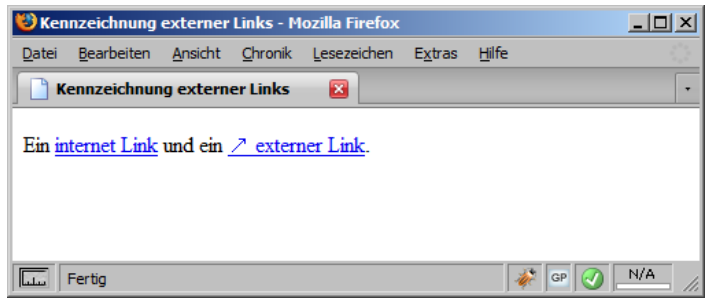


Abbildung 6.3: Das Pseudoelement `:before` können Sie einsetzen, um Pfeile vor externen Links darzustellen

Links, die zu Dokumenten der gleichen Domain führen, sind als relative Verweise notiert, entweder relativ zum Dokument oder relativ zum *Document Root*. Wenn Sie sich daran halten und absolute Verweise nur für Links nach außen verwenden, können Sie ausnutzen, dass der Wert des `href`-Attributs externer Links stets mit »http://« oder »https://« beginnt. Alle Browser, die den CSS 3-Attributselektor schon implementiert haben und zudem das Pseudoelement `:before` verstehen, zeigen vor externen Links einen Nordostpfeil an – ein für diesen Zweck in unterschiedlichen Varianten gern verwendetes Symbol.

Selektor	Bezeichnung	Bedeutung
`E:selection`	Elementfragment-Pseudoelement	Teil eines Elements `E`, der vom Nutzer markiert ist

Tabelle 6.16: Elementfragment-Pseudoelement

Mit dem Elementfragment-Pseudoelement `:selection` wird der gerade vom Nutzer markierte Text ausgewählt. Ein sinnvoller Einsatz mag dann entstehen, wenn das Farbschema der Seite sehr wahrscheinlich keinen guten Kontrast mehr zu den üblichen Farbmarkierungen bietet, etwa Weiß auf Blau.

```
p:selection {
  color: black;
  background-color: yellow;
}
```

Browserunterstützung

Pseudoelemente werden im Allgemeinen von allen modernen Browsern erkannt, allerdings gibt es einige Ausnahmen:

In vielen Gecko-Browsern lässt sich generierter Inhalt weder kopieren noch positionieren.

Das Elementfragment-Pseudoelement wird bisher nur von KHTML-Browsern (Konqueror und Safari) unterstützt.

6.5.7 Kombinatoren

Oft genügen einfache Selektoren nicht zur richtigen Auswahl. Abhilfe schaffen Kombinatoren, die dabei helfen, zusätzliche Übereinstimmungsbeschränkungen zu formulieren, um nur spezielle Elemente zu erfassen.

Kombinatoren dürfen nicht innerhalb der Negations-Pseudoklasse `:not()` verwendet werden.

Selektor	Bezeichnung	Bedeutung
`F E`	Nachfahrkombinator	Element `E`, das Nachfahre eines Elements vom Typ `F` ist
`F > E`	Kindkombinator	Element `E`, das Kind eines Elements vom Typ `F` ist
`F + E`	direkter Nachbarkombinator	Element `E`, das unmittelbar einem Geschwisterelement vom Typ `F` folgt
`F ~ E`	indirekter Nachbarkombinator	Element `E`, das einem Geschwisterelement vom Typ `F` folgt

Tabelle 6.17: Kombinatoren

Nachfahrkombinator

Der einfachste Kombinator ist der Nachfahrkombinator. Er ist Teil von CSS 1, besteht aus mindestens einem Leerzeichen und wählt alle Elemente aus, die irgendwo innerhalb eines anderen stehen. Die Regel

```
li p { margin: 0; }
```

formatiert alle Absätze innerhalb eines Listenpunkts.

Kindkombinator

Der Kindkombinator wählt nur Kindelemente aus.

```
/* Äquivalent zum Selektor 'body, head' */
html > * { margin: 0; }
```

Direkter Nachbarkombinator

Über den direkten Nachbarselektor ist es möglich, ein Element anzusprechen, das unmittelbar einem anderen auf gleicher Ebene folgt. Betrachten Sie folgendes XHTML-Beispiel:

```
<h2>Überschrift</h2>

<p>Erster Absatz</p>

<p>Zweiter Absatz</p>
```

Den ersten Absatz nach der Überschrift können Webautoren über folgende Regel formatieren:

```
h2 + p { margin-top: 0; }
```

Indirekter Nachbarkombinator

Der indirekte Nachbarselektor spricht alle Elemente auf der gleichen Ebene wie ein vorausgegangenes Element an, auch wenn sie nicht unmittelbar folgen.

Folgende Regel wird auf alle Absätze des oben angeführten XHTML-Beispiels angewandt.

```
h2 ~ p { margin: 0; }
```

Browserunterstützung

Alle Browser – Die vorgestellten Kombinatoren werden von allen Browsern in ihren aktuellen Versionen erkannt.

6.6 Vererbung und Initialwerte

Es gibt in CSS so viele Eigenschaften, dass man unmöglich alle für jedes Element setzen kann. CSS sieht daher vor, dass jedes Element automatisch für jede denkbare Eigenschaft einen Wert hat. Mit ihren Deklarationen ändern Webautoren also immer nur den Wert einer Eigenschaft und setzen nichts grundlegend Neues.

Falls der Webautor für ein Element keine explizite Deklaration vorgenommen hat, erfährt der Browser aus zwei Quellen, welchen Wert er auf ein Element anwenden muss: aus den Initialwerten und der Vererbung.

DEFINITION

Die CSS-Empfehlung bestimmt für fast jede Eigenschaft einen sogenannten ***Initialwert****, der vom Browser angenommen wird, wenn Sie nichts anderes angeben. Nur wenige Eigenschaften verfügen über keinen Initialwert, beispielsweise die Eigenschaft* `color`*, deren Initialwert den Einstellungen des Browsers überlassen bleibt.*

Da CSS für hierarchische Auszeichnungssprachen konzipiert wurde, bei denen jedes Element außer dem Wurzelelement innerhalb eines anderen liegt, können Eigenschaften entlang dieser Verschachtelung vererbt werden. Aus diesem Grund hat der Inhalt der `em`-Elemente im folgenden Beispiel eine Schrift derselben Schriftfamilie wie der ihrer Elternelemente `h1` und `p`.

```
<h1>Überschrift mit einem <em>betonten</em> Wort</h1>

<p>Absatz mit einem <em>betonten</em> Wort</p>
```

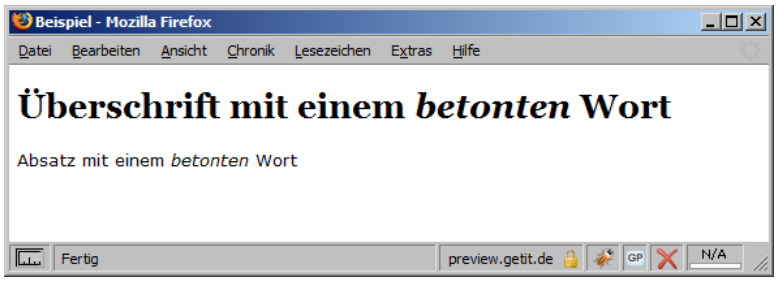


Abbildung 6.4: Das Element `em` erbt die Schriftdefinition ihres Elternelements. Lediglich der Kursivschnitt kommt hinzu

Vererbung hilft sehr dabei, effiziente Stylesheets zu schreiben. So genügt es meistens, Schriftart, Zeilenhöhe, Vorder- und Hintergrundfarbe für das Element `body` zu setzen – die Vererbungsregeln nehmen Ihnen die Arbeit ab, dies für spätere Elemente zu wiederholen.

Es gibt Ausnahmen, in denen die Vererbung nicht funktioniert oder browserübergreifend nicht oder schlecht implementiert ist. Links haben meistens eine andere Farbe als gewöhnlicher Fließtext (für gewöhnlich Blau für unbesuchte, Violett für besuchte Links), ebenso Formularelemente. Im Quirks Mode (siehe Kapitel 6.8.2 »Doctype Switching«) werden einige Schriftangaben nicht in Tabellenzellen hinein vererbt.

Zusammenfassende Eigenschaften

In einem Fall lässt sich die Vererbung implizit außer Kraft setzen, also ohne dass Sie einen neuen Wert angeben. CSS kennt zusammenfassende Eigenschaften, mit denen viele verwandte Eigenschaften in einer Deklaration gebündelt werden können. Die Eigenschaften

```
p {
  font-weight: bold;
  font-size: 2em;
  line-height: 1.5;
  font-family: Georgia;
}
```

können Sie wie folgt zusammenfassen:

```
p { font: bold 2em/1.5 Georgia; }
```

Das Kürzel `font` fasst allerdings nicht nur die vier oben genannten Eigenschaften zusammen, sondern auch noch die Eigenschaften `font-style` und `font-variant`, selbst wenn Sie sie nicht explizit hinschreiben! Geben Sie zu diesen Eigenschaften nichts an, werden sie automatisch auf ihren Initialwert gesetzt. So haben Überschriften in den meisten Browsern standardmäßig einen fetten Schriftschnitt. Mit der Deklaration `font: 3em "Palatino Linotype"`; heben Sie diesen Schriftschnitt jedoch auf, weil der Initialwert für `font-weight` eben `normal` ist und nicht `bold`.[10]

`inherit`

Über das Schlüsselwort `inherit` als Wert lässt sich Vererbung erzwingen. Damit lassen sich selbst solche Werte vererben, die das normalerweise nicht tun, `background-color` etwa. Beachten Sie, dass Sie `inherit` in zusammenfassenden Eigenschaften nicht mit anderen Werten zusammen verwenden dürfen.

6.7 Die Kaskade

Die vielen Möglichkeiten, CSS einzubinden, weisen schon auf eine Besonderheit der Sprache hin: Das »C« in CSS steht für »Cascading« (kaskadierend). Damit ist gemeint, dass mehrere Stylesheets zusammenwirken können, nacheinander oder parallel, selbst wenn sie konkurrierende Regeln enthalten.

Browserstylesheet

So ist das **Browserstylesheet** das erste Stylesheet, das auf eine Webseite angewandt wird. Darin stehen bestimmte Regeln, die sich zum größten Teil an den Initialwerten der CSS-Empfehlung anlehnen, aber auch häufig von ihnen abweichen – bei Schriftgrößen für Überschriften, Unterstreichungen für Links, Abständen für Formulare und Absätze und dergleichen mehr. Auch wenn

10 In CSS 3 werden Webautoren mit dem Schlüsselwort `initial` eine weitere Möglichkeit zum Aufheben der Vererbung bekommen. Doch ehe dies praxistauglich wird, vergehen wahrscheinlich noch ein paar Jahre.

nicht alle Browser tatsächlich ein in CSS geschriebenes Standardstylesheet verwenden, so lassen sich die mitgebrachten Formatierungsregeln doch größtenteils in CSS ausdrücken. Webautoren müssen sich der Tatsache bewusst sein, dass jeder Browser Standardeinstellungen mit sich bringt, die sich alle leicht voneinander unterscheiden. So erzeugt Opera den Abstand des Elements `body` zum Rand der Anzeigefläche mit der Eigenschaft `padding`, während Gecko und der Internet Explorer hierfür `margin` verwenden. Um den Abstand zu entfernen, müssen Webautoren daher beiden Eigenschaften den Wert `0` zuweisen und damit zwei unterschiedliche Browserstylesheets überschreiben.

Userstylesheet

Weiterhin werden zunehmend Userstylesheets eingesetzt. Dabei handelt es sich um CSS-Dateien, die sich Ihre Besucher erstellt haben, um bestimmte Vorlieben durchzusetzen, etwa eine bestimmte Schriftgröße oder eine Kombination aus Vorder- und Hintergrundfarben. Viele Browser bieten Konfigurationsmöglichkeiten, derartige Einstellungen ohne hinterlegtes CSS-Stylesheet vornehmen zu können. Opera bietet beispielsweise die Möglichkeit, die Unterstreichung von Links ein- oder auszuschalten sowie deren Farbe im besuchten und unbesuchten Zustand zu bestimmen. All diese nutzerspezifischen Einstellungen überschreiben die generellen Einstellungen im Browserstylesheet.

Autorenstylesheet

Schließlich können Sie auch Ihre Autorenstylesheets kaskadieren lassen, etwa indem Sie schon im Markup mehrere Stylesheets referenzieren oder mit der `@import`-Regel zusätzliche Stylesheets einbinden – all diese Stylesheets kaskadieren sozusagen zu einem Gesamtstylesheet.

Es liegt nahe, dass es durch Kaskade oft zu konkurrierenden Regeln kommt. So kann im Browserstylesheet die Schrift »Times New Roman« für `body` gesetzt sein, im Autorenstylesheet »Georgia« an der einen, »Arial« an der anderen Stelle und im Userstylesheet zudem »Verdana«. Selbst innerhalb eines einzigen Stylesheets oder gar innerhalb einer einzigen Regel können solche Situationen auftreten. Aber wovon hängt es ab, welche Deklaration gewinnt?

6.7.1 Reihenfolge

Um die Regeln zu bestimmen, die für ein Element angewandt werden sollen, muss sich der Browser verhalten, als arbeite er diese Reihenfolge ab:

1. Finde alle Deklarationen, die in diesem Ausgabemedium für das Element angewandt werden müssen.
2. Sortiere die Deklarationen aufsteigend nach Wichtigkeit:

   zuerst die Deklarationen des Browserstylesheets, dann die der Userstylesheets, dann die des Autors, dann die des Autors mit einem `!important` und zuletzt die des Userstylesheets mit einem `!important`.

DEFINITION

*Das **Schlüsselwort** `!important` kennzeichnet eine Deklaration als wichtiger gegenüber einer vergleichbaren Deklaration ohne dieses Schlüsselwort. Die Deklaration einer zusammenfassenden Eigenschaft als `!important` (z. B. `background: transparent none !important;`) ist äquivalent zur Deklaration aller ihrer untergeordneten Eigenschaften als `!important`.*

3. Sortiere alle Deklarationen nach Gewicht, entsprechend der Spezifität der zugehörigen Selektoren.

   Für jeden Selektor wird ein Wert berechnet, der die Relevanz des Selektors angibt, also dessen Durchsetzungsfähigkeit, wenn Sie so wollen. Diesen Wert bezeichnet man als »Spezifität«, der sich in Zahlenwerten ausdrücken lässt. Selektoren höherer Spezifität sind wichtiger als Selektoren geringerer Spezifität, solange kein `!important` im Spiel ist; der Selektor mit der höchsten Spezifität »gewinnt« und überschreibt die Regeln, deren Selektoren eine niedrigere Spezifität haben. Falls Ihr Stylesheet also nicht so funktioniert, wie Sie sich das vorstellen, eine Regel also anscheinend nicht angewandt wird, ist es wahrscheinlich, dass Sie sich die Spezifität der Selektoren genauer anschauen müssen.

Berechnung der Spezifität

Die Spezifität eines Selektors besteht aus vier einzelnen Werten, die nacheinander notiert werden: *A*, *B*, *C* und *D*.

1. *A* wird für CSS-Regeln auf 1 gesetzt, die durch ein Attribut `style` im Quelltext definiert sind, ansonsten auf 0.
2. *B* ist zdie Anzahl der `id`-Attribute eines Selektors.
3. *C* ist die Anzahl aller anderen Attribute einschließlich Klassen und Pseudoklassen innerhalb des Selektors.
4. *D* ist die Anzahl der Elementnamen und Pseudoelemente, die der Selektor beinhaltet.

Die Spezifität ergibt sich aus der Verknüpfung der vier Zahlen. In CSS 2.1 werden die Beispiele aus Tabelle 6.18 aufgeführt, die nach Spezifität aufsteigend sortiert sind:

Selektor	Wertzuweisung	Spezifität
`*`	A=0 B=0 C=0 D=0	0,0,0,0
`li`	A=0 B=0 C=0 D=1	0,0,0,1
`li:first-line`	A=0 B=0 C=0 D=2	0,0,0,2
`ul li`	A=0 B=0 C=0 D=2	0,0,0,2

Tabelle 6.18: Berechnung der Spezifität von Selektoren

Selektor	Wertzuweisung	Spezifität
`ul ol+li`	A=0 B=0 C=0 D=3	0,0,0,3
`h1 + *[rel=up]`	A=0 B=0 C=1 D=1	0,0,1,1
`ul ol li.red`	A=0 B=0 C=1 D=3	0,0,1,3
`li.red.level`	A=0 B=0 C=2 D=1	0,0,2,1
`#x34y`	A=0 B=1 C=0 D=0	0,1,0,0
`style=""`	A=1 B=0 C=0 D=0	1,0,0,0

Tabelle 6.18: Berechnung der Spezifität von Selektoren (Fortsetzung)

4. Sortiere die Deklarationen in der vorgegebenen Reihenfolge.

   Haben zwei Regeln dieselbe Gewichtung, denselben Ursprung und dieselbe Spezifität, erhält die jeweils zuletzt angegebene den Vorrang.

Der Leser gewinnt

Aus der Reihenfolge ist übrigens eines abzulesen: Der Leser hat stets das letzte Wort. Manche Autoren wollen das nicht einsehen und versuchen, ihren Lesern um jeden Preis etwas aufzudrängen (Schriftgrößen oder Breitenangaben sind die Klassiker). Profis wissen, dass sie nie gegen ihre Leser arbeiten können und sollen, sondern immer nur mit ihnen. Wer CSS verwendet, hat automatisch Koautoren.

Beispiel

Ein abschließendes Beispiel soll die Berechnung der Spezifität noch einmal verdeutlichen.

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

<html xmlns="http://www.w3.org/1999/xhtml"
  lang="de" xml:lang="de">
  <head>
    <meta http-equiv="Content-Type"
      content="text/html; charset=utf-8" />
    <title>Spezifität von Selektoren</title>

    <style type="text/css">
      em      { color: blue;  }
      em.foo  { color: green; }
      p em    { color: red;   }
    </style>
  </head>

  <body>
    <p>Absatz mit einem
      <em class="foo">betonten</em> Wort</p>
  </body>
</html>
```

Listing 6.4: Spezifität von Selektoren

Elemente vom Typ `em` sollen in blauer Schriftfarbe angezeigt werden, `em`-Elemente der Klasse `foo` in Grün, solche in Absätzen `p` in Rot. In welcher Schriftfarbe zeigen Browser das `em`-Element nun an? Wissen Sie die Antwort? Falls nicht, werfen Sie einfach einen Blick auf die Spezifität der Selektoren:

Selektor	Wertzuweisung	Spezifität
`em`	A=0 B=0 C=0 D=1	0,0,0,1
`em.foo`	A=0 B=0 C=1 D=1	0,0,1,1
`p em`	A=0 B=0 C=0 D=2	0,0,0,2

Tabelle 6.19: Der Klassenselektor hat die höhere Spezifität

Es gewinnt folglich die zweite Regel, nicht etwa die nachfolgende, wie Sie womöglich vermutet haben. Der betonte Text erscheint in grüner Schrift.

6.8 Browserkompatibilität

Die Unterstützung vieler CSS-Eigenschaften lässt heute leider noch zu wünschen übrig. Dabei sind es nicht die alten, heute überholten Browsergenerationen, die die meisten Probleme bereiten, denn sie verdienen aufgrund ihrer geringen Verbreitung keine Beachtung mehr und machen es ziemlich einfach, CSS vor ihnen zu verbergen. Problematisch sind fehlerhafte Browser mit großer Verbreitung, vor allem der Internet Explorer 6, der immer noch – je nach herangezogener Statistik – von 40 bis 50 Prozent aller Nutzer verwendet wird. Würden Webautoren nur die Teile von CSS verwenden, die auch der IE 6 versteht, bewegten sie sich irgendwo zwischen CSS 1 und CSS 2.

Dieser Abschnitt hilft Ihnen, mit gängigen Browserproblemen zurechtzukommen.

6.8.1 Browserunterstützung

In der Praxis kann das Zusammenspiel zwischen verschiedenen Elementen und Eigenschaften einen nicht zu unterschätzenden Einfluss auf die Darstellung haben. Fehler treten oft erst durch das Zusammenwirken mehrerer Eigenschaften an unterschiedlichen Elementen auf.

Aufgrund der rasanten Entwicklung am Browsermarkt und der Vielzahl der verfügbaren Browser ist es nicht möglich, alle Eigenschaften auf jedem Browser in jeder Version durchzutesten – es ist vielmehr davon auszugehen, dass es mittlerweile neue Versionen der getesteten Browser gibt, in denen die beschriebenen Fehler nicht mehr auftauchen, wenn Sie dieses Buch in der

Hand halten. Ich habe mich auf die in Tabelle 6.20 aufgeführten Browser beschränkt. Dennoch erheben die Tests keinen Anspruch auf Vollständigkeit; sie sind vielmehr lediglich als kleine Hilfestellung zu verstehen – wenn Sie kritische Eigenschaften und Werte verwenden, sind eigene Tests unumgänglich!

Symbol	Browser	Kürzel
	Internet Explorer 6	IE 6
	Internet Explorer 7	IE 7
	Internet Explorer 8	IE 8
	Firefox 3	
	Opera 8, Opera 9, Opera 9.5	
	Safari 3	
	Konqueror 3.5, Konqueror 4	

Tabelle 6.20: Browser, deren CSS-Fähigkeiten für dieses Buch getestet wurden

6.8.2 Doctype Switching

Viele Webseiten sind noch für vor einigen Jahren aktuelle, heute allerdings nur noch sehr selten verwendete Browser wie die 5er-Versionen des Internet Explorers geschrieben. Diese Browser zeigen jedoch an vielen Stellen eine mangelhafte und nicht standardkonforme CSS-Umsetzung. Ganz gleich, ob Entwickler sich der Fehler als solche bewusst waren, es blieb ihnen nichts anderes übrig, als Webseiten entgegen den Standards auf diese Browser hin zu konzipieren, um das gewünschte Layout umzusetzen. Das führte dazu, dass Webseiten auf damals aktuellen Browsern wie gewünscht aussahen; in heutigen modernen Browsern jedoch würden sie zwar standardkonform, aber falsch im Sinne der ursprünglichen Layoutvorstellungen des Designers dargestellt.

Quirks und Standards Compliance Mode

Die Browserhersteller mussten damit rechnen, dass niemand aktuelle Browserversionen verwenden würde, wenn alte Webseiten darin nicht mehr wie gewohnt aussähen. Daher wurden zwei unterschiedliche Darstellungsarten eingeführt: Der erste Modus, bekannt als *Quirks Mode* oder *Compatible Mode*, stellt eine Webseite wie alte, inkompatible Browser dar; der zweite Modus,

bekannt als *Standards Compliance Mode*, bemüht sich um eine Darstellung gemäß den W3C-Empfehlungen. Darüber hinaus unterteilen aktuelle Gecko- und Opera-Versionen den letzten Modus in einen *Almost Standards Mode* und einen *Full Standards Mode*.

Der Internet Explorer 8 verfügt ebenfalls über drei Darstellungsmodi: Der Quirks Mode stellt eine Webseite wie alte, inkompatible Internet Explorer dar; dabei entspricht der Quirks Mode des IE 8 dem des IE 7. Daneben verfügt der IE 8 über zwei standardkonforme Modi: einen herkömmlichen IE 7-Standards-Modus, der dem des IE 7 mit all seinen Fehlern entspricht, und einen neuen (IE 8-)Standards-Modus, der die stark verbesserte, komplett erneuerte Rendering Engine des IE 8 verwendet.

Die technischen Details der Umsetzungen sind in den Browserdokumentationen zu finden.

Microsoft und die Mozilla-Entwickler haben sich entschieden, über die Dokumenttyp-Deklaration zu differenzieren. Andere Browser zogen nach. Die Unterscheidung funktioniert in Firefox, Safari und Opera nach folgendem Schema: Wenn das Dokument keine Dokumenttyp-Deklaration hat oder eine aus einer Positivliste, dann gehe in den Quirks Mode. In allen anderen Fällen gehe in den Standards Compliance Mode.

Im Internet Explorer sieht die Regel im Allgemeinen so aus: Wenn die erste Zeile nicht mit einer Dokumenttyp-Deklaration beginnt, die in der Positivliste steht, gehe in den Quirks Mode, es sei denn, es wurde ein Systembezeichner angegeben.

Dieses Verfahren wird *Doctype Switching* oder *Doctype Sniffing* genannt.

Die einzelnen Browser entscheiden nach folgenden Regeln, mit welchem Darstellungsmodus sie eine Webseite darstellen:

- Der **Internet Explorer 6** schaltet in den Quirks Mode, wenn keine Dokumenttyp-Deklaration angegeben wird, ebenso bei HTML 4 und 4.01 bei Angabe der Transitional- und Frameset-DTD, wenn kein System Identifier angegeben wird. In allen anderen Fällen, also auch bei allen XHTML-Dokumenttyp-Deklarationen, schaltet IE 6 in den Standards Compliance Mode.

  Leider zeigt sich an dieser Stelle wieder einer der vielen IE-Bugs: Der Internet Explorer erwartet die Dokumenttyp-Deklaration immer in der ersten Zeile des Dokuments. Wenn zum Beispiel eine XML-Deklaration ein XHTML-Dokument einleitet, rendert der IE im Quirks Mode. Da die XML-Deklaration bei XHTML-Dokumenten nicht vorgeschrieben ist, lassen Webautoren sie aus diesem Grund häufig weg. Der Artikel ***[Silver 2006]***

führt eine Tabelle auf und dokumentiert die Unterschiede zwischen Quirks und Standards Compliance Mode.

- Der **Internet Explorer 7** verhält sich so wie der IE 6, allerdings haben die Microsoft-Entwickler den XML-Deklaration-Bug behoben.

- Der **Internet Explorer 8** fällt seine Entscheidung über den Darstellungsmodus analog zu seinem Vorgänger. Allerdings verfügt er über zwei standardkonforme Modi. Webautoren haben die Möglichkeit, über ein `meta`-Element anzugeben, welchen der beiden Modi sie auf ihrer Website angewendet wissen wollen, den IE 7-Standards-Modus oder den IE 8-Standards-Modus. Das `meta`-Element sieht wie folgt aus: `<meta http-equiv="X-UA-Compatible" content="IE=8" />`, wobei der Wert des `content`-Attributs entweder `IE=7` oder `IE=8` lautet.

- **Opera** verhält sich so wie der Internet Explorer ab Version 7. ***[Opera]*** führt die Unterschiede zwischen beiden Modi auf.

- Komplizierter wird es bei **Firefox**-Browsern. Die aktuellen Versionen unterscheiden nur bei Dokumenten, die mit dem MIME-Typ `text/html` ausgeliefert werden, zwischen verschiedenen Modi. Dokumente, die als `application/xhtml+xml` oder mit einem anderen XML-MIME-Typ ausgeliefert werden, werden immer im Standards Compliance Mode dargestellt. Weitere Informationen und eine ausführliche Übersicht dazu, wann in den Quirks und wann in den Full Standards oder Almost Standards Mode geschaltet wird, finden Sie in den Artikeln ***[Baron 2004]*** und ***[Trebing 2002]***.

Fragwürdiges Konzept

Aufgrund der Fehler, die Browserhersteller in der Vergangenheit gemacht haben, war eine Einführung unterschiedlicher Darstellungsmodi unumgänglich. Ob es sinnvoll ist, gerade über Art und Notation der verwendeten Dokumenttyp-Deklaration den Darstellungsmodus für ein Dokument festzulegen, ist sehr umstritten. Schließlich ist diese Art der Differenzierung alles andere als intuitiv. Die HTML 4-Empfehlung des W3C etwa ist ein standardkonformes Dokument, dennoch wird es nur im Quirks Mode dargestellt. Man kann durchaus bezweifeln, dass die Autoren dieser Website damit gerechnet haben, dass die Darstellung dieses Dokuments in einem Modus erfolgt, der die standardkonforme Darstellung umgeht und dessen Rendering Engine mit voller Absicht an der einen oder anderen Stelle fehlerhaft ist.

Empfehlung

Uns Webautoren bleibt nichts anders übrig, als das Beste aus diesen Voraussetzungen zu machen. Ich persönlich arbeite ausschließlich im Standards Compliance Mode. Mit Fehlern haben Webautoren in allen Modi zu kämpfen, aber wenn Sie im standardkonformen Modus arbeiten, wissen Sie wenigstens, dass die Browser daran schuld sind und nicht Sie – und die Fehler, die auftreten, sind in den meisten Fällen verhältnismäßig einfach zu beheben oder zu umgehen.

6.8.3 Der Grund für viele IE-Bugs: hasLayout

hasLayout ist ein proprietäres Konzept (siehe ***[Chao, Rudel 2007]***) des Internet Explorers bis zur Version 7, das bestimmt, wie Elemente ihren Inhalt darstellen, wie sie mit anderen Elementen interagieren und wie sie auf Anwendungs- oder Benutzeraktionen reagieren und sie übertragen. Einige Elemente haben von sich aus »Layout«, bei anderen ist es erforderlich, eine CSS-Eigenschaft anzuwenden, die »Layout« für das fragliche Element verursacht.

Das hasLayout-Konzept stellt Webentwickler vor eine Vielzahl von Problemen. »Layout« hat ungewöhnliche und schwer vorhersehbare Auswirkungen auf die Darstellung von Boxen – und ebenso Folgen für deren Nachfahrenelemente. Ob ein Element »Layout« hat (oder auch nicht), kann unter anderem Folgendes nach sich ziehen:

- Viele Fehler im Zusammenhang mit Floatierung (siehe Kapitel 7.6 »Ausrichtung und Elementfluss«) treten auf, weil ein Element nicht »Layout« hat.
- Die betroffenen Boxen behandeln Eigenschaften vollkommen unterschiedlich, abhängig davon, ob sie »Layout« haben oder nicht.
- Verschiedene Probleme bei der Darstellung von Listen, dem Positionieren von Hintergrundbildern und beim Einsatz von Scripts hängen mit »Layout« – beziehungsweise dessen Fehlen – zusammen.

IE 6 und 7

Folgende Elemente haben von sich aus »Layout« (nicht alle davon werden im Rahmen dieser Einführung behandelt):

- `html`, `body`
- `table`, `tr`, `th`, `td`
- `img`
- `hr`
- `input`, `button`, `select`, `textarea`, `fieldset`, `legend`
- `iframe`, `embed`, `object`, `applet`
- `marquee`

Anderen Elementen wird »Layout« zugewiesen, sobald Sie (unter anderem) mindestens eine der folgenden Deklarationen für dieses Element vornehmen:

- `position: absolute`
- `float: left`, `float: right`
- `display: inline-block`

- `width`: beliebiger Wert außer `auto`
- `height`: beliebiger Wert außer `auto`

Mit der Einführung des Internet Explorers 7 kamen weitere Deklarationen hinzu: IE 7

- `overflow: hidden`, `overflow: scroll`, `overflow: auto`
- `overflow-x: hidden`, `overflow-x: scroll`, `overflow-x: auto`
- `overflow-y: hidden`, `overflow-y: scroll`, `overflow-y: auto`
- `position: fixed`
- `min-width`: beliebiger Wert
- `max-width`: beliebiger Wert
- `min-height`: beliebiger Wert
- `max-height`: beliebiger Wert

Für Inline-Elemente gilt: `width` und `height` lösen »Layout« lediglich im Quirks Mode aus.

Es gibt keine Möglichkeit, »Layout« für ein Element explizit zu entfernen. Stattdessen müssen Sie die entsprechenden Eigenschaften auf ihren Initialwert zurücksetzen. Im Allgemeinen ist »Layout« jedoch erwünscht, oder anders formuliert: Viele IE-Bugs können Sie beheben, indem Sie bei einem Element »Layout« verursachen. Ihre Aufgabe ist es, den Weg zu finden, der im konkreten Fall die wenigsten »Nebenwirkungen« verursacht. Im weiteren Verlauf dieses Buchs werden Sie einige Beispiele finden.

6.8.4 CSS-Hacks für den Internet Explorer

Wie bereits erwähnt, sind es nicht die Browser, die CSS überhaupt nicht verstehen, mit denen Webautoren täglich zu kämpfen haben, sondern die Browser, die CSS fehlerhaft interpretieren: Von unlesbaren Texten und unzugänglichen Inhalten über inaktive Links bis hin zum Browserabsturz kann in solchen Fällen alles passieren. Die Vielzahl an Browsern und deren zum Teil eklatant unterschiedliche CSS-Implementierungen verbieten es, ein einfaches Stylesheet ungetestet anzubieten in der Hoffnung, die Webseite sehe schon auf allen Browsern so aus wie auf dem Entwicklungsbrowser. Deshalb müssen Webautoren ihre Stylesheets entschärfen und dürfen jedem Browser nur das zumuten, was er entweder richtig oder gar nicht interpretiert. Um dies zu bewerkstelligen, setzen Webautoren sogenannte CSS-Hacks ein, also Regeln bestimmter Syntax, die entweder nur spezielle Browser bedienen oder CSS vor diesen verbergen.

CSS-Hacks sind oft nicht besonders intuitiv; sie verlangen nach gut dokumentierenden Kommentaren, regelmäßiger Kontrolle in den aktuellen Browserversionen und einem vertrauten Umgang mit der Syntax von CSS. Darüber hinaus lässt sich nahezu jeder Filter mit einem anderen kombinieren. Das »Austricksen« unterschiedlicher Browser ist manchen eine Qual, anderen eine allzu große Verführung.

Webautoren müssen bei jedem Projekt erneut die Balance finden zwischen aufgeräumtem Code und einer möglichst breiten Kompatibilität – keine leichte Aufgabe, sogar für Profis ... Es empfiehlt sich ein regelmäßiger Blick auf die Seite der CSS-Hacks[11], um halbwegs auf dem aktuellen Stand zu bleiben. Die Seite wird gut gepflegt und ist immer ein ergiebiger Ausgangspunkt für gegenwärtige und zukünftige Recherchen.

Die große Anzahl aller bekannten CSS-Hacks macht es leider unmöglich, sie alle im Rahmen dieser Einführung zu behandeln. Exemplarisch folgen an dieser Stelle einige Filter für den Internet Explorer, den Browser, mit dessen unterschiedlichen Problemen in seinen verschiedenen Varianten Webautoren am häufigsten zu kämpfen haben.

CSS-Hacks für den Internet Explorer

IE 6 Der IE 6 interpretiert den Selektor `* html` als einfaches `html`. Dies nennt man den »Star-HTML-Hack«. Ein Element `strong` unterhalb eines Elements `html` unterhalb eines beliebigen Elements kann es nicht geben, da das `html`-Element keine Vorfahrenelemente besitzt. Vernünftige Browser verstehen Regeln daher nicht, die durch diesen Selektor eingeleitet werden – der IE ignoriert den Stern und wendet die Regel an. Listing 6.5 zeigt die Anwendung des Hacks.

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

<html xmlns="http://www.w3.org/1999/xhtml"
  lang="de" xml:lang="de">
  <head>
    <meta http-equiv="Content-Type"
      content="text/html; charset=utf-8" />
    <title>Browsertest</title>

    <style type="text/css">
      /* <![CDATA[ */
      * html strong {
          display: none;
      }
```

11 *http://css-discuss.incutio.com/?page=CssHack*

```
      /* ]]> */
    </style>
  </head>

  <body>
    <p>Sie verwenden <strong>nicht</strong>
      den Internet Explorer ab Version 5.</p>
  </body>
</html>
```

Listing 6.5: Die Anwendung des Star-HTML-Hacks

IE 7

Eine Variante des Star-HTML-Hacks betrifft den IE ab Version 7. Da das Element `html` das Stammelement eines Webdokuments ist, kann es auch keine Geschwisterelemente haben. Dennoch verstehen IE 7 und IE 8 den eigentlich unsinnigen Selektor `* + html`; in diesen Browsern selektiert der Universalselektor auch Kommentare und Dokumenttyp-Deklarationen, die er als Geschwisterelemente des Elements `html` ansieht.

```
* + html strong {
  display: none;
}
```

Leider interpretiert auch Opera ab Version 7.5 diesen Selektor, allerdings nur bei gleichzeitiger Verwendung einer XML-Deklaration. Die Alternative `*:first-child + html` zeigt dieses Verhalten nicht und spricht ausschließlich den IE an.

```
*:first-child + html strong {
  display: none;
}
```

IE 8

Um CSS lediglich an den IE 8 auszuliefern, können Sie sich einen Kommentarfehler zunutze machen. Folgende Regel ist eigentlich auskommentiert; der Kommentar wird erst nach dem Deklarationsblock ordnungsgemäß geschlossen. IE 8 interpretiert die Regel trotzdem:

```
/*/ strong {
  display: none;
} /* */
```

Zum Zeitpunkt der Drucklegung dieses Buchs war der IE 8 nur als Betaversion verfügbar. Möglicherweise haben die Entwickler den Fehler in der Zwischenzeit behoben. Werfen Sie in diesem Fall einen Blick auf die Seite *http://css-discuss.incutio.com/?page=IE8* und schauen Sie, ob mittlerweile ein neuer CSS-Hack bekannt ist.

Progressive Enhancement

Im Englischen hat sich der Begriff *Progressive Enhancement* für CSS-Anwendungen eingebürgert, die alte Browser zwar nicht verstehen, neuere aber zum Nutzen des Lesers umsetzen. Diese Aufwertung, und damit das Einsetzen der neuen Features, verfolgt zwei Ziele:

- Die Ergonomie von Websites soll verbessert werden, und sei es nur für die 20 oder 30 Prozent der Nutzer, deren Browser das CSS in vollem Umfang verstehen.
- Sowohl die Browserhersteller als auch die Autoren der W3C-Empfehlungen sind darauf angewiesen, dass CSS in der Praxis eingesetzt wird, um Rückmeldungen von Nutzern und Webautoren zu erhalten, damit sie sinnvoll an CSS und dessen Implementierung weiterarbeiten können.

Darüber hinaus bietet modernes CSS so viele faszinierende Möglichkeiten, dass es einfach Freude bereitet, sie einzusetzen und mit den zahlreichen Funktionen zu spielen.

7 Eigenschaften und Werte

7.1 Werte

Die folgenden Abschnitte sind alle ähnlich aufgebaut. Ich stelle einzelne Eigenschaften und deren erlaubte Werte vor. Dazu folgt dann jeweils eine Erläuterung, zumeist mit einem oder mehreren Beispielen.

Bei den meisten Eigenschaften habe ich einige mögliche Werte in Gruppen zusammengefasst und nur die jeweiligen Schlüsselwörter aufgeführt. Was diese Schlüsselwörter bedeuten, erläutere ich in diesem Abschnitt.

7.1.1 <Farbe>

In CSS können Sie Farben auf zwei Arten angeben: als RGB-Farbwert oder über ein Schlüsselwort.

RGB-Farbwert

Ein Monitor arbeitet mit drei Spektralfarben: Rot, Grün und Blau. Jedes Pixel der Desktop-Fläche kann in einer der drei Spektralfarben leuchten. Die einzelnen Pixel nehmen Sie jedoch nicht mehr wahr, da sie klein genug sind, um sich optisch zu vermischen, sodass unterschiedliche Farben und Helligkeiten

wahrgenommen werden. RGB-Werte setzen sich aus Rot-, Grün- und Blauanteilen zusammen. Pro Kanal sind 255 Werte möglich (0 bis 255). So ergibt die CSS-Deklaration `color: rgb(255,0,0);` ein reines Rot. Teilen Sie jedem Kanal den höchsten Wert zu (jeweils 255), erhalten Sie die Farbe Weiß.

RGB-Farbwerte lassen sich in hexadezimaler oder funktionaler Notation angeben.

Hexadezimale Notation

Das Format eines RGB-Werts in hexadezimaler Notation ist ein #, unmittelbar gefolgt von sechs oder drei Hexadezimalzeichen. Hexadezimale Werte haben nicht nur die zehn Ziffern des Dezimalsystems, sondern bestehen aus 16 Ziffern. Zusätzlich zu den Ziffern 0 bis 9 gibt es hier noch die Ziffern A bis F, die den dezimalen Zahlen 10 bis 15 entsprechen. So wird aus der Dezimalzahl 255 die Hexadezimalzahl FF. Diese Umrechnung ist gewöhnungsbedürftig. Eine Angabe wie beispielsweise `#FFFFFF` bedeutet nichts anderes als FF Anteile Rot, FF Anteile Grün und FF Anteile Blau; sie ergibt folglich ein reines Weiß. Zwischen Groß- und Kleinschreibung wird nicht unterschieden. Anstatt `#FFFFFF` können Sie auch `#ffffff` schreiben.

Der Farbwert wird im Allgemeinen mit sechs Stellen angegeben, jeweils zwei für Rot, Grün und Blau, jedoch lassen sich die Angaben für jede einzelne Farbe (also paarweise) zusammenfassen, wenn sie jeweils gleich sind. Aus `#FFDDEE` wird dann `#FDE`, aus `#FFFFFF` wird `#FFF`.

Funktionale Notation

Das Format eines RGB-Werts in der funktionalen Notation ist die Zeichenkette `rgb(`, gefolgt von einer Liste mit durch Kommata voneinander getrennten Einträgen der drei numerischen Werte (entweder drei ganzzahlige Werte oder drei Prozentwerte), gefolgt von `)`. Der ganzzahlige Wert 255 entspricht `100%` und `F` oder `FF` in der hexadezimalen Notation. Die folgenden Angaben ergeben jeweils reines Weiß.

```
color: #FFF;
color: #FFFFFF;
color: rgb(255,255,255);
color: rgb(100%, 100%, 100%);
```

Schlüsselwörter

CSS 2 kennt Schlüsselwörter für 16 Farbnamen. Der aktuelle CSS 2.1-Arbeitsentwurf führt noch eine weitere auf. Orange wurde anscheinend so häufig fälschlicherweise angegeben, dass die CSS-Arbeitsgruppe sich erbarmt hat, die Farbe aufzunehmen, auch wenn der Farbwert nicht so recht ins Schema passt. Tabelle 7.1 führt alle 17 Farbnamen und deren Farbwerte in hexadezimaler und funktionaler Notation auf.

Farbname		Farbwert	
CSS-Farbname	**deutsche Übersetzung**	**hexadezimal**	**funktional**
black	Schwarz	#000000	rgb(0, 0, 0)
gray	Grau	#808080	rgb(128, 128, 128)
silver	Silber	#C0C0C0	rgb(192, 192, 192)
white	Weiß	#FFFFFF	rgb(255, 255, 255)
purple	Violett	#800080	rgb(128, 0, 128)
fuchsia	Fuchsia	#FF00FF	rgb(255, 0, 255)
maroon	Kastanienbraun	#800000	rgb(128, 0, 0)
red	Rot	#FF0000	rgb(255, 0, 0)
olive	Olivgrün	#808000	rgb(128, 128, 0)
yellow	Gelb	#FFFF00	rgb(255, 255, 0)
green	Grün	#008000	rgb(0, 128, 0)
lime	Limonengelb	#00FF00	rgb(0, 255, 0)
navy	Navyblau	#000080	rgb(0, 0, 128)
blue	Blau	#0000FF	rgb(0, 0, 255)
teal	Blaugrün	#008080	rgb(0, 128, 128)
aqua	Aquamarinblau	#00FFFF	rgb(0, 255, 255)
orange	Orange	#FFA500	rgb(255, 165, 0)

Tabelle 7.1: Liste aller in CSS 2.1 spezifizierten Farbnamen

Anwenderdefinierte Farben

Bei anwenderorientierten Farben handelt es sich um Schlüsselwörter, die sich auf die grafische Umgebung des Benutzers, das heißt dessen Systemfarben beziehen. Damit lässt sich eine Website individuell an das vom Nutzer eingerichtete Look & Feel anpassen, das den Gewohnheiten und speziellen Bedürfnissen des Nutzers womöglich mehr entgegenkommt, als es ein Webautor vermag.

Tabelle 7.2 führt alle anwenderdefinierten Farbnamen auf, die in CSS 2.1 definiert sind.

CSS-Farbname	Erklärung
ActiveBorder	aktiver Fensterrahmen
ActiveCaption	aktiver Fenstertitel

Tabelle 7.2: Liste aller in CSS 2.1 spezifizierten, anwenderdefinierten Farbnamen

CSS-Farbname	Erklärung
AppWorkspace	Hintergrundfarbe einer MDI-Anwendung
Background	Desktop-Hintergrundfarbe
ButtonFace	Schriftfarbe für dreidimensionale Anzeigeelemente
ButtonHighlight	helle Farbe für dreidimensionale Anzeigeelemente
ButtonShadow	Schattenfarbe für dreidimensionale Anzeigeelemente
ButtonText	Text auf Schaltflächen
CaptionText	Text in Überschriften, Size Boxes und Pfeilfeld des Rollbalkens
GrayText	grau unterlegter, deaktivierter Text
Highlight	Elemente, die in einem Steuerelement markiert sind
HighlightText	Text von Elementen, die in einem Steuerelement markiert sind
InactiveBorder	nicht aktiver Fensterrahmen
InactiveCaption	nicht aktivierter Fenstertitel
InactiveCaptionText	Textfarbe in einem nicht aktivierten Fenstertitel
InfoBackground	Hintergrundfarbe von Tooltipps
InfoText	Textfarbe von Tooltipps
Menu	Menühintergrund
MenuText	Menütext
Scrollbar	grauer Bereich der Rollbalken
ThreeDDarkShadow	dunkler Schatten für dreidimensionale Anzeigeelemente
ThreeDFace	Schriftfarbe für dreidimensionale Anzeigeelemente
ThreeDHighlight	Farbe für die Hervorhebung von dreidimensionalen Anzeige-elementen
ThreeDLightShadow	helle Farbe für dreidimensionale Anzeigeelemente
ThreeDShadow	dunkler Schatten für dreidimensionale Anzeigeelemente
Window	Fensterhintergrund
WindowFrame	Fensterrahmen
WindowText	Text in Fenstern

Tabelle 7.2: Liste aller in CSS 2.1 spezifizierten, anwenderdefinierten Farbnamen (Fortsetzung)

7.1.2 <Integer>

Das Schlüsselwort `<Integer>` steht für eine ganze Zahl aus einer oder mehreren Ziffern von 0 bis 9 (zum Beispiel 0, 17, 4911).

7.1.3 <Zahl>

Das Schlüsselwort `<Zahl>` steht für eine reelle Zahl. Diese kann entweder ein `<Integer>` sein oder aus einer Ziffer oder mehreren Ziffern bestehen, gefolgt von einem Punkt (.) und einer weiteren Ziffer oder mehreren weiteren Ziffern (zum Beispiel `0` oder `3.14`). Sowohl den ganzen als auch den reellen Zahlen kann ein Vorzeichen vorangestellt werden.

7.1.4 <Länge>

Längenangaben stellen horizontale oder vertikale Maße dar und bestehen entweder aus einem Schlüsselwort oder einem optionalen Vorzeichen (+ oder –), einer `<Zahl>` und einer Einheit. Die Einheit ist nur hinter der Zahl 0 optional. Einige Eigenschaften erlauben negative Längenangaben. Man unterscheidet zwischen absoluten und relativen Längenangaben.

Absolute Längen

Absolute Längen sollten Sie nur dann verwenden, wenn die physikalischen Gegebenheiten des Ausgabemediums genau bekannt sind, also zum Beispiel die Ausmaße eines Blatt Papiers.

Punkt (pt) und Pica (pc)

Punkt und Pica sind typografische Maßeinheiten, die besonders im Printdesign verwendet werden. 1 Punkt entspricht 1/72 Zoll (1 Zoll = 2,54 Zentimeter), 1 Pica entspricht 12 Punkten.

Bildschirmmaße berechnen sich jedoch nicht in Punkten oder Pica, sondern in Pixeln. Angaben in beiden Einheiten müssen folglich entsprechend umgerechnet werden. Dies geschieht anhand der ppi-Zahl (*pixel per inch*, Pixel pro Zoll), die jedoch von System zu System unterschiedlich sein kann. Auf Windows-Systemen sind es 96 ppi, wenn Sie kleine Schriftarten eingestellt haben, und 120 ppi, wenn Sie große Schriftarten verwenden. Auf Macintosh-Betriebssystemen sind es nur 72 ppi. Eine 9 Punkt große Schrift würde auf einem Mac 9 Pixel groß sein, auf einem Windows-Betriebssystem hingegen 12 oder 15 Pixel. Dies bedeutet, dass ein auf einem Windows-System noch gut lesbarer Text auf einem Macintosh im schlimmsten Fall nicht mehr zu entziffern ist. Aus diesem Grund eignen sich beide Einheiten ausschließlich bei der Verwendung in Stylesheets für die Ausgabe auf Printmedien.

Zentimeter (cm) und Millimeter (mm)

Auch Angaben in cm oder mm müssen für die Bildschirmdarstellung in Pixelwerte umgerechnet werden. Kann das nicht exakt erfolgen, muss gerundet werden. Für diese Einheiten gilt im Prinzip dasselbe wie für Punkt und Pica: Die tatsächliche Länge kann sich von System zu System deutlich unterscheiden, daher sind beide Einheiten für die Darstellung auf dem Screen ebenfalls nicht wirklich zu gebrauchen.

Schlüsselwörter

Es gibt sieben Schlüsselwörter für absolute Größen:

- `xx-small`: winzig
- `x-small`: sehr klein
- `small`: klein
- `medium`: mittel
- `large`: groß
- `x-large`: sehr groß
- `xx-large`: riesig

Die CSS 2-Empfehlung sieht vor, dass sich die Größen bei der Ausgabe auf Computerbildschirmen jeweils um den Faktor 1.2 voneinander unterscheiden.

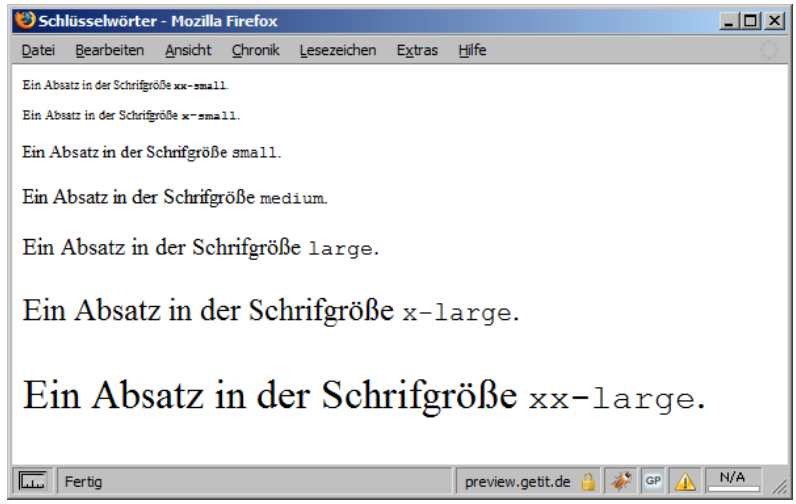


Abbildung 7.1: Darstellung der Schlüsselwörter im Firefox; die Darstellung in anderen aktuellen Browsern ist vergleichbar

Relative Längen

Relative Längen eignen sich für die Bildschirmdarstellung besser als absolute Einheiten, da sie »von Natur aus« skalierbar sind. Das bedeutet, Nutzer können die Ausgabe an ihre eigenen Bedürfnisse anpassen, etwa die Schrift vergrößern, wenn sie durch eine Sehschwäche so sehr eingeschränkt sind, dass sie Texte in Standardschriftgröße auf dem Bildschirm nicht oder nur sehr mühevoll lesen können.

X-Höhe (ex)

Die Größe der Einheit `ex` wird durch die Höhe des Kleinbuchstabens »x« der verwendeten Schrift bestimmt.

Geviertbreite (em)

Die Einheit `em` ist gleich dem berechneten Wert der Eigenschaft `font-size` des Elements, auf das sie angewendet wird. Wird `em` für `font-size` selbst verwendet, bezieht es sich auf die Schriftgröße des übergeordneten Elements; ist keine solche vorgegeben, ist die Angabe `1em` gleich der vom Benutzer eingestellten Schriftgröße. Diese entspricht im Allgemeinen 16 Pixeln. Die Regel

```
h1 { line-height: 1.5em; }
```

bedeutet, dass die Zeilenhöhe von Elementen vom Typ `h1` um 50 Prozent größer ist als deren Schriftgröße. Hingegen legt

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

fest, dass die Schriftgröße von Elementen vom Typ `h1` um 50 Prozent größer ist als die berechnete (vererbte) Schriftgröße ihrer Elternelemente.

Schlüsselwörter

Es gibt zwei Schlüsselwörter für relative Größen:

- `larger`: größer als das Elternelement
- `smaller`: kleiner als das Elternelement

Die Schrift wird relativ zur Schriftgröße des Elternelements um den Faktor 1.2 vergrößert beziehungsweise verkleinert.

Pixel (px)

Pixel nehmen eine Sonderstellung ein: Sie verhalten sich nicht relativ zur eingestellten Schrift oder anderen (vererbten) Größen, sondern sind relativ zur Auflösung des Anzeigegeräts. Verändern Sie die Auflösung Ihres Monitors

beispielsweise von SVGA (Super Video Graphics Array, 800 x 600 Bildpunkte) auf UXGA (Ultra Extended Graphics Array, 1.600 x 1.200 Bildpunkte), erscheinen alle Pixellängen nur noch halb so groß.

Berechnete Werte

Untergeordnete Elemente erben nicht die für ihre Vorfahrenelemente angegebenen relativen Werte, sondern die berechneten Werte. Folgendes Beispiel soll dies demonstrieren:

```
body {
  font-size: 20px;
  text-indent: 2em;
}

p { font-size: 30px; }
```

Das Element `p` erbt den berechneten Wert der Eigenschaft `text-indent`, wird also nicht um 60 Pixel eingerückt (2 x 30 Pixel), sondern nur um 40 Pixel (2 x 20 Pixel).

7.1.5 <Prozent>

Prozentwerte bestehen aus einem optionalen Vorzeichen (+ oder –), einer `<Zahl>` und der Einheit `%`. Sie sind immer relativ zu einem anderen Wert und verhalten sich in etwa so wie Werte mit der Einheit `em`.

7.1.6 <URI>

Das Schlüsselwort `<URI>` steht für die Adressierung einer Ressource, zumeist einer Grafik. Das Format ist `url(`, gefolgt vom eigentlichen URI, gefolgt von `)`. Der URI kann ein relativer oder absoluter Verweis sein und in einfachen (`'`) oder doppelten (`"`) Anführungszeichen stehen.

Relative Verweise gehen vom Stylesheet aus, nicht von dem Dokument, das das Stylesheet einbindet. Die Regel

```
body { background-image: url(images/bg.png); }
```

in einem Stylesheet mit dem URI *http://www.example.org/stylesheets/default.css* verweist auf ein Hintergrundbild mit der Adresse *http://www.example.org/stylesheets/images/bg.png*.

7.1.7 <Zeichenkette>

Zeichenketten bestehen aus fast beliebigen Zeichen, Wörtern oder Sätzen. Sie werden immer in doppelten oder einfachen Anführungszeichen notiert. Innerhalb von doppelten Anführungszeichen darf kein weiteres doppeltes Anführungszeichen vorkommen, sonst muss es mit einem Escape-Zeichen (\) maskiert werden. Entsprechendes gilt auch für einfache Anführungszeichen.

```
li.linkliste a[href^="http://"]:before,
li.linkliste a[href^="https://"]:before {
  content: "Externer Link: ";
}
```

7.1.8 <inherit>

Mit dem Wert `inherit` können Sie explizit angeben, dass eine Eigenschaft eines Elements den (berechneten) Wert des Elternelements erben soll.

7.2 Farben und Hintergründe

Die in diesem Abschnitt vorgestellten Eigenschaften geben Ihnen Möglichkeiten an die Hand, Ihre Websites farblich aufzupeppen. Farben sind für eine Webseite von großer Bedeutung. Zum einen beeinflussen Farben die optische Erscheinung wesentlich, zum anderen sind sie wichtig, um die Übersichtlichkeit und den Aufbau der Seite zu unterstützen.

7.2.1 Vordergrundfarbe: Die Eigenschaft color

DEFINITION

*Die **Eigenschaft** `color` bestimmt die Vordergrundfarbe eines Elements, die als Schriftfarbe verwendet wird.*

Werte	<Farbe>, `inherit`
Initialwert	browserabhängig
Vererbung	ja
Anwendung	global

Beispiel:

```
p.warnung {
  color: red;
  background-color: white;
}
```

Browserunterstützung

Die Eigenschaft `color` wird von allen modernen Browsern unterstützt mit folgender Einschränkung:

IE ignoriert bis Version 7 den Wert `inherit`.

7.2.2 Hintergrundfarbe: Die Eigenschaft background-color

DEFINITION

*Die **Eigenschaft** `background-color` bestimmt die Hintergrundfarbe eines Elements.*

Werte	`<Farbe>`, `transparent`, `inherit`
Initialwert	`transparent`
Vererbung	nein
Anwendung	global

Beispiel:

```
p.warnung {
  color: red;
  background-color: white;
}
```

Das Schlüsselwort `transparent` legt fest, dass ein Element einen transparenten Hintergrund hat, darunterliegende Farben und Inhalte somit durchscheinen.

Die Eigenschaften `color` und `background-color` sollten Sie immer zusammen angeben. Angenommen, Sie deklarieren für ein Element eine Hintergrundfarbe, die in Ton und Helligkeit der Farbe ähnelt, die ein Besucher Ihrer Seite in seinem Browser für die Vordergrundfarbe eingestellt hat. Der Nutzer würde dann nichts mehr lesen können. Achten Sie daher darauf, dass Vorder- und Hintergrundfarben stets ausreichend kontrastieren.

Browserunterstützung

Die Eigenschaft `background-color` wird von allen modernen Browsern unterstützt mit folgender Einschränkung:

IE ignoriert bis Version 7 den Wert `inherit`.

7.2.3 Hintergrundbild: Die Eigenschaft background-image

DEFINITION

*Die **Eigenschaft** `background-image` gibt das Hintergrundbild eines Elements an.*

Werte	`<URI>`, `none`, `inherit`
Initialwert	`none`
Vererbung	nein
Anwendung	global

Beispiel:

```
body {
  color: red;
  background-color: white;
  background-image: url(redbox.gif);
}
```

Kontrast

Sie sollten stets zusätzlich eine Hintergrundfarbe im Grundton des Bilds angeben, um extreme Kontraste zwischen dem Zustand vor und nach dem Laden der Grafik zu vermeiden. Transparente Teile des Hintergrundbilds lassen die Hintergrundfarbe oder den Hintergrund eines Vorfahrenelements durchscheinen.

Kachelung

Im Allgemeinen werden Hintergrundbilder gekachelt, also über die gesamte Anzeigefläche nebeneinandergelegt. Dieses Verhalten ist über die Eigenschaft `background-repeat` steuerbar.

Keine Skalierbarkeit

Hintergrundbilder sind nicht skalierbar. So ist es nicht oder nur durch eine überdimensionierte Grafik möglich, bei allen Auflösungen die volle Fensterbreite mit nur einer Instanz der Grafik auszufüllen.

Browserunterstützung

Die Eigenschaft `background-image` wird von allen modernen Browsern unterstützt mit folgender Einschränkung:

IE ignoriert bis Version 7 den Wert `inherit`. Zudem positionieren beide Browser das Hintergrundbild nur dann korrekt, wenn sie »Layout« haben.

7.2.4 Wiederholung eines Hintergrundbilds: Die Eigenschaft background-repeat

DEFINITION

*Die **Eigenschaft** `background-repeat` bestimmt, ob und in welche Richtung das Hintergrundbild eines Elements wiederholt werden soll.*

Werte	`repeat`, `no-repeat`, `repeat-x`, `repeat-y`, `inherit`
Initialwert	`repeat`
Vererbung	nein
Anwendung	global

Beispiel:

```
body {
  color: red;
  background-color: white;
  background-image: url(redbox.gif);
  background-repeat: no-repeat;
}
```

Geben Sie den Wert `repeat` an, wird das Hintergrundbild über die volle Anzeigefläche gekachelt. Bei `repeat-x` wird das Bild nur in x-Richtung wiederholt, bei `repeat-y` entsprechend nur in y-Richtung. Geben Sie `no-repeat` an, wird das Hintergrundbild nicht wiederholt, sondern nur ein Mal eingefügt. An welcher Stelle genau das Bild eingefügt wird, bestimmen Sie über die Eigenschaft `background-position`.

Browserunterstützung

Die Eigenschaft `background-repeat` wird von allen modernen Browsern unterstützt mit folgender Einschränkung:

Bis Version 7 interpretiert der IE den Wert `inherit` wie `repeat`.

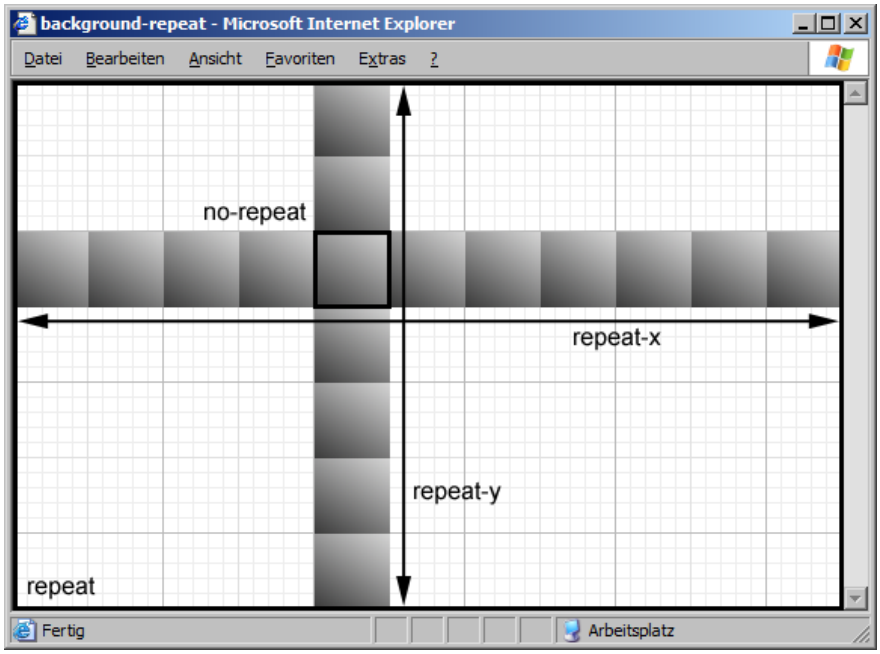


Abbildung 7.2: Die Eigenschaft `background-repeat`

7.2.5 Scrolleigenschaft eines Hintergrundbilds: Die Eigenschaft background-attachment

DEFINITION

*Die **Eigenschaft** `background-attachment` gibt an, ob ein Hintergrundbild mit der Seite mitrollen oder auf der Stelle stehen bleiben soll.*

Werte	`scroll`, `fixed`, `inherit`
Initialwert	`scroll`
Vererbung	nein
Anwendung	global

Beispiel:

```
body {
  color: red;
  background-color: white;
  background-image: url(redbox.gif);
```

```
    background-repeat: no-repeat;
    background-attachment: scroll;
  }
```

Der Wert `scroll` sorgt dafür, dass das Hintergrundbild mit der Seite mitscrollt. Geben Sie `fixed` an, bleibt es beim Scrollen der Seite auf der Stelle stehen. Dieses Verhalten ist auch als »Wasserzeicheneffekt« bekannt. Dabei werden fixierte Hintergrundbilder stets am Browserfenster ausgerichtet und nicht am Element, für das diese Eigenschaft deklariert ist. So ist es möglich, dass eine fixierte Hintergrundgrafik scheinbar nicht angezeigt wird (oder erst beim Scrollen), weil sie sich außerhalb der Anzeigefläche des Elements befindet. Abhilfe schafft hierbei eine Wiederholung des Bilds mit `background-repeat`.

Browserunterstützung

Die Eigenschaft `background-attachment` wird von allen modernen Browsern unterstützt mit folgenden Einschränkungen:

IE bis Version 6 interpretiert den Wert `fixed` nur am Element `body` korrekt. Ferner ignoriert er den Wert `inherit`.

IE 7 erkennt den Wert `fixed` korrekt, ignoriert aber den Wert `inherit`.

7.2.6 Position eines Hintergrundbilds: Die Eigenschaft background-position

DEFINITION

*Die **Eigenschaft** `background-position` bestimmt die Position eines Hintergrundbilds.*

Werte	`<Prozent>` oder `<Prozent> <Prozent>`, `<Länge>` oder `<Länge> <Länge>`, `top`, `center` oder `bottom` zusammen mit `left`, `center` oder `right`, `inherit`
Initialwert	`0% 0%`
Vererbung	nein
Anwendung	Blockelemente

Es gibt verschiedene Möglichkeiten, die Position eines Hintergrundbilds zu bestimmen.

Erste Möglichkeit: Prozentwerte (<Prozent> oder <Prozent> <Prozent>)

Notieren Sie zwei Werte, definiert der erste den horizontalen und der zweite den vertikalen Abstand, jeweils gemessen von der oberen linken Ecke des Elements; geben Sie nur einen Wert an, wird lediglich die horizontale Position bestimmt, die vertikale entspricht dem Wert `50%`.

Die so positionierte Hintergrundgrafik wird jedoch nicht mit ihrer oberen linken Ecke an diesem Punkt platziert, sondern entsprechend den angegebenen Werten nach oben links verschoben. So liegt ein mit der Deklaration `background-position: 100% 100%;` nach unten rechts positioniertes Bild nicht außerhalb der Anzeigefläche, sondern genau in der unteren rechten Ecke, da die Grafik selbst um ihre vollen Ausmaße verschoben wird. Abbildung 7.3 illustriert dieses Verhalten anhand der Deklaration `background-position: 30% 60%;`. Die Linien kreuzen sich genau an dieser Koordinate. Sie sehen, wie das Bild nach oben links verschoben wird.

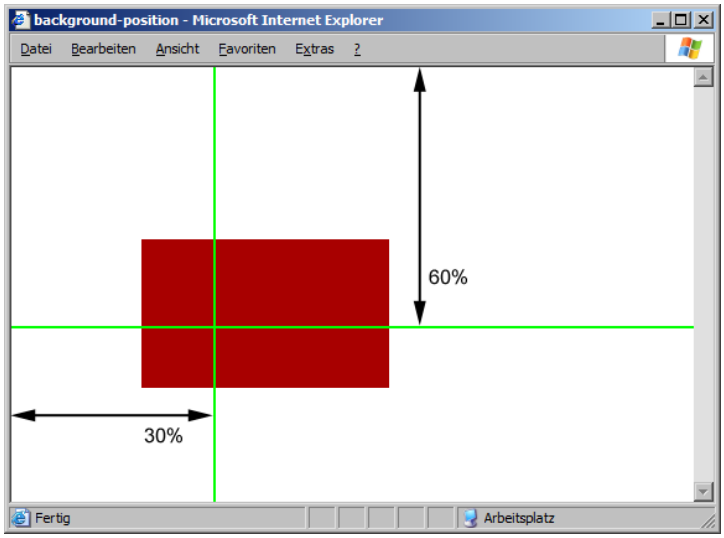


Abbildung 7.3: Positionierung eines Hintergrundbilds

Zweite Möglichkeit: Angabe von Längen (<Länge> oder <Länge> <Länge>)

Die Positionierung mit Längen funktioniert so wie die Positionierung mit Prozentwerten, allerdings wird die Grafik nicht um ihre Ausmaße verschoben. Im folgenden Beispiel wird das Hintergrundbild 100 Pixel in x-Richtung (nach rechts) und 200 Pixel in y-Richtung (nach unten) ausgerichtet.

Beispiel:

```
body {
  color: red;
  background-color: white;
  background-image: url(redbox.gif);
  background-repeat: no-repeat;
  background-attachment: scroll;
  background-position: 100px 200px;
}
```

Dritte Möglichkeit: Schlüsselwörter (top, center oder bottom zusammen mit left, center oder right)

Die Werte `top` und `left` entsprechen der vertikalen beziehungsweise horizontalen Position `0%`. `center` entspricht in beiden Richtungen `50%`. `right` und `bottom` entsprechen horizontal beziehungsweise vertikal `100%`.

Ein Bild, das zum Beispiel in der rechten unteren Ecke des Fensters platziert werden soll, entspricht entweder `100% 100%`, `bottom right` oder `right bottom`.

Schlüsselwörter dürfen mit Prozent- oder Längenangaben gemischt werden, wobei `left` und `right` in solchen Kombinationen nur als erster Wert und `top` und `bottom` nur als zweiter Wert notiert werden dürfen.

Browserunterstützung

Die Eigenschaft `background-position` wird von allen modernen Browsern unterstützt mit einer Ausnahme:

IE behandelt den Wert `inherit` bis Version 7 wie die Angabe `top left`.

7.2.7 Hintergrund: Die zusammenfassende Eigenschaft background

DEFINITION

*Die **Eigenschaft** `background` fasst die Einstellung der einzelnen Hintergrundeigenschaften (`background-color`, `background-image`, `background-repeat`, `background-attachment` und `background-position`) zusammen. Sie setzt zunächst die einzelnen Hintergrundeigenschaften auf ihre Initialwerte und weist ihnen dann die explizit in der Deklaration angegebenen Werte zu.*

Das Beispiel können Sie wie folgt abkürzen:

```
body {
  color: red;
  background: white url(redbox.gif) no-repeat
    scroll 100px 200px;
}
```

Browserunterstützung

Alle Browser – Die Eigenschaft `background` wird von allen modernen Browsern unterstützt, allerdings gibt es einige Einschränkungen, die bei den einzelnen `background-*`-Eigenschaften aufgeführt sind.

7.3 Schrift

7.3.1 Schriftfamilie: Die Eigenschaft font-family

DEFINITION

*Die **Eigenschaft** `font-family` gibt die Schriftfamilie (Schriftart) an, die zur Darstellung textuellen Elementinhalts verwendet werden soll.*

Der Begriff »Schriftfamilie« kommt aus der Typografie und bezeichnet eine Gruppe zusammengehörender Schriften (Schriftschnitten und Schriftstilen) mit unterschiedlichen Breiten, Strichstärken (leicht, normal, fett, …) und Zeichenlagen (normal, kursiv, …).

Werte	`<Schriftfamilie>`, `<generische Schriftfamilie>`, `inherit`
Initialwert	browserabhängig
Vererbung	ja
Anwendung	global

`<Schriftfamilie>` steht für den Namen einer konkreten Schriftfamilie, zum Beispiel Courier, Courier New, Verdana, Arial, Trebuchet MS oder Times New Roman. Enthalten Schriftfamiliennamen Leerzeichen, sollten sie in Anführungszeichen notiert werden.

Generische Schriftfamilien stellen Gruppierungen von »typografisch verwandten« Schriften dar und dienen Webautoren dazu, den Stil der Schrift festzulegen, wenn sie keine konkrete(n) Schriftfamilie(n) angeben oder diese auf

dem System des Nutzers nicht installiert ist beziehungsweise sind. Es gibt insgesamt fünf generische Familien: `serif`, `sans-serif`, `cursive`, `fantasy` und `monospace`. Abbildung 7.4 beschreibt die Schriftfamilien und zeigt Beispiele.

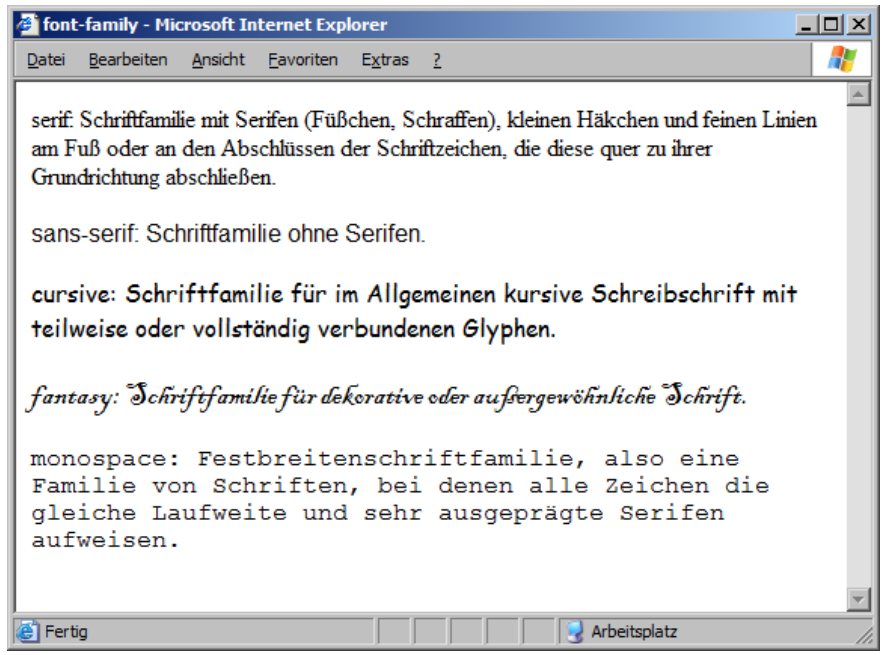


Abbildung 7.4: Die Eigenschaft `font-family`: Beispiele für die Schriftfamilien `serif`, `sans-serif`, `cursive`, `fantasy` und `monospace`

Die Frage nach der richtigen Schriftfamilie

Jetzt kennen Sie zwar die Eigenschaft, mit der Sie eine Schriftfamilie (oder mehrere) angeben, aber die grundlegenden Fragen müssen Sie sich noch immer stellen: Welche Schriftfamilie sollten Sie für welches Element wählen? Welche Schriftfamilie ist am besten zu lesen?

Grundsätzlich gilt: Sie können jede Schriftfamilie angeben, die Sie möchten. Sie können sogar eine beliebig große Anzahl von Schriftfamilien angeben. Aufgrund der Angaben

Verbreitete Schriftdeklarationen

```
body { font-family: Verdana, Helvetica, sans-serif; }

h1 { font-family: Georgia, "Times New Roman", Times, serif; }

code { font-family: "Courier New", Courier, monospace; }
```

würde ein CSS-fähiger Browser versuchen, den Elementinhalt von Überschriften der ersten Ordnung in der Schriftfamilie Georgia darzustellen. Sollte Georgia auf dem System nicht installiert sein, sucht er nach Times New Roman; findet er auch diese Schriftfamilie nicht, wählt er die Schriftfamilie, die der Benutzer auf seinem System standardmäßig für die generische Schriftfamilie `serif` angegeben hat beziehungsweise die im Browser voreingestellt ist.

Die Wahl der richtigen Schriftfamilie sollte gut überlegt sein. So sollten Sie sich zunächst fragen, ob es wirklich notwendig ist, überhaupt eine Schriftfamilie zu definieren. Aus zwei Gründen könnte dies überflüssig sein:

1. Jeder Browser stellt Text in einer voreingestellten Standardschriftfamilie dar. Man sollte davon ausgehen können, dass diese im Allgemeinen gut lesbar ist.

   Leider ist das nicht immer der Fall. Die auf Windows-Betriebssystemen voreingestellte Schriftfamilie Times New Roman versagt vor allem bei längeren Fließtexten oder kleinen Schriftschnitten.

2. Jeder Browser ermöglicht es Nutzern, eigene Schriftfamilien anzugeben, im einfachsten Fall über die allgemeinen Einstellungen des Browsers.

   Leider nutzt kaum jemand diese Möglichkeit, nicht zuletzt deshalb, weil das Gros der Nutzer nichts davon weiß oder nicht bereit ist, den Browser nach dessen Vorlieben zu konfigurieren.

Geben Sie daher eine Schriftfamilie an, von der Sie wissen, dass sie die Lesbarkeit Ihrer Website im Vergleich zur voreingestellten Standardschriftfamilie verbessert. Die im vorausgehenden Beispiel angegebenen Kombinationen haben sich in der Praxis bewährt. Nutzer, die in der Lage sind, eine Schriftart zu finden, die die Lesbarkeit darüber hinaus erhöht, wissen in den meisten Fällen auch, wie sie sie so einbinden, dass die Einstellungen des Autors ihre Angaben nicht überschreiben – im Allgemeinen per Userstylesheet in Verbindung mit dem Schlüsselwort `!important`.

Mit oder ohne Serifen?

Die Frage, ob man für Fließtext auf Webseiten eher eine Schriftfamilie mit oder ohne Serifen wählen sollte, wird schon so lange diskutiert, wie es Typografie (für die Darstellung auf Computerbildschirmen) gibt, und die Antwort darauf ist wohl eher eine individuelle Entscheidung als eine allgemeine Regel. Im Druck werden im Allgemeinen Serifenschriften verwendet, da sich vor allem durch die Betonung der Grund- und Mittellinie durch die Serifen, die das Auge führen, eine bessere Lesbarkeit ergibt. Serifenlose Schriften werden dagegen eher für kurze Texte und Überschriften eingesetzt. Für die Darstellung auf dem Bildschirm gehen Webautoren generell umgekehrt vor: serifenlose Schriften für lange Texte, Serifenschriften über Überschriften. Auf einem

(flackernden) Monitor können Sie nicht mehr davon ausgehen, dass die Schrift so klar und deutlich dargestellt wird wie auf dem Papier; die Führung des Auges kann dadurch schnell verloren gehen. Wichtiger sind große Zeilenabstände und kurze Zeilenlängen oder zumindest ein fließendes Design, das es Nutzern ermöglicht, Schriftgrößen zu skalieren und Zeilenlängen durch Verkleinern des Browserfensters auf eine lesbare Länge zu kürzen. Bei vielen Webagenturen ist Verdana für Fließtexte die Schriftfamilie der Wahl.

Browserunterstützung

Die Eigenschaft `font-family` wird von allen modernen Browsern unterstützt mit folgender Einschränkung:

IE ignoriert bis Version 7 den Wert `inherit`.

7.3.2 Schriftstil: Die Eigenschaft font-style

DEFINITION

*Die **Eigenschaft** `font-style` gibt an, ob eine normale oder eine schräg laufende Schriftauszeichnungsart gewählt wird.*

Werte	`italic, oblique, normal, inherit`
Initialwert	`normal`
Vererbung	ja
Anwendung	global

Der Wert `italic` bestimmt eine Schrift aus der angegebenen Schriftfamilie, die als »italic« (echt kursiv) gekennzeichnet ist, deren Glyphen also speziell kursiv entwickelt wurden. Falls eine solche Schrift nicht verfügbar ist, wird eine Schrift gewählt, die als »oblique« gekennzeichnet ist.

Der Wert `oblique` bestimmt eine Schrift aus der angegebenen Schriftfamilie, die als »oblique« gekennzeichnet ist. Häufig handelt es sich dabei um eine Schrift, die durch das elektronische Neigen einer normalen Schrift entstanden ist, wodurch die Lesbarkeit im Allgemeinen nicht so gut ist wie bei echt kursiven Schriften.

In der Realität unterscheiden sich Schriften in den Stilen `oblique` und `italic` innerhalb einer Schriftfamilie zumeist nicht oder kaum erkennbar.

Browserunterstützung

Die Eigenschaft `font-style` wird von allen modernen Browsern unterstützt mit folgenden Einschränkungen:

IE ignoriert bis Version 7 den Wert `inherit`. Zudem berechnen beide Browser die Breite von Inline-Blöcken falsch, wenn dessen Schrift kursiv gesetzt ist. Das kann zu überbreiten Elementen und daraus resultierenden Verschiebungen im Layout führen. Eine Lösung für dieses Problem finden Sie in Kapitel 6.4.3.

Alle Browser – Die Behandlung des Werts `oblique` entspricht mit unterschiedlichen Abstufungen nicht vollständig den Vorgaben der W3C-Empfehlung.

7.3.3 Schriftvariante: Die Eigenschaft font-variant

DEFINITION

*Die **Eigenschaft** `font-variant` gibt an, ob Normalschrift oder eine Schrift mit Kapitälchen angezeigt wird.*

Werte	`small-caps`, `normal`, `inherit`
Initialwert	`normal`
Vererbung	ja
Anwendung	global

Geben Sie den Wert `small-caps` an, wird die Schrift mit Kapitälchen angezeigt, wobei die Kleinbuchstaben ähnlich wie Großbuchstaben aussehen, aber etwas kleiner sind und andere Proportionen haben. Steht in der Schriftfamilie keine eigene Kapitälchenschrift zur Verfügung, simulieren Browser eine solche, indem sie im Allgemeinen eine normale Schrift verwenden und die Kleinbuchstaben durch verkleinerte Großbuchstaben ersetzen.

Browserunterstützung

Die Eigenschaft `font-variant` wird von allen modernen Browsern unterstützt mit folgender Einschränkung:

IE ignoriert bis Version 7 den Wert `inherit`.

7.3.4 Schriftgewichtung: Die Eigenschaft font-weight

DEFINITION

*Die **Eigenschaft** font-weight gibt die Gewichtung (Schriftdicke) der verwendeten Schrift an.*

Werte	100, 200, 300, 400, 500, 600, 700, 800, 900, normal, bold, bolder, lighter, inherit
Initialwert	normal
Vererbung	ja
Anwendung	global

Die Werte 100 bis 900 bilden eine Sequenz von »sehr dünn« bis »sehr dick«. Es kommt auf das Benutzerprogramm an, wie diese Werte dargestellt werden. In der Praxis gibt es aber bis heute kaum eine Schriftart, die in neun verschiedenen Gewichtungen existiert.

Der Wert normal entspricht dem Wert 400, der Wert bold dem Wert 700.

Der Wert bolder verschiebt die Schriftdicke relativ zum Elternelement (beziehungsweise zum berechneten vererbten Wert) um einen Wert nach oben, lighter analog um einen Wert nach unten.

Browserunterstützung

Die Eigenschaft font-weight wird von allen modernen Browsern unterstützt mit folgenden Einschränkungen:

IE ignoriert bis Version 7 den Wert inherit.

Alle Browser – Die meisten Browser kennen bei den meisten Schriftfamilien nur zwei Schriftdicken. Die Werte 100 bis 500 entsprechen dabei dem Wert normal, die Werte 600 bis 900 dem Wert bold. Wenn Sie eine Schriftdicke angeben, die in der entsprechenden Schriftfamilie nicht enthalten ist, hat der Browser zwei Möglichkeiten: Entweder er ignoriert die Angabe und nimmt einfach das Nächstliegende, oder er versucht, selbst etwas zu simulieren. Bevor Sie für font-weight also etwas anderes als normal oder bold angeben, vergewissern Sie sich, dass die vorgeschlagenen Schriften diese Schnitte auch enthalten und einigermaßen häufig sind.

7.3.5 Schriftgröße: Die Eigenschaft font-size

DEFINITION

*Die **Eigenschaft** `font-size` gibt die Schriftgröße eines Elements an.*

Werte	`<Länge>`, `<Prozent>`, `inherit`
Initialwert	`medium`
Vererbung	ja
Anwendung	global

Die Frage nach der richtigen Schriftgröße

Die Frage nach der richtigen Schriftgröße gehört wie die Frage nach der richtigen Schriftart zu den ersten und fundamentalsten in der Typografie. Dabei gibt es so viele Meinungen wie Möglichkeiten:

- Pessimisten sagen, die richtige Schriftgröße könne es nicht geben, da das, was für den einen richtig ist, für den anderen falsch ist.
- Puristen sagen, die richtige Schriftgröße sei die, die der Nutzer in seinem Browser eingestellt hat.
- Designer sagen, die richtige Schriftgröße sei eine solche, die das Gesamtbild der Seite prägt und sich geschmeidig in das Designkonzept der Website integriert.
- Pragmatiker sagen, die richtige Schriftgröße sei die, die auf den meisten Webseiten zu finden ist.
- Optimisten sagen, die richtige Schriftgröße sei die, die auf den meisten Webseiten zu finden ist, wobei Nutzer, denen sie zu klein oder zu groß ist, schon in der Lage seien, sie nach ihren Wünschen zu skalieren.

Einflussfaktoren

Viele Faktoren beeinflussen die Schriftgröße auf dem Bildschirm. Was auf einem 17-Zoll-TFT mit XGA-Auflösung (1.024 x 768) schön und übersichtlich aussieht, kann auf einem 21-Zoll-TFT mit UXGA-Auflösung (1.600 x 1.200) unlesbar klein und auf einem 14-Zoll-Notebook mit SVGA-Auflösung (800 x 600) klobig und viel zu groß wirken. Hinzu kommt, dass dieselbe Schriftgrößenangabe bei unterschiedlichen Schriftfamilien keinesfalls gleich wirkt. Viele serifenlose Schriften wirken größer als vergleichbare Serifenschriften und »zerfallen« bei großen Schriftgraden dadurch, dass die Buchstaben zu dünn wirken und zu weit auseinanderstehend scheinen. Abbildung 7.5 demonstriert dies anhand der Schriftgröße 16 px und den Schriftfamilien Verdana, Arial und Times New Roman.

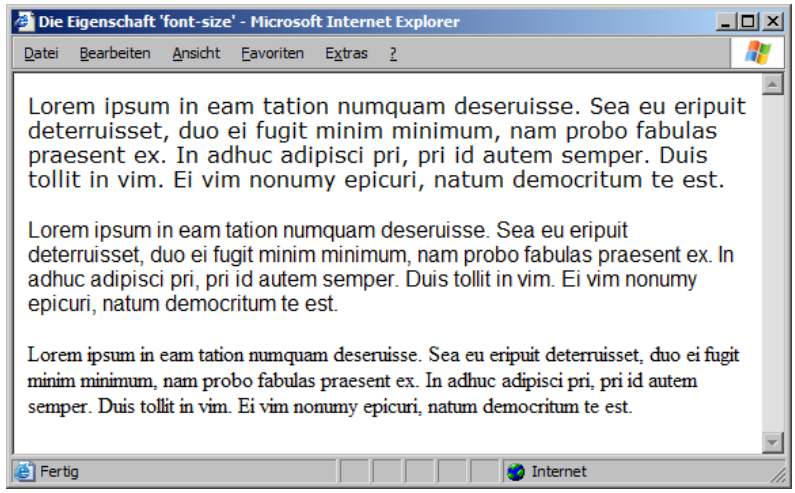


Abbildung 7.5: Die Schriftfamilien Verdana, Arial und Times New Roman wirken bei einer Schriftgröße von 16 px unterschiedlich groß

Sehbehinderte Menschen

Viel wichtiger als technische Faktoren ist jedoch der Nutzer mit seiner individuellen Sehkraft, der vor dem Ausgabegerät sitzt. Ältere Menschen haben im Allgemeinen größere Schwierigkeiten, Texte in kleiner Schrift zu lesen, als Jugendliche oder junge Erwachsene. Aber auch Normalsichtige haben oftmals nach Stunden des Lesens auf einem flackernden Monitor Probleme bei der Lesbarkeit von längeren Textpassagen. Wer sich schon mal dabei ertappt hat, wie er nach vorne gebeugt angestrengt auf einen viel zu kleinen Text starrt, wird wissen, was ich meine; wie viel angenehmer ist es doch, entspannt und zurückgelehnt einen Text zu lesen, dessen ausreichend große Schriftgröße diese angenehme Lesehaltung ermöglicht.

Die einfachste Lösung ist, dem Anwender die Wahl der für ihn richtigen Schriftgröße zu überlassen, also keine Angabe zur Schriftgröße vorzunehmen, zumindest nicht für Fließtexte. Daraus ergeben sich jedoch zwei Probleme:

1. Viele Anwender wissen nicht, wie sie die Schriftgröße ihres Browsers skalieren.
2. Die im Browser eingestellte Standardschriftgröße unterscheidet sich von der Schriftgröße, die auf den meisten Websites zu sehen ist. Das Gros aller Websites verwendet einen reduzierten Schriftgrad. Belassen Sie die Schriftgröße bei den Standardeinstellungen des Browsers, erscheint der Text Ihrer Website im Vergleich zu den meisten anderen Sites ungewohnt groß. Dieses Phänomen ist auch als das »Problem der zu großen Schrift« bekannt.

Reduzierung des Schriftgrads

Die meisten Webautoren setzen die Schriftgröße auf ihren Websites ein wenig herab, im Allgemeinen über folgende Deklaration für das Element `body`, wobei die angegebene Größe je nach verwendeter Schriftfamilie geringfügig variieren kann:

```
body { font-size: 80%; }
```

Wenn Sie die Schriftgröße nicht verändern wollen, sollten Sie explizit für `body` eine Schriftgrößendeklaration von `100.01%` vornehmen und sich im weiteren Verlauf (beispielsweise bei Überschriften) relativ darauf beziehen, etwa mittels Angaben in `em` (zum Beispiel `h1 { font-size: 1.4em; }`). Schreiben Sie `100.01%` und nicht `100%`, um einigen Browser-Bugs in Internet Explorer, Opera 6 und Safari auszuweichen[1].

```
/* Bugfixing bei Standardschriftgröße */
body { font-size: 100.01%; }
```

Ein weiterer sehr ärgerlicher Fehler begegnet Ihnen in vielen veralteten Browsern, vor allem im Quirks Mode des Internet Explorers (siehe Kapitel 6.8.2 »Doctype Switching«). In diesem Darstellungsmodus werden Schriftgrößendeklarationen nicht in Tabellenzellen hinein vererbt; darin enthaltene Schrift würde folglich deutlich größer wirken als auf dem Rest der Webseite. Um dieses Problem zu beheben, gehen Sie in zwei Schritten vor.

1. Zunächst deklarieren Sie die gewünschte Schriftgröße neben `body` auch für die Elemente `td` (Datenzellen) und `th` (Kopfzellen). Standardkonforme Browser würden Schrift in Tabellenzellen nun zu klein darstellen, nämlich in 64 Prozent der Standardschriftgröße (80 Prozent von 80 Prozent). Für diese gilt es also, die Schrift wieder hochzusetzen.

2. Dafür machen Sie sich im zweiten Schritt zunutze, dass alle Browser, die den Darstellungsfehler aufweisen, keine Maskierung verstehen und die zweite Regel daher ignorieren; so setzen Sie die Schriftgröße für standardkonforme Browser wieder auf 100 Prozent der vererbten Größe und erreichen, dass die Schrift in allen Browsern innerhalb und außerhalb von Tabellenzellen die gewünschten 80 Prozent der Standardschriftgröße erreicht.

```
/* Erster Schritt: Schriftgröße für den Dokumentkörper
  und Tabellenzellen setzen */
body, td, th { font-size: 80.01%; }

/* Zweiter Schritt: Schriftgröße für standardkonforme
  Browser für Tabellenzellen wieder hochsetzen */
td, th { f\ont-size: 100.01%; }
```

1 siehe ***[Caspers 2003]***

Browserunterstützung

Die Eigenschaft `font-size` wird von allen modernen Browsern unterstützt mit folgenden Einschränkungen:

IE ignoriert bis Version 7 den Wert `inherit`.

Alle Browser – Der Unterschied zwischen den Werten `x-small` und `xx-small` ist oft nicht erkennbar. Das ist kein Fehler, sondern folgt einer Empfehlung des W3C.

7.3.6 Schrift: Die zusammenfassende Eigenschaft font

DEFINITION

*Die **Eigenschaft** font fasst die Einstellungen der einzelnen Schrifteigenschaften (font-family, font-style, font-variant, font-weight und font-size) sowie die Eigenschaft line-height zur Bestimmung der Zeilenhöhe zusammen. Sie setzt zunächst die einzelnen Eigenschaften auf ihre Initialwerte und weist ihnen dann die explizit in der Deklaration angegebenen Werte zu. Dabei gilt folgende Reihenfolge, wobei Sie beachten müssen, dass Schriftgröße und Zeilenhöhe mit einem Schrägstrich (/) voneinander getrennt sind; optionale Angaben stehen in eckigen Klammern: [font-style] [font-variant] [font-weight] font-size[/line-height] [font-family]*

Die Regeln

```
body {
  font-size: 80%;
  line-height: 170%;
  font-family: Verdana, sans-serif;
}
h1 {
  font-family: Georgia, "Times New Roman", serif;
  font-size: 1.6em;
  font-variant: small-caps;
}
```

lassen sich somit wie folgt abkürzen:

```
body { font: 80%/170% Verdana, sans-serif; }

h1 {
  font: small-caps bold 1.6em Georgia, "Times New Roman", serif;
}
```

Die Angabe des Schriftgewichts der Überschrift erster Ordnung ist in der abgekürzten Schreibweise notwendig. Überschriften werden zwar standardmäßig fett dargestellt, der Initialwert der Eigenschaft `font-weight` ist jedoch normal, nicht `bold`, sodass Sie, würden Sie nicht explizit `bold` angeben, das Browserstylesheet überschreiben und normal gewichtete Überschriften erhalten würden.

Browserunterstützung

Alle Browser – Die Eigenschaft `font` wird von allen modernen Browsern unterstützt. Auf Einschränkungen wird bei den einzelnen Eigenschaften hingewiesen.

7.4 Text

7.4.1 Texteinzug: Die Eigenschaft text-indent

DEFINITION

*Die **Eigenschaft** `text-indent` bestimmt den Texteinzug der ersten Zeile eines Blockelements.*

Werte	`<Länge>`, `<Prozent>`, `inherit`
Initialwert	`0`
Vererbung	ja
Anwendung	Blockelemente

Geben Sie eine Länge an, wird der Text um den entsprechenden Wert eingezogen. Geben Sie einen negativen Wert an, wird die erste Zeile nicht eingezogen, sondern nach außen gerückt. Der Initialwert ist `0`, standardmäßig haben Blockelemente daher keinen Texteinzug.

Bei der Angabe eines Prozentwerts verhält sich der Texteinzug wie bei der Angabe einer Länge, wobei der Prozentwert sich auf die Breite des umschließenden Blocks bezieht.

Browserunterstützung

Alle Browser – Die Eigenschaft `text-indent` wird von allen modernen Browsern ohne Einschränkung unterstützt.

7.4.2 Horizontale Ausrichtung: Die Eigenschaft text-align

DEFINITION

*Die **Eigenschaft** `text-align` bestimmt, wie Inline-Inhalte eines Blockelements horizontal ausgerichtet werden.*

Werte	`left`, `right`, `center`, `justify`, `inherit`
Initialwert	Vom Browser und der Schriftrichtung abhängig. In unseren Breiten ist das für gewöhnlich `left`.
Vererbung	ja
Anwendung	Blockelemente

Über die Eigenschaft `text-align` können Sie Inhalte linkbündig (`left`), rechtsbündig (`right`) beziehungsweise zentriert (`justify`) ausrichten. Zudem können Sie mittels des Werts `justify` beidseitig im sogenannten Blocksatz ausrichten. Dabei wird die gesamte zur Verfügung stehende Breite ausgefüllt, indem die Wortabstände verändert werden. Passt ein Wort nicht mehr in eine Zeile, wird es in die nächste Zeile übernommen und der Wortabstand der Vorzeile erweitert.

Probleme mit Blocksatz

Im traditionellen Schriftsatz, beispielsweise im Buchdruck, wird meist Blocksatz verwendet, da durch das in sich geschlossene Satzbild Ruhe und Ausgeglichenheit vermittelt wird, was sich positiv auf die Lesbarkeit auswirkt. Auf Webseiten werden Sie Blocksatz aus zwei Gründen eher selten antreffen:

- Browser beherrschen die automatische Worttrennung bis heute leider noch nicht. Webautoren können Sollbruchstellen durch ein explizites Einfügen des Trennungsstrichs selbst erzeugen, üblicherweise durch die benannte Zeichenreferenz ``, davon ist jedoch abzuraten, da Sie sich damit eine ganze Reihe von Problemen einhandeln. Gerade ältere Browser zeigen den Strich immer an, andere interessieren sich überhaupt nicht dafür, sodass Sie sich ohnehin nicht darauf verlassen können, dass an der ausgewählten Stelle getrennt wird. Darüber hinaus ist die Verwendung auch in Bezug auf Suchmaschinen problematisch. So behandelt Google Trennstriche wie Leerzeichen; `Typografie` (in Googles Augen: `Ty po gra fie`) würde somit nicht gefunden werden, wenn nach »Typografie« gesucht würde.
- Bei langem Lesen auf der unruhigen Leseoberfläche eines Monitors ermüden unterschiedlich große Wortzwischenräume das Auge stärker als gleich große Wortzwischenräume bei linksbündigem Flattersatz.

Browserunterstützung

Alle Browser – Die hier vorgestellten Werte der Eigenschaft `text-align` werden von allen modernen Browsern ohne Einschränkung unterstützt. Darüber hinaus sieht CSS 2 die Möglichkeit vor, Text innerhalb von Tabellenzellen anhand einer vorgegebenen Zeichenkette auszurichten, beispielsweise Fließkommazahlen anhand des Kommas (`text-align: ',';`). Aufgrund mangelnder Browserunterstützung ist dies in CSS 2.1 nicht mehr möglich.

7.4.3 Ausschmückung: Die Eigenschaft text-decoration

DEFINITION

*Die **Eigenschaft** `text-decoration` beschreibt Ausschmückungen, die dem Text eines Elements hinzugefügt werden können. Die für die Textausschmückung benötigten Farben entsprechen dem Wert der Vordergrundfarbe `color` des Elements.*

Werte	`none`, `underline`, `overline`, `line-through`, `blink`, `inherit`
Initialwert	`none`
Vererbung	nein
Anwendung	global

- `none`: Der Text wird nicht ausgeschmückt.
- `underline`: Der Inhalt des Elements wird unterstrichen. Diesen Wert sollten Sie nur für Links verwenden, um Ihre Nutzer nicht zu verwirren. Es ist im Web allgemein üblich, dass alles, was unterstrichen ist, von Nutzern auch angeklickt werden kann.
- `overline`: Der Inhalt des Elements wird überstrichen.
- `line-through`: Der Inhalt des Elements wird durchgestrichen. Bedenken Sie, dass Sie dem Text keine logische Bedeutung verleihen, wenn Sie ihn mit dieser CSS-Eigenschaft durchstreichen. Womöglich ist das XHTML-Element `del` in Verbindung mit `ins` besser geeignet (siehe Kapitel 5.5.2).
- `blink`: Der Inhalt des Elements blinkt. Glücklicherweise ist es Browsern nicht vorgeschrieben, diesen Wert zu implementieren – leider verstehen ihn viele dennoch.

`text-decoration` können Sie mehrere Werte zuweisen, wenn Sie sie durch Leerzeichen getrennt nacheinander aufführen. Die Reihenfolge ist dabei beliebig.

Browserunterstützung

Die Eigenschaft `text-decoration` wird von allen modernen Browsern unterstützt mit folgenden Einschränkungen:

IE erkennt den Wert `blink` nicht.

Ein Nachkommenelement kann den Wert `blink` nicht erben, auch nicht durch die Deklaration `inherit`.

Konqueror akzeptiert den Wert `blink` nicht.

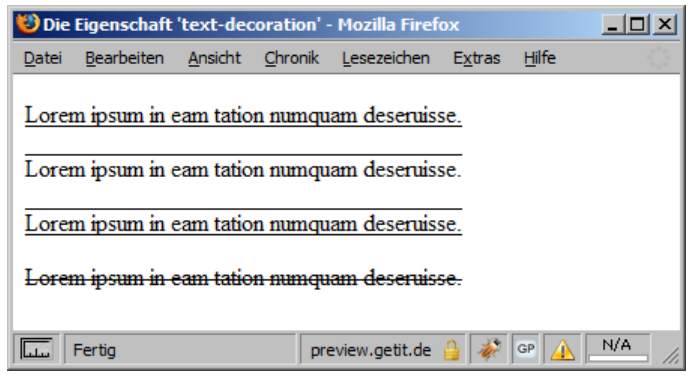


Abbildung 7.6: Über-, unter- und durchgestrichener Text; die Werte der Eigenschaft `text-decoration` können Sie miteinander kombinieren

7.4.4 Laufweite: Die Eigenschaft letter-spacing

DEFINITION

*Die **Eigenschaft** `letter-spacing` gibt das Abstandverhalten zwischen Zeichen an.*

Werte	`<Länge>`, `normal`, `inherit`
Initialwert	`normal`
Vererbung	ja
Anwendung	global

Wenn Sie eine Länge angeben, gibt diese den Abstand zwischen den Zeichen zusätzlich zu dem Standardabstand zwischen Zeichen an. Der Wert kann auch negativ sein; in diesem Fall würde der Abstand entsprechend verringert werden.

Relative Angabe

Geben Sie die Laufweite am besten relativ an; so passen sich die Leerräume der Schriftgröße an, wodurch die Homogenität des Schriftbilds erhalten bleibt.

Browserunterstützung

Alle Browser – Die Eigenschaft `letter-spacing` wird von allen modernen Browsern ohne Einschränkung unterstützt.

7.4.5 Wortabstände: Die Eigenschaft word-spacing

DEFINITION

*Die **Eigenschaft** `word-spacing` gibt das Abstandverhalten zwischen Wörtern an.*

Werte	`<Länge>`, `normal`, `inherit`
Initialwert	`normal`
Vererbung	ja
Anwendung	global

Eine Länge gibt den Abstand zwischen Wörtern zusätzlich zu dem Standardabstand zwischen Wörtern an. Ist der Wert negativ, wird der Abstand entsprechend verringert.

Relative Angabe

Wie bei der Eigenschaft `letter-spacing` sollten Sie die Wortabstände mit einer relativen Länge angeben.

Der Wortabstand wird nicht nur über `word-spacing`, sondern auch durch die Ausrichtung beeinflusst (siehe dazu Kapitel 7.4.2).

Browserunterstützung

Die Eigenschaft `word-spacing` wird von allen modernen Browsern unterstützt mit folgender Einschränkung:

Deklarieren Sie einen negativen Wert, der größer ist als der normale Raum zwischen Wörtern, würden sich benachbarte Wörter überlappen. Opera bis Version 7 fügt einfach einen Zeilenvorschub ein, sodass jedes Wort am Beginn einer neuen Zeile steht. Ab Version 8 werden in diesem Fall die Wörter ohne Abstand hintereinandergesetzt. Dies widerspricht nicht der Spezifikation.

7.4.6 Groß- und Kleinschreibung: Die Eigenschaft text-transform

DEFINITION

*Die **Eigenschaft** `text-transform` bestimmt die Groß- und Kleinschreibung von Text unabhängig von der tatsächlichen Schreibweise im Dokument.*

Werte	`none`, `uppercase`, `lowercase`, `capitalize`, `inherit`
Initialwert	`normal`
Vererbung	ja
Anwendung	global

Beim Wert `uppercase` werden alle Buchstaben als Großbuchstaben dargestellt, bei `lowercase` als Kleinbuchstaben. Geben Sie den Wert `capitalize` an, wird der erste Buchstabe jedes Worts als Großbuchstabe dargestellt.

SIE SOLLTEN VERMEIDEN, TEXTE, DIE MEHR ALS EINE HANDVOLL WÖRTER UMFASSEN, KOMPLETT IN GROSSBUCHSTABEN ZU SETZEN. MENSCHEN LESEN GÄNGIGE WÖRTER NICHT BUCHSTABE FÜR BUCHSTABE, SONDERN NEHMEN SIE ALS GANZES AUFGRUND IHRER BEKANNTEN FORM WAHR. TEXTE WIE DIESER HIER LASSEN SICH DEUTLICH LANGSAMER LESEN ALS NORMAL GESETZTER TEXT. FOLGENDER VERGLEICH LIEGT NAHE: STELLEN SIE SICH VOR, JEMAND WÜRDE IHNEN DIESEN TEXT INS OHR BRÜLLEN.

das gleiche gilt für text, der komplett in kleinbuchstaben gesetzt ist. der text wirkt unbetont und liest sich so, als ob er genuschelt würde. so viele möglichkeiten css bietet, so viele chancen haben schlechte webautoren, die lesbarkeit ihrer webseiten vollkommen zu ruinieren; das betrifft nicht nur schwer lesbare schriftschnitte und -größen, sondern eben auch die textgestaltung.

Browserunterstützung

Alle Browser – Die Eigenschaft `text-transform` wird von allen modernen Browsern ohne Einschränkung unterstützt.

7.4.7 Behandlung von Leerraum: Die Eigenschaft white-space

DEFINITION

*Die **Eigenschaft** `white-space` bestimmt die Darstellung von Leerraumzeichen (Leerzeichen, Tabulatoren, Wagenrücklauf und Zeilenvorschub) innerhalb eines Elements.*

Werte	`normal`, `pre`, `nowrap`, `inherit`
Initialwert	`normal`
Vererbung	ja
Anwendung	global

Beim Wert `normal` werden mehrere Leerraumzeichen hintereinander zu einem zusammengezogen. Wo Inhalte länger sind, als ihr Element oder die Breite des Browserfensters es erlaubt, werden Zeilenumbrüche eingefügt. Weitere Zeilenumbrüche lassen sich durch das (X)HTML-Element `br` einfügen.

Der Wert `pre` gibt an, dass die Darstellung des Texts der von präformatiertem Text entspricht, also dem XHTML-Element `pre`. Text wird genau so dargestellt, wie er im Editor notiert wird; Sequenzen von Leerraumzeichen werden vollständig ausgegeben. Elementtypen wie `br`, die die Zeilenumbrüche der Ausgabe kontrollieren, werden interpretiert.

`nowrap` bedeutet, dass Zeilen zwar nicht umbrochen werden, Leerraum aber ansonsten wie gewohnt behandelt wird. Elementtypen wie `br`, die die Zeilenumbrüche der Ausgabe kontrollieren, werden interpretiert.

Browserunterstützung

Alle Browser – Die vorgestellten Werte der Eigenschaft `white-space` werden von allen modernen Browsern ohne Einschränkung unterstützt. CSS 2.1 erlaubt darüber hinaus die Werte `pre-wrap` und `pre-line`, die jedoch noch von keinem Browser richtig verstanden und daher an dieser Stelle nicht vorgestellt werden.

7.5 Das CSS-Boxmodell

Aus der Sicht von CSS setzt sich ein XHTML-Dokument – wie auch jedes andere strukturierte (XML-)Dokument – aus einer Vielzahl von rechteckigen, im Allgemeinen unsichtbaren Boxen zusammen, deren Verhalten, Fluss und Ausdehnung Webautoren mit einer ganzen Reihe von Eigenschaften beeinflus-

sen können. Das sogenannte CSS-Boxmodell ist eines der wichtigsten Teile von CSS und dient als Grundlage der Darstellung und Positionierung aller Elemente.

7.5.1 Polsterung, Rahmen und Randabstand

DEFINITION

Jedes Element erzeugt eine rechteckige Box, die das Element umgibt und sich aus weiteren Boxen zusammensetzt: einer ***Contentbox*** *(zum Beispiel ein Text oder ein Bild), einer* ***Paddingbox*** *(Polsterung), einer* ***Borderbox*** *(Rahmen) und einer* ***Marginbox*** *(Randabstand). Jeder dieser Bereiche kann in oben, rechts, unten und links aufgeteilt werden, siehe Abbildung 7.7.*

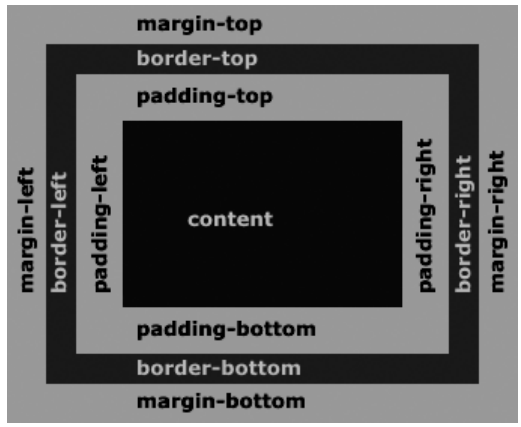


Abbildung 7.7: Das CSS-Boxmodell

Kanten

Der Umfang einer Box wird als Kante (*edge*) bezeichnet. Jede Box hat vier Kanten:

- **Content-Kante** oder innere Kante (*content edge*): Umgibt die Contentbox, also den Elementinhalt.
- **Kante der Polsterung** (*padding edge*): Umgibt die Paddingbox, die den Abstand des Elementinhalts zum Rahmen darstellt. Wenn die Paddingbox die Breite 0 hat, entspricht die Kante der Polsterung der Content-Kante.
- **Rahmenkante** (*border edge*): Umgibt die Borderbox, die dem Rahmen des Elements entspricht. Wenn die Borderbox die Breite 0 hat, entspricht die Rahmenkante der Kante der Polsterung.

- **Abstandkante** oder äußere Kante (*margin edge*): Umgibt die Marginbox, die den Abstand des Elements zu den umgebenden Elementen bildet. Wenn die Marginbox die Breite 0 hat, entspricht die Abstandkante der Rahmenkante.

Elementbreite und -höhe

Die Gesamtbreite und -höhe eines Elements addiert sich aus der Breite beziehungsweise Höhe der einzelnen Boxen. Ein Element `E` mit

```
E {
  width: 100px;
  padding: 20px;     /* je 20px links und rechts */
  border: 1px solid; /* je  1px links und rechts */
  margin: 0 30px;    /* je 30px links und rechts */
}
```

hat folglich eine Gesamtbreite von 202 Pixeln. Bei einem Element `E` mit

```
E {
  margin: 0;
  border: 0;
  padding: 0;
}
```

entspricht die Gesamtbreite der Breite der Contentbox, die im Allgemeinen so breit ist, wie es ihr Inhalt erfordert.

> > > HINWEIS
> *Weitere Informationen zur Berechnung der Breite und Höhe von Elementen erhalten Sie in Kapitel 7.8 »Breite und Höhe«.*

Hintergrund

Der Hintergrund der einzelnen Boxen ist wie folgt definiert:

- Der Hintergrund der Content-, Padding- und zunächst auch der Borderbox wird durch die Eigenschaft `background-color` des Elements bestimmt.
- Der Hintergrund der Borderbox wird explizit durch die Eigenschaft `border-color` bestimmt. Bei unterbrochenen Rahmenstilen (`dashed` oder `dotted`) scheint die Hintergrundfarbe durch.
- Der Hintergrund der Marginbox ist transparent.

7.5.2 Boxtypen

Wie ein Element dargestellt wird und wie es benachbarte und verschachtelte Elemente beeinflusst, ist abhängig davon, welche Art von Box es umgibt, das heißt, welche Art von Box ein CSS-fähiges Benutzerprogramm für dieses Element generiert. Stellen Sie sich alle im Folgenden aufgeführten Boxen als nor-

malerweise unsichtbare Rechtecke vor, die wie Ziegelsteine in einer Linie aneinanderstoßen oder auf- beziehungsweise untereinanderliegen.

DEFINITION

Jedes Element bildet eine Box, an der sich seine Kindelemente ausrichten. Diese Box wird ***umschließender Block*** *genannt.*

Man spricht davon, dass eine Box den umschließenden Block für ihre Nachkommen einrichtet; normalerweise liegen alle Boxen, die durch die Nachkommen gebildet werden, innerhalb des umschließenden Blocks. Besonderheiten gibt es bei positionierten Elementen zu beachten, worauf ich in weiteren Abschnitten dieser Einführung genauer eingehen werde.

Was der äußerste umschließende Block eines XHTML-Dokuments ist, der alle anderen Boxen enthält, wird durch den Browser festgelegt, wobei auch die Fenstergröße eine Rolle spielt.

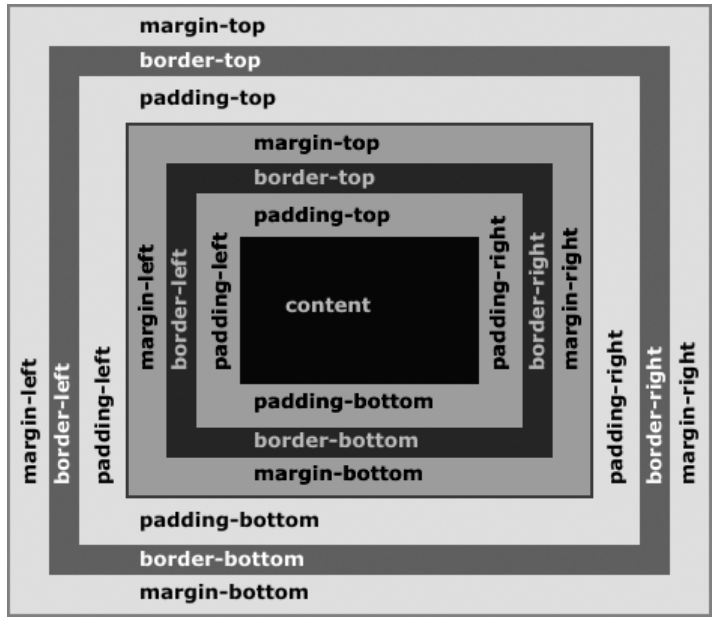


Abbildung 7.8: Umschließender Block

In Abbildung 7.8 zeigt der äußere Rand den äußersten umschließenden Block, der hier gleichzeitig der umschließende Block für das hell dargestellte Element ist. Das helle Element richtet wiederum einen umschließenden Block ein, der

für das dunkler gezeichnete innere Element gilt. Folgendes XHTML-Beispiel erzeugt Blöcke wie in dieser Abbildung illustriert.

```
<body>
  <div>
    <div>content</div>
  </div>
</body>
```

DEFINITION

*Alle Blockelemente erzeugen eine sogenannte **Hauptblockbox**, die entweder ausschließlich Blockboxen oder ausschließlich Inlineboxen enthält. Die Hauptblockbox richtet den umschließenden Block dieser Boxen ein.*

Im folgenden Beispiel enthält die durch das äußere `div`-Element gebildete Hauptblockbox eine durch das innere `div`-Element gebildete konkrete Blockbox (die wiederum die Hauptblockbox für ihre Inhalte darstellt). Aus der Definition, dass eine Hauptblockbox entweder nur Blockboxen oder nur Inlineboxen enthält, ergibt sich die Forderung, dass die durch »Lorem ipsum in eam tation numquam deseruisse« und »Duo ei fugit minim minimum« gebildeten Boxen ebenfalls Blockboxen sind. Diese virtuellen Blockboxen werden »anonyme Blockboxen« genannt.

DEFINITION

***Anonyme Blockboxen** sind nicht konkret existierende Boxen innerhalb einer Hauptblockbox. Sie werden eingefügt, sobald eine Hauptblockbox Blockboxen enthält, und dienen dazu, die Forderung zu erfüllen, dass eine Hauptblockbox ausschließlich Blockboxen enthält, sobald sie mindestens eine enthält.*

```
<div>
  Lorem ipsum in eam tation numquam deseruisse
  <div>Sea eu eripuit deterruisset</div>
  Duo ei fugit minim minimum
</div>
```

Abbildung 7.9 stellt alle Blockboxen dar, die in diesem Beispiel erzeugt werden.

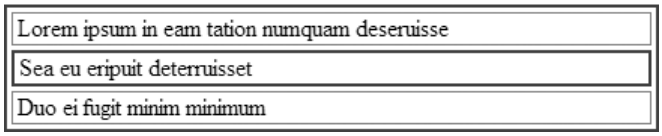


Abbildung 7.9: Konkrete und anonyme Blockboxen

DEFINITION

*Alle Inline-Elemente erzeugen eine sogenannte **Inlinebox**, die weitere Inlineboxen enthalten kann.*

Folgendes Beispiel erweitert das vorangehende um ein `span`-Element. Damit verfügt das innere `div`-Element über drei Inlineboxen: zum einen die konkrete Inlinebox, die durch das `span`-Element gebildet wird, und zum anderen die beiden anonymen Inlineboxen, die durch »Sea eu« und »deterruisset« erzeugt werden.

DEFINITION

***Anonyme Inlineboxen** sind nicht konkret existierende Inlineboxen, die von einem Blockelement, das keine anderen Elemente oder nur Inline-Elemente enthält, um Inhalte herum erzeugt werden, die nicht selbst eine Inlinebox generieren.*

```
<div>
  Lorem ipsum in eam tation numquam deseruisse
  <div>Sea eu <span>eripuit</span> deterruisset</div>
  Duo ei fugit minim minimum
</div>
```

Abbildung 7.10 stellt neben den Blockboxen auch alle Inlineboxen dar, die in diesem Beispiel erzeugt werden. Natürlich erzeugen auch die anonymen Blockboxen entsprechende (anonyme) Inlineboxen.

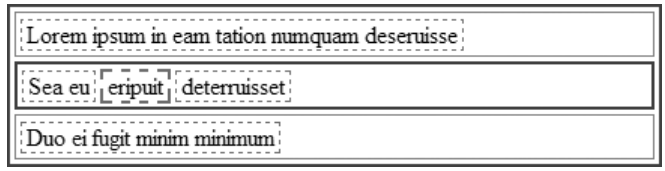


Abbildung 7.10: Konkrete und anonyme Blockboxen (durchgezogene Linien) sowie konkrete und anonyme Inlineboxen (gestrichelte Linien)

Zwei besondere Arten von Boxen, die sich je nach Kontext wie eine Inline- oder wie eine Blockbox verhalten, sind kompakte Boxen und Run-in-Boxen.

DEFINITION

*Eine **kompakte Box** (compact box) ist eine spezielle Box, die sich wie folgt verhält: Folgt der kompakten Box eine (nicht gefloatete und nicht absolut positionierte) Blockbox, wird die kompakte Box wie eine einzeilige Inlinebox formatiert. Die Breite der kompakten Box wird mit der Breite des Margin-Bereichs der Blockbox verglichen: Ist die Breite kleiner oder gleich, wird die kompakte Box im Margin-Bereich dargestellt (also neben der Blockbox), ansonsten erfolgt ein Zeilenumbruch.*

Folgendes Beispiel zeigt die Anwendung anhand einer Definitionsliste. Die `dt`-Elemente sollen als kompakte Box neben den `dd`-Elementen dargestellt werden, solange sie eine Länge von 6 em nicht überschreiten. Abbildung 7.11 zeigt, wie Opera 9 das Dokument anzeigt. Der Definitionsterm »Benutzer« passt in den Randbereich hinein, der zweite Definitionsterm hingegen nicht; er wird als Blockbox dargestellt und erzeugt einen Zeilenumbruch. Andere Browser wie leider auch aktuelle Safari- und Firefox-Versionen – vom Internet Explorer ganz zu schweigen – können mit kompakten Boxen nichts anfangen und zeigen die Definitionsliste in der Standarddarstellung.

```
dt {
  display: compact;
  font-weight: bold;
}
dd { margin: 0 0 .5em 6em; }
```

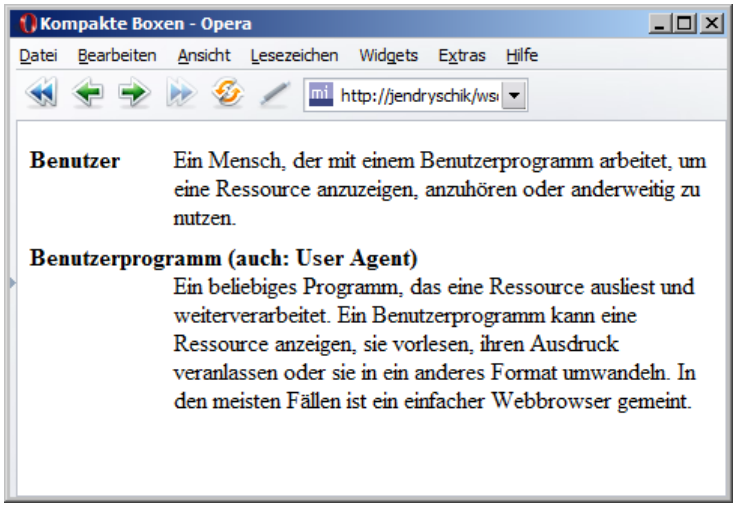


Abbildung 7.11: Kompakte Boxen

DEFINITION

*Eine **Run-in-Box** ist eine spezielle Box, die sich wie folgt verhält: Folgt der Run-in-Box eine (nicht gefloatete und nicht absolut positionierte) Blockbox, wird die Run-in-Box zur ersten Inlinebox der Blockbox, andernfalls wird die Run-in-Box zu einer Blockbox.*

```
dt {
  display: run-in;
  font-weight: bold;
  padding: 0 1em 0 0;
}
```

Auch bei Run-in-Boxen ist es leider so, dass kaum ein Browser sie so darstellt, wie in Abbildung 7.12 gezeigt.

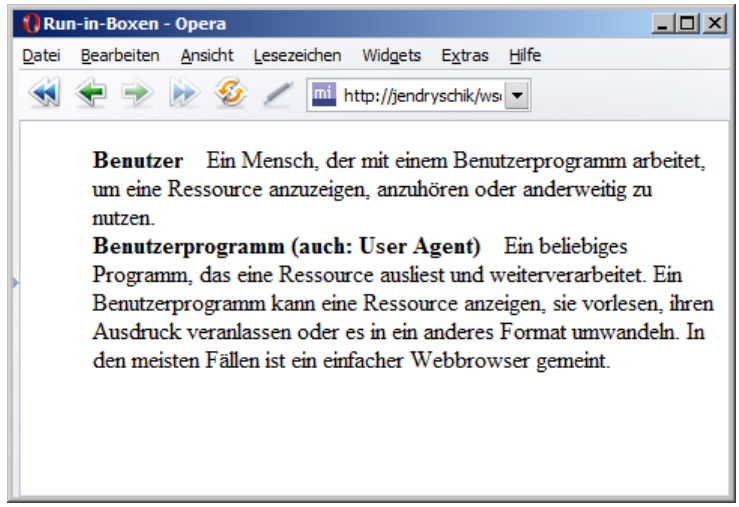


Abbildung 7.12: Run-in-Boxen

DEFINITION

Inline-Elemente werden horizontal nebeneinander dargestellt, beginnend von oben in einem umschließenden Block. Der unsichtbare rechteckige Bereich, der eine Zeile Inline-Elemente enthält, wird ***Zeilenbox*** *genannt.*

Zeilenumbruch

Passen mehrere Inlineboxen nicht in eine einzelne Zeilenbox, weil der horizontal zur Verfügung stehende Platz nicht ausreicht (etwa weil das Browserfenster nicht groß genug oder die Breite des Elements auf einen kleineren Wert festgelegt ist), erfolgt ein Zeilenumbruch, und es werden so viele weitere Zeilenboxen eingefügt, wie notwendig sind, um alle Inlineboxen dazustellen.

7.5.3 Polsterung: Die Eigenschaft padding

DEFINITION

Die ***zusammenfassende Eigenschaft padding*** *gibt die Breite der Paddingbox eines Elements an. Die Eigenschaften* `padding-top`, `padding-right`, `padding-bottom` *und* `padding-left` *setzen die Breite entsprechend nur für eine Seite der Paddingbox.*

Werte	<Länge>, <Prozent>, inherit
Initialwert	0
Vererbung	nein
Anwendung	global

Die Angabe in Prozent wird relativ zur Breite des umschließenden Blocks der erzeugenden Box berechnet, auch für `padding-top` und `padding-bottom`.

Kurzschreibweisen

Wird für `padding` nur ein Wert angegeben, bezieht er sich – wie in Tabelle 7.3 zu sehen – auf alle vier Seiten. Werden zwei Werte angegeben, bezieht sich der erste auf oben und unten und der zweite auf rechts und links. Werden drei Werte angegeben, bezieht sich der erste auf oben, der zweite auf rechts und links und der dritte auf unten. Werden vier Werte angegeben, bezieht sich der erste auf oben, der zweite auf rechts, der dritte auf unten und der vierte auf links.

Deklaration	**Polsterung**			
	oben	**rechts**	**unten**	**links**
`padding: 1em;`	1 em	1 em	1 em	1 em
`padding: 1em 2em;`	1 em	2 em	1 em	2 em
`padding: 1em 2m 3em;`	1 em	2 em	3 em	2 em
`padding: 1em 2em 3em 4em;`	1 em	2 em	3 em	4 em

Tabelle 7.3: Die Eigenschaft `padding` fasst die Eigenschaften `padding-top`, `padding-right`, `padding-bottom` und `padding-left` zusammen

Korrektur der Zeilenhöhe

Wenn Sie `padding-top` und `padding-bottom` auf Inline-Elemente anwenden, werden Sie in einigen Browsern möglicherweise Überlappungen mit darüber- und/oder darunterliegenden Zeilen feststellen. Dieses Verhalten können Sie korrigieren, indem Sie die Zeilenhöhe mit der Eigenschaft `line-height` entsprechend anpassen.

Browserunterstützung

Alle Browser – Die Eigenschaft `padding` wird von allen modernen Browsern korrekt dargestellt.

7.5.4 Rahmenstil: Die zusammenfassende Eigenschaft border-style

DEFINITION

*Die **zusammenfassende Eigenschaft** `border-style` gibt die Art der gezeichneten Linie des Rahmens an. Die Eigenschaften `border-top-style`, `border-right-style`, `border-bottom-style` und `border-left-style` setzen die Linienart entsprechend nur für eine Seite der Borderbox.*

Werte	`none`, `hidden`, `dotted`, `dashed`, `solid`, `double`, `groove`, `ridge`, `inset`, `outset`, `inherit`
Initialwert	`none`
Vererbung	nein
Anwendung	global

Die einzelnen Werte haben jeweils folgende Bedeutung:

- `none` und `hidden`: Diese Werte geben explizit an, dass kein Rahmen dargestellt werden soll. `none` ist der Initialwert.
- `dotted`: Gepunkteter Rahmen.
- `dashed`: Gestrichelter Rahmen.
- `solid`: Durchgezogener Rahmen.
- `double`: Doppelter durchgezogener Rahmen.
- `groove`: Der Rahmen sieht aus wie in die Oberfläche geritzt.
- `ridge`: Der Rahmen sieht aus, als würde er auf der Oberfläche aufliegen.
- `inset`: Der Rahmen lässt das Element so aussehen, als liege es tiefer in der Oberfläche.
- `outset`: Der Rahmen lässt das Element so aussehen, als hebe es sich ein wenig aus der Oberfläche heraus.

Werden für `border-style` mehrere durch Leerraum getrennte Werte angegeben, gelten die gleichen Regeln wie für die Eigenschaft `padding`. Folgende Regeln sind äquivalent und setzen den Linienstil eines Elements `E` auf `solid` oben, rechts und links und `dashed` unten:

```
E {
  border-style: solid;
  border-bottom-style: dashed;
}

E { border-style: solid solid dashed solid; }
```

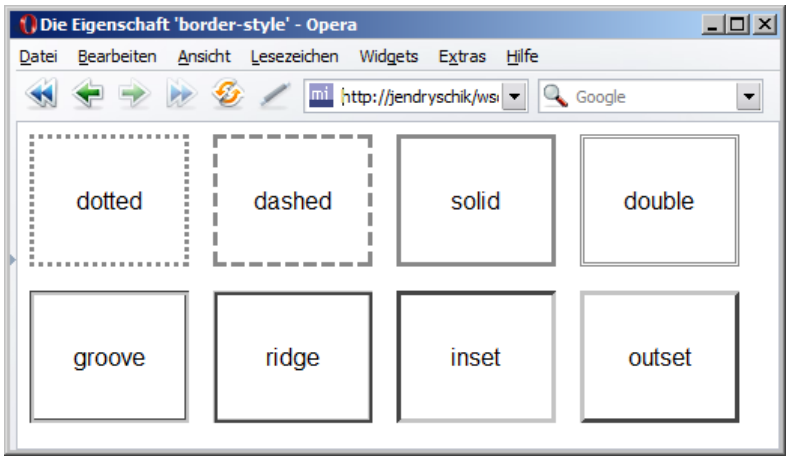


Abbildung 7.13: Die Eigenschaft `border-style`

Browserunterstützung

Die Eigenschaft `border-style` wird von allen modernen Browsern unterstützt mit folgenden Einschränkungen:

- IE 6 erkennt den Wert `inherit` nicht, an Tabellenelementen wird der Wert `hidden` ignoriert, und der Wert `dotted` wird wie `dashed` dargestellt, wenn die Rahmenbreite nur 1 Pixel ist.
- IE 7 erkennt den Wert `inherit` nicht, und an Tabellenelementen wird der Wert `hidden` ignoriert.
- Opera 8 erkennt den Wert `inherit` nur für die zusammenfassende Eigenschaft, nicht für die Einzeleigenschaften.
- Konqueror macht im normalen Elementfluss einen nicht erlaubten Unterschied zwischen den Werten `none` und `hidden`, indem er `hidden` wie einen transparenten Rahmen darstellt.

7.5.5 Rahmenbreite: Die Eigenschaft border-width

DEFINITION

*Die **zusammenfassende Eigenschaft border-width** gibt die Breite der Borderbox an. Die Eigenschaften `border-top-width`, `border-right-width`, `border-bottom-width` und `border-left-width` setzen die Breite entsprechend nur für eine Seite der Borderbox.*

Werte	`medium`, `thin`, `thick`, `<Länge>`, `inherit`
Initialwert	`medium`
Vererbung	nein
Anwendung	global

Der Wert `medium` zeichnet einen Rahmen mittlerer Dicke, `thin` zeichnet einen dünnen Rahmen, `thick` einen dicken Rahmen. Die Interpretation der Werte `thin`, `medium` und `thick` ist browserabhängig.

Auch für `border-width` können bis zu vier Werte angegeben werden, die analog zur Eigenschaft `border-style` für die entsprechenden Seiten der Borderbox gelten.

Browserunterstützung

Die Eigenschaft `border-width` wird von allen modernen Browsern unterstützt mit folgenden Einschränkungen:

IE ignoriert bis Version 7 den Wert `inherit`.

Opera 8 erkennt den Wert `inherit` nur für die zusammenfassende Eigenschaft, nicht für die Einzeleigenschaften.

7.5.6 Rahmenfarbe: Die zusammenfassende Eigenschaft border-color

DEFINITION

*Die **zusammenfassende Eigenschaft** `border-color` gibt die Farbe der Borderbox an. Die Eigenschaften `border-top-color`, `border-right-color`, `border-bottom-color` und `border-left-color` setzen die Farbe entsprechend nur für eine Seite der Borderbox.*

Werte	<Farbe>, transparent, inherit
Initialwert	Der Initialwert der Eigenschaft border-color ist der Wert der Eigenschaft color dieses Elements.
Vererbung	nein
Anwendung	global

Wenn Sie den Wert transparent angeben, ist der Rahmen durchsichtig, das heißt, darunterliegende Elemente scheinen durch.

Auch für border-color können Sie bis zu vier Werte angeben, die analog zur Eigenschaft border-style für die entsprechenden Seiten der Borderbox gelten.

Browserunterstützung

Die Eigenschaft border-color wird von allen modernen Browsern unterstützt mit folgender Einschränkung:

Der IE bis Version 7 ignoriert die Werte transparent und inherit.

7.5.7 Rahmen: Die zusammenfassende Eigenschaft border

DEFINITION

*Die **zusammenfassende Eigenschaft** border fasst die Eigenschaften border-style, border-width und border-color zusammen. Die Eigenschaften border-top, border-right, border-bottom und border-left geben Rahmenstil, -breite und -farbe entsprechend nur für eine Seite der Borderbox an. Weggelassene Werte werden auf ihre Initialwerte gesetzt.*

Sie können die Werte durch Leerzeichen getrennt in beliebiger Reihenfolge einfach hintereinanderschreiben. Folgende Regeln führen immer zum gleichen Ergebnis:

```
h1 {
  border-width: 2px;
  border-style: solid;
  border-color: #309;
}

h1 {
  border-top: 2px solid #309;
  border-right: 2px solid #309;
  border-bottom: 2px solid #309;
  border-left: 2px solid #309;
```

```
}

h1 { border: 2px solid #309; }

h1 { border: solid #309 2px; }
```

Browserunterstützung

Alle Browser – Die Eigenschaft `border` wird von allen modernen Browsern unterstützt, auf Einschränkungen wird bei den Einzeleigenschaften hingewiesen.

7.5.8 Randabstand: Die Eigenschaft margin

DEFINITION

*Die **zusammenfassende Eigenschaft** `margin` gibt die Breite der Marginbox eines Elements an. Die Eigenschaften `margin-top`, `margin-right`, `margin-bottom` **und** `margin-left` setzen die Breite entsprechend nur für eine Seite der Marginbox.*

Werte	`<Länge>`, `<Prozent>`, `auto`, `inherit`
Initialwert	`0`
Vererbung	nein
Anwendung	Blockelemente, Inline-Elemente nur eingeschränkt

Negative Werte sind erlaubt; ein Element mit der Deklaration `margin-left: -10px;` hat einen negativen Abstand von 10 Pixeln und wird um diese Länge nach links verschoben.

Die Angabe in Prozent wird relativ zur Breite des umschließenden Blocks der erzeugenden Box berechnet, auch für `margin-top` und `margin-bottom`.

Wenn Sie den Wert `auto` angeben, wird der Abstand automatisch berechnet.

Wird für `margin` nur ein Wert angegeben, gelten dieselben Regeln wie für die Eigenschaft `padding` (siehe Kapitel 7.5.3).

Zusammenfallende Randabstände

Vertikale Randabstände werden im Gegensatz zu horizontalen Abständen im Allgemeinen nicht addiert, sondern fallen zusammen, wobei der größere Abstand »gewinnt«. Der Abstand der beiden folgenden Elemente zueinander beträgt somit nicht 50, sondern nur 30 Pixel:

```
<p style="margin-bottom: 30px;">Absatz</p>

<p style="margin-top: 20px;">Absatz</p>
```

Aber nicht nur die vertikalen Ränder benachbarter Boxen fallen zusammen, sondern auch Ränder verschachtelter Boxen. Aus diesem Grund liegt das `div`-Element im folgenden Listing 7.1 nicht direkt an der Oberkante des Browserfensters an, sondern wird mit einem Abstand von 20 Pixeln dargestellt, den es von seinem Kindelement `p` »übernimmt« (siehe Abbildung 7.14).

Ränder fallen nur dann zusammen, wenn sie durch keine Polster- oder Rahmenbereiche voneinander getrennt sind, das heißt, wenn Padding- und Borderbox auf 0 gesetzt sind. Da der Wert der Eigenschaft `padding-bottom` des `div`-Elements ungleich 0 ist, fallen die unteren Randabstände nicht zusammen.

> > > HINWEIS

Vertikale Ränder zwischen einer fließenden und einer nicht fließenden Box sowie von absolut und relativ positionierten Boxen fallen nicht zusammen. Mehr dazu in Kapitel 7.7 »Positionierung«.

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

<html xmlns="http://www.w3.org/1999/xhtml"
  lang="de" xml:lang="de">
  <head>
    <meta http-equiv="Content-Type"
      content="text/html; charset=utf-8" />
    <title>Zusammenfallende Randabstände</title>

    <style type="text/css">
      /* <![CDATA[ */
      * {
        margin: 0;
        padding: 0;
      }
      div {
        color: black;
        background-color: lime;
        padding-bottom: 1px;
      }
      p {
        color: black;
        background-color: yellow;
        margin: 20px;
```

```
        }
        /* ]]> */
      </style>
    </head>

    <body>
      <div>
        <p>Lorem ipsum in eam tation numquam deseruisse. Sea eu
          eripuit deterruisset, duo ei fugit minim minimum,
          nam probo fabulas praesent ex. In adhuc adipisci
          pri, pri id autem semper. Duis tollit in vim. Ei vim
          nonumy epicuri, natum democritum te est.</p>
      </div>
    </body>
  </html>
```

Listing 7.1: Zusammenfallende Randabstände

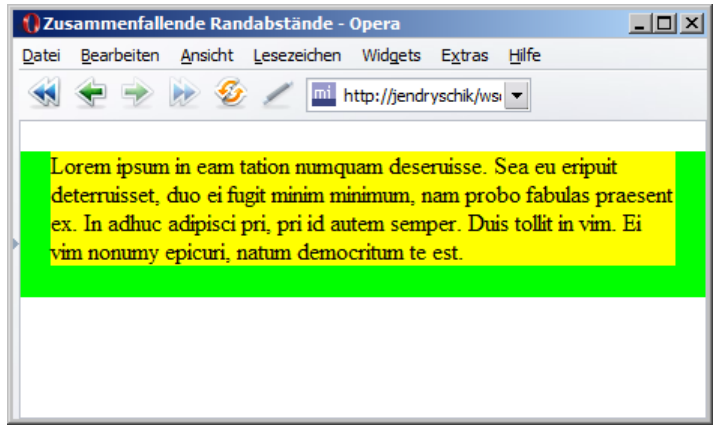


Abbildung 7.14: Zusammenfallende Randabstände

Browserunterstützung

Alle Browser – Die Eigenschaft `margin` wird von allen modernen Browsern ohne Einschränkungen unterstützt.

7.5.9 Boxtyp eines Elements: Die Eigenschaft display

DEFINITION

*Die **Eigenschaft** `display` gibt an, ob und mit welcher Art von Box ein Element dargestellt werden soll.*

Werte	`inline`, `block`, `inline-block`, `list-item`, `marker`, `none`, `run-in`, `compact`, `table`, `inline-table`, `table-row-group`, `table-column`, `table-column-group`, `table-header-group`, `table-footer-group`, `table-row`, `table-cell`, `table-caption`, `inherit`
Initialwert	`inline`
Vererbung	nein
Anwendung	global

- `inline`: Das Element wird als Inline-Element dargestellt, erzeugt also eine oder mehrere Inlineboxen. Dieser Wert ist zwar der Initialwert, jedoch wird er üblicherweise direkt durch das Browserstylesheet für Elemente wie Absätze (`p`), Überschriften (`h1` bis `h6`), Container (`div`) oder Tabellen- und Listenelemente überschrieben.
- `block`: Das Element wird als Blockelement dargestellt, erzeugt also eine Hauptblockbox.
- `inline-block`: Das Element wird als Blockelement innerhalb der Textzeile dargestellt, also wie ein Inline-Element, hat aber ansonsten alle Eigenschaften eines Blockelements. Dieser Wert wurde in CSS 2.1 neu eingeführt.
- `list-item`: Das Element wird als Listenpunkt dargestellt und erzeugt eine Hauptblockbox und eine inzeilige Listenelementbox. Beachten Sie: Die Deklaration `display: list-item;` ersetzt nicht die logische Auszeichnung von Listen in XHTML, sondern simuliert nur deren Darstellung.
- `marker`: Dieser Wert deklariert über die Pseudoelemente `:before` und `:after` erzeugten Inhalt vor oder hinter einer Box als Markierung. Auf Markierungen wird in dieser Einführung nicht weiter eingegangen, da sie auch in modernen Browsern kaum oder gar nicht unterstützt werden.

- `none`: Das Element erzeugt keine Box, verschwindet also aus der Darstellung des Dokuments. Dieser Wert wird oft in Kombination mit clientseitigen Scripts verwendet, um ein Element zunächst auszublenden und als Folge einer Benutzerinteraktion (zum Beispiel Klick auf einen Link) schließlich einzublenden, indem der `display`-Wert dynamisch geändert wird, meistens auf `block`.
- `run-in` und `compact`: Es wird eine Run-in- beziehungsweise eine kompakte Box erzeugt, die sich abhängig vom Kontext entweder als Inline- oder als Blockbox darstellt. Der Wert `compact` wurde nicht in CSS 2.1 übernommen.
- `table`, `inline-table`, `table-row-group`, `table-column`, `table-column-group`, `table-header-group`, `table-footer-group`, `table-row`, `table-cell` und `table-caption`: Diese Werte dienen dazu, das Verhalten von Elementen als Element einer Tabelle zu simulieren. Die einzelnen Werte lassen sich, wie in Listing 7.2 aufgeführt, den XHTML-Tabellenelementen zuordnen. Bitte beachten Sie auch hier, dass dadurch keinesfalls die logische Auszeichnung von Inhalten mittels der XHTML-Tabellenelemente ersetzt werden kann.

```
table    { display: table;              }
tr       { display: table-row;          }
thead    { display: table-header-group; }
tbody    { display: table-row-group;    }
tfoot    { display: table-footer-group; }
col      { display: table-column;       }
colgroup { display: table-column-group; }
td, th   { display: table-cell;         }
caption  { display: table-caption;      }
```

Listing 7.2: XHTML-Elemente und die ihnen zugeordneten `display`-Werte

Browserunterstützung

Kein Browser unterstützt alle Werte der Eigenschaft `display`.

Firefox erkennt die Werte `inline-block`, `compact` und `run-in` nicht. Folgende Werte für Tabellen werden unterstützt: `table`, `inline-table`, `table-column-group`, `table-column`, `table-cell` und `table-caption`.

Der IE bis Version 7 erkennt die Werte `run-in`, `compact` und `marker` nicht. Die Werte für Tabellen werden alle nicht unterstützt.

Opera erkennt alle Werte außerhalb von Tabellen; bis Version 8 wird der Inhalt von mit `display: inline-block` deklarierten Elementen bis an die Oberkante des `margin-top` hochgeschoben. Die Werte für Tabellen werden vom Opera-Browser bis Version 9 mit Ausnahme von `table-column-group` und `table-column` alle erkannt. Die 9er-Versionen interpretieren sämtliche Werte korrekt.

Safari unterstützt den Wert `compact` nicht, der allerdings auch nicht Teil von CSS 2.1 ist.

Konqueror bis Version 3.5 erkennt die Werte `run-in`, `compact` und `marker` nicht.

7.6 Ausrichtung und Elementfluss

7.6.1 Vertikale Ausrichtung: Die Eigenschaft vertical-align

DEFINITION

*Die **Eigenschaft** `vertical-align` bestimmt die vertikale Ausrichtung von Inline-Elementen innerhalb von Zeilenboxen oder Tabellenzellen.*

Werte	`baseline`, `sub`, `super`, `top`, `text-top`, `middle`, `bottom`, `text-bottom`, `<Länge>`, `<Prozent>`, `inherit`
Initialwert	`inherit`
Vererbung	nein
Anwendung	Inline-Elemente, Tabellenzellen (Elemente `td` und `th`) sowie alle Elemente, deren `display`-Eigenschaft den Wert `table-cell` besitzt

Die Bedeutung der folgenden Werte ist abhängig davon, ob die Eigenschaft `vertical-align` auf Inline-Elemente oder Tabellenzellen angewendet wird.

Wert	Inline-Elemente	Tabellenzellen
`baseline`	Ausrichtung an der Basislinie. Falls es keine Basislinie gibt, wird das Element bündig mit der Unterkante der Elternbox ausgerichtet.	Ausrichtung an der Basislinie, wobei die Basislinie einer Zellenreihe die Basislinie der ersten Textzeile der ersten Zelle ist. Bei leeren Zellen entspricht `baseline` dem Wert `bottom`.
`sub`	Tieferstellen des Elements ohne Veränderung der Schriftgröße.	Keine Auswirkung.
`super`	Höherstellen des Elements ohne Veränderung der Schriftgröße.	Keine Auswirkung.

Tabelle 7.4: Auswirkung der Werte der Element `vertical-align` auf Inline-Elemente und Tabellenzellen

Wert	Inline-Elemente	Tabellenzellen
top	Ausrichtung bündig mit der Oberkante der Zeilenbox. Dieser Wert hat nur eine Wirkung, wenn die Zeilenhöhe des Elternelements größer als normal eingestellt ist.	Ausrichtung der letzten Textzeile bündig mit der Oberkante der Tabellenzelle.
text-top	Ausrichtung am oberen Schriftrand. Dieser Wert hat nur dann eine sichtbare Wirkung, wenn der Schriftgrad des Elements vom Schriftgrad des Elternelements abweicht.	Keine Auswirkung.
middle	Richtet die vertikale Mitte der Textbox an einem Punkt aus, der etwa um 1/4 der Schriftgröße oberhalb der Basislinie des Elternelements liegt.	Ausrichtung der vertikalen Mitte der Textzeilen an der vertikalen Mitte der Tabellenzelle.
bottom	Ausrichtung bündig mit der Unterkante der Zeilenbox. Dieser Wert hat nur eine Wirkung, wenn die Zeilenhöhe des Elternelements größer als normal eingestellt ist.	Ausrichtung der letzten Textzeile bündig mit der Unterkante der Tabellenzelle.
text-bottom	Ausrichtung am unteren Schriftrand. Dieser Wert hat nur dann eine sichtbare Wirkung, wenn der Schriftgrad des Elements vom Schriftgrad des Elternelements abweicht.	Keine Auswirkung.
<Länge>	Neben den Schlüsselwörtern können Sie auch Längen oder Prozente angeben. Ein positiver Wert hebt die Basislinie des Texts an, ein negativer senkt sie ab. Die Werte 0 und 0% haben die gleiche Bedeutung wie baseline.	
<Prozent>		
inherit		

Tabelle 7.4: Auswirkung der Werte der Element vertical-align auf Inline-Elemente und Tabellenzellen (Fortsetzung)

Referenzdarstellung

Abbildung 7.15 zeigt, wie Werte sich bei Inlineboxen auswirken. Anhand dieser Referenzdarstellung können Sie vergleichen, ob Ihr Lieblingsbrowser alle Werte richtig interpretiert. Abbildung 7.16 zeigt die Auswirkung auf Tabellenzellen. Die Bezugslinie ist hierbei schwarz markiert. Besonders interessant ist, wie der Wert baseline interpretiert wird.

Browserunterstützung

Die Eigenschaft `vertical-align` wird von allen modernen Browsern unterstützt mit folgenden Einschränkungen:

IE bis Version 6 stellt die Werte `top` beziehungsweise `bottom` im Text genau so wie `text-top` beziehungsweise `text-bottom` dar.

IE hat gelegentlich Probleme mit der Ausrichtung von Grafiken im Text.

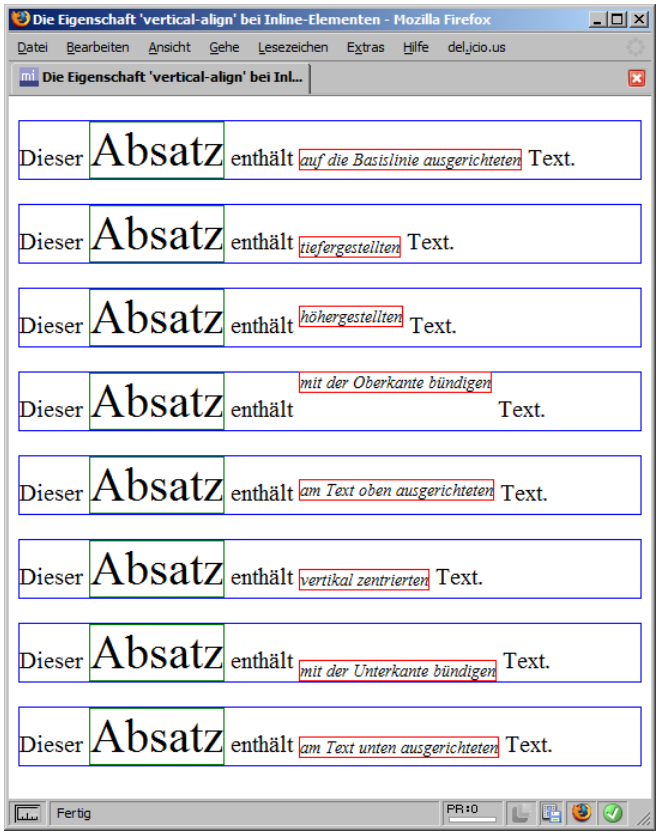


Abbildung 7.15: Die Eigenschaft `vertical-align` bei Inline-Elementen

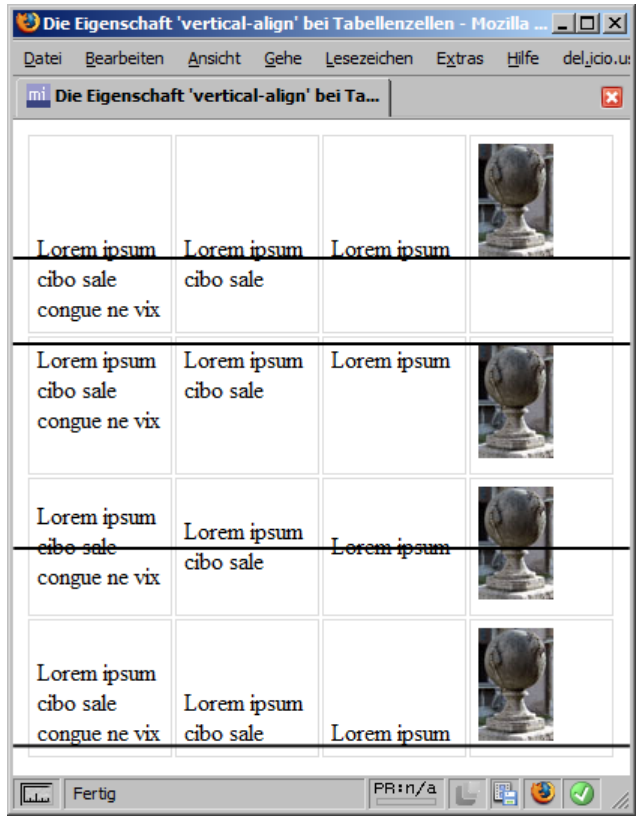


Abbildung 7.16: Die Auswirkungen der Eigenschaft `vertical-align` bei Tabellenzellen

7.6.2 Floatierte Boxen: Die Eigenschaft float

Boxen können an bestimmten Stellen aus dem normalen Fluss herausgelöst und ganz links oder ganz rechts am Rand des umschließenden Blocks platziert werden. In diesem Zusammenhang spricht man auch von *floatierter Positionierung*.

DEFINITION

*Die **Eigenschaft** `float` gibt an, ob ein Element nach links oder rechts floatieren soll.*

Werte	`none`, `left`, `right`, `inherit`
Initialwert	`none`
Vererbung	nein
Anwendung	nicht positionierte Elemente

Der Wert `none` ist der Initialwert, standardmäßig floatiert eine Box also nicht. Geben Sie hingegen `left` an, erzeugt das Element eine Blockbox, die nach links ausgerichtet wird. Nachfolgender Inhalt fließt an der rechten Seite der Box vorbei. Der Wert `right` erzeugt entsprechend eine Blockbox, die nach rechts ausgerichtet wird.

> HINWEIS
>
> *Wenn Sie eine Box floatieren lassen, sollten Sie stets auch eine Breite angeben (siehe die Eigenschaft `width`, Abschnitt 7.8.1). Floatierte Boxen schrumpfen auf eine minimale Breite zusammen, das heißt, sie werden so breit, wie das größte nicht umbrechbare Inhaltselement (beispielsweise ein langes Wort) es erfordert. Eine Ausnahme bilden sogenannte ersetzte Inline-Elemente, das heißt inzeilige Elemente, deren Dimensionen nicht berechnet werden müssen, sondern bereits bekannt sind. Dazu gehören Bilder (`img`) oder Objekte (`object`); diese benötigen keine explizite Breitenzuweisung.*

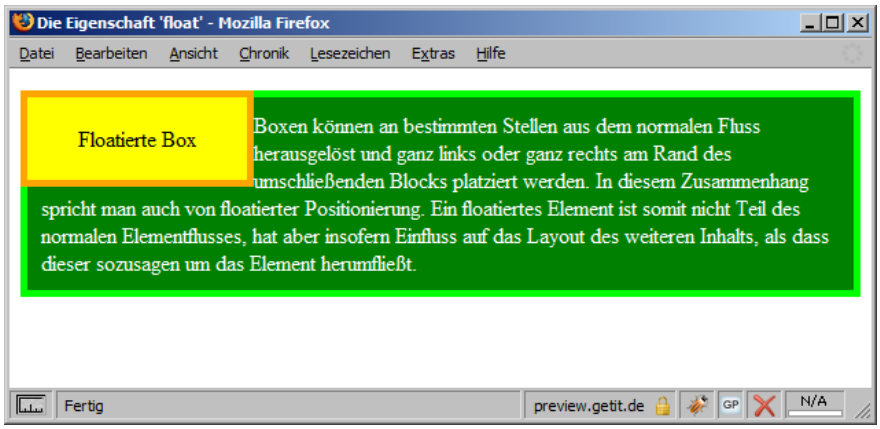


Abbildung 7.17: Floatierte Positionierung

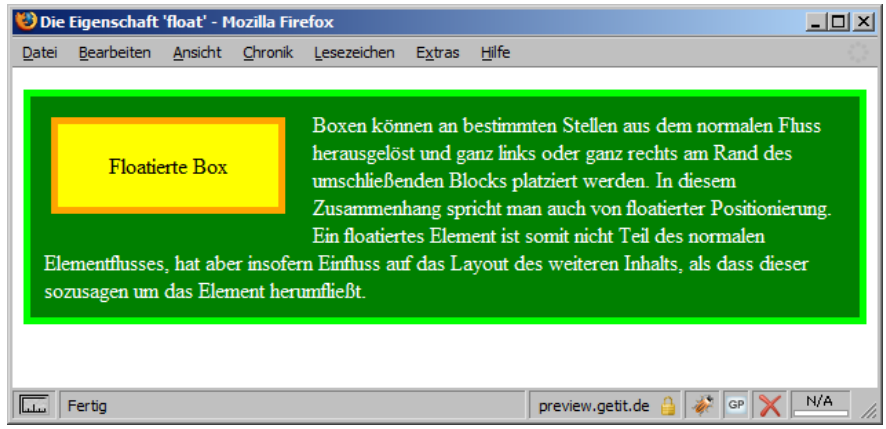


Abbildung 7.18: Die Rechteckform nachfolgender Elemente wird nicht verändert, lediglich die Zeilenboxen werden gekürzt

Ein floatiertes Element ist nicht Teil des normalen Elementflusses, hat aber insofern Einfluss auf das Layout des weiteren Inhalts, als dass dieser sozusagen um das Element herumfließt. Dabei gibt es jedoch einiges zu beachten:

- Damit Elementinhalte um floatierte Boxen herumfließen können, werden alle Zeilenboxen neben einem floatierten Element um den Betrag der Breite dieses Elements gekürzt.
- Die tatsächliche Breite eines Elements, das einem floatierten Element nachfolgt, wird nicht verändert, um die Rechteckform des Blocks beizubehalten. Dadurch richtet sich ein Hintergrundbild an der tatsächlichen Breite des Elements aus und wird nicht seitlich verschoben. Es kann folglich von dem floatierten Element überdeckt werden. Dies ist gut in Abbildung 7.18 zu erkennen: Die floatierte Box erhält einen Margin-Bereich von 20 Pixeln und rückt vom Rand weg, sodass Sie den Rahmen und den Hintergrund des dahinterliegenden Elements sehen können – es ist rechteckig und vollständig.
- Der Eigenschaft `display` floatierter Elemente wird stets die Eigenschaft `block` zugewiesen (es sei denn, `display` hat den Wert `none`), sodass diese immer den Status eines Blockelements erhalten.

Die Eigenschaft `float` kann nicht auf absolut oder fest positionierte Elemente und generierten Content angewendet werden. Zudem unterliegt eine floatierte Box nachfolgend erläuterten Regeln.

Regeln für floatierte Boxen

- Bei der Platzierung einer floatierten Box werden nicht nur die Inhalte, sondern auch Padding-, Border- und Margin-Bereiche voll berücksichtigt; Randbereiche fallen nicht zusammen.
- Eine floatierte Box wird so weit nach links oder rechts verschoben, bis sie entweder an die Kante des umschließenden Blocks oder eines anderen floatierten Elements stößt. Dadurch liegen floatierte Boxen so lange nebeneinander, bis nicht mehr ausreichend Platz vorhanden ist und das Element in die nächste Zeile verschoben wird.
- Eine floatierte Box darf mit seiner Oberkante nicht höher platziert werden als ein anderes floatiertes Element oder Text, das beziehungsweise der im Dokument vor diesem notiert ist.

Floatierte Boxen werden aus dem Elementfluss herausgenommen. Was das bedeutet, zeigt Abbildung 7.19. Das Element, das die floatierte Box umschließt, enthält nun keinen Inhalt mehr. Die floatierte Box hat keinen Einfluss auf die Höhe des Elements, sodass dieses auf eine minimale Höhe, die sich aus Rahmen und Padding addiert, zusammenfällt. Um dieses Verhalten zu vermeiden, können Sie die Eigenschaft `overflow` einsetzen (siehe Kapitel 7.9.2).

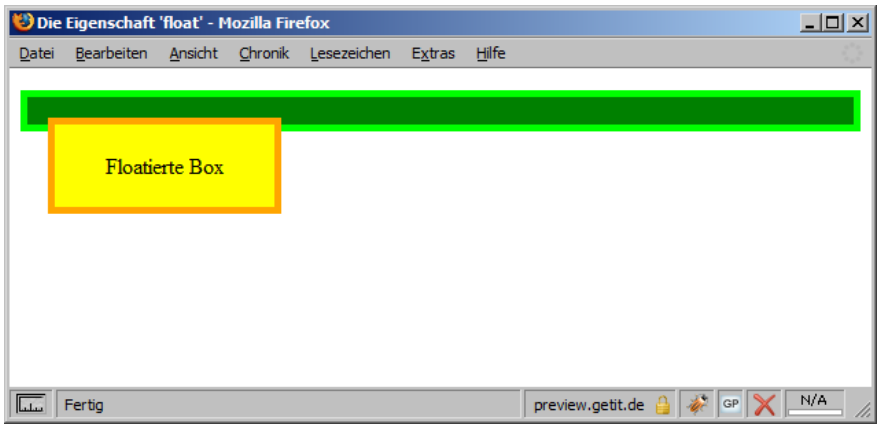


Abbildung 7.19: Floatierte Elemente werden aus dem Elementfluss herausgenommen

Browserunterstützung

Die Eigenschaft `float` wird von allen modernen Browsern unterstützt mit folgender Einschränkung:

Probleme treten häufig bei komplizierteren Layouts auf. Dort sollten Sie genau auf das Zusammenspiel mit geschachtelten oder absolut positionierten Boxen sowie der Eigenschaften `float` und `clear` achten. Vor allem das hasLayout-Konzept stellt eine besondere Herausforderung für Webautoren dar.

7.6.3 Steuerung des Elementflusses um floatierte Boxen: Die Eigenschaft clear

In vielen Situationen soll der Elementfluss um floatierte Boxen unterbrochen werden, das heißt, ein nachfolgendes Element soll nicht neben floatierten Boxen stehen, sondern darunter: die Fußzeile einer Webseite beispielsweise, die diese abschließen und daher stets an ihrem Ende dargestellt werden soll, oder eine umrahmte Box, deren Rahmen nicht durch das floatierte Element unterbrochen beziehungsweise überdeckt werden soll. Um den Fluss um ein floatiertes Element zu unterbrechen, gibt es die Eigenschaft `clear`.

DEFINITION

*Die **Eigenschaft** `clear` gibt an, auf welchen Seiten eines Elements keine vorhergehenden floatierten Elemente dargestellt werden dürfen. Sie ist nur auf Blockelemente anwendbar.*

Werte	`none`, `left`, `right`, `both`, `inherit`
Initialwert	`none`
Vererbung	nein
Anwendung	Blockelemente

Elemente, für die der Wert `left` angegeben ist, unterbrechen die Floatierung nach links, das heißt, sie stehen stets unterhalb von linksfloatierten Elementen. Entsprechend unterbrechen Elemente, für die `clear` auf `right` steht, die Floatierung nach rechts. Der Wert `both` unterbricht die Floatierung auf beiden Seiten.

Abbildung 7.20 zeigt die Anwendung der Eigenschaft `clear` auf einen Absatz. Der Absatz steht nicht neben, sondern unter der floatierten Box.

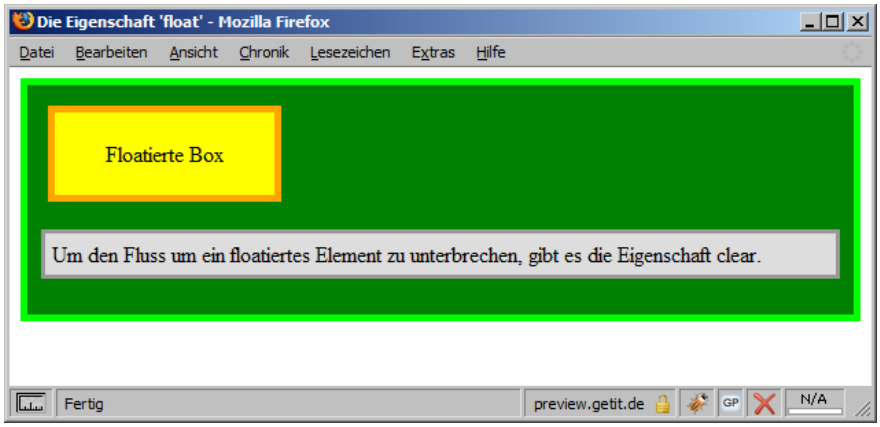


Abbildung 7.20: Die Eigenschaft `clear` unterbricht die Floatierung

Zusammenspiel von `clear` und `float`

Die Eigenschaften `float` und `clear` lassen sich auch zusammen auf ein Element anwenden, wie in Listing 7.3 zu sehen. Für Box 3 wird der Elementfluss um linksfloatierte Elemente aufgehoben, daher steht das Element unterhalb der Boxen 1 und 2. Box 4 floatiert zwar rechts wie Box 3, steht aber unter dieser, da gleichzeitig der Elementfluss um rechtsfloatierte Elemente aufgehoben wird.

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

<html xmlns="http://www.w3.org/1999/xhtml"
  lang="de" xml:lang="de">
  <head>
    <meta http-equiv="Content-Type"
      content="text/html; charset=utf-8" />
    <title>Die Eigenschaft 'clear'</title>

    <style type="text/css">
      /* <![CDATA[ */
      div {
        padding: .5em;
        margin: .5em;
        border: 2px solid black;
        color: black;
        text-align: justify;
      }
      #box1 {
        background-color: #ff3;
        float: left;
      }
      #box2 {
        background-color: #ff9;
```

```
        float: left;
      }
      #box3 {
        background-color: #ffb;
        float: right;
        clear: left;
      }
      #box4 {
        background-color: #ffd;
        float: right;
        clear: right;
      }
      /* ]]> */
    </style>
  </head>

  <body>
    <div id="box1">Box 1</div>
    <div id="box2">Box 2</div>
    <div id="box3">Box 3</div>
    <div id="box4">Box 4</div>
  </body>
</html>
```

Listing 7.3: Zusammenspiel der Eigenschaften `clear` und `float`

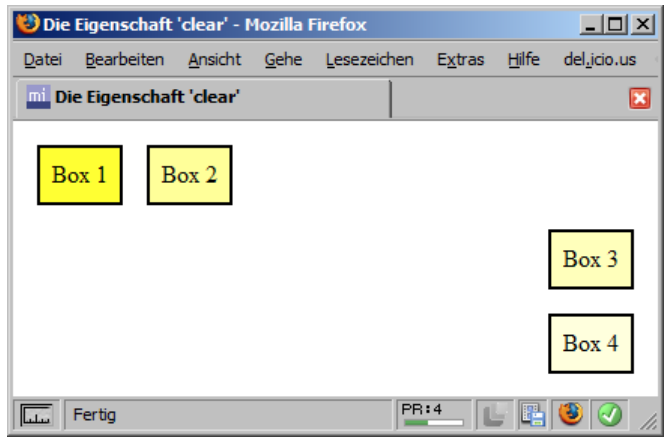


Abbildung 7.21: Zusammenspiel der Eigenschaften `clear` und `float`

Browserunterstützung

Die Eigenschaft `clear` wird von allen modernen Browsern unterstützt mit folgender Einschränkung:

Probleme treten häufig bei komplizierteren Layouts im Zusammenspiel der Eigenschaften `clear` und `float` auf (siehe oben).

7.6.4 YAML

YAML ist ein (X)HTML/CSS-Framework zur Erstellung moderner und flexibler Layouts auf Grundlage von `float`-Umgebungen, konzipiert und entwickelt von Dirk Jesse. Dabei stehen ein möglichst hohes Maß an Flexibilität für den Webentwickler und Zugänglichkeit für die Nutzer im Vordergrund.

YAML wurde als Basis für die Entwicklung flexibler Layouts entworfen. Einer der Schwerpunkte liegt damit bei den Anforderungen, die sich aus der Arbeit mit variablen Größenangaben und der Verwendung unterschiedlicher Maßeinheiten ergeben. YAML bietet Ihnen folgende Vorteile:

- auf Flexibilität und Barrierefreiheit ausgelegtes Layoutkonzept zur Erstellung von Spalten- und Grid-basierten CSS-Layouts,
- umfassende Browserkompatibilität (ab IE 5.0/Win) für ein einheitliches und fehlerfreies Erscheinungsbild des Layouts in allen Situationen,
- größtmögliche Gestaltungsfreiheit für den Webdesigner (fixe und flexible Layouts, variable Spaltenbreiten),
- beliebige Anordnung der Inhalte im Quelltext bei spaltenbasierten Layouts,
- effizientes Arbeiten durch funktional gegliederte Stylesheetvorlagen,
- vielseitig einsetzbare, flexible und verschachtelbare Grid-Bausteine.

Mit der grundlegenden, spaltenbasierten Quelltextstruktur können Sie Designs mit ein bis drei Spalten (mit fixen oder flexiblen Breiten) innerhalb sehr kurzer Zeit entwickeln. Über Subtemplates (flexible Grid-Bausteine) können Sie das Spaltensystem nahezu beliebig erweitern oder alternativ zur Erstellung von Grid-Layouts verwenden. Das über den standardmäßig vorgegebenen Quelltext definierte Basislayout enthält ebenfalls Container, die die Festlegung der Layoutbreite vereinfachen und zum Beispiel grafische Umrandungen des Layouts ermöglichen.

Unter *http://www.yaml.de* können Sie sich das Framework genauer anschauen und herunterladen.

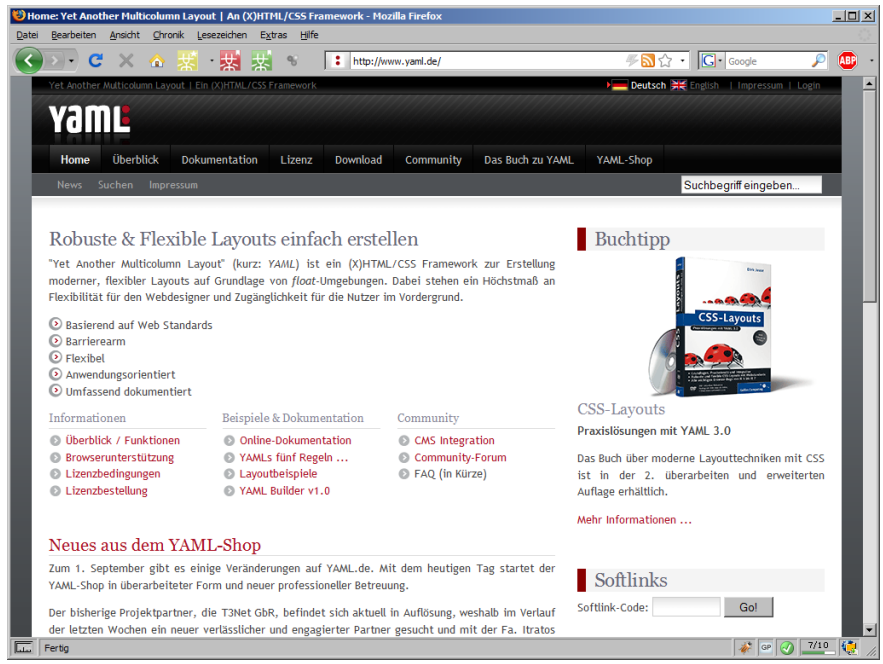


Abbildung 7.22: YAML-Startseite

Nach der Lektüre dieses Buchs sollten Sie in der Lage sein, YAML im Rahmen eigener Projekte verwenden und auf Ihre eigenen Bedürfnisse anpassen zu können. Ich erwähne YAML an dieser Stelle, da ich kein anderes Framework kenne, das auf einem Layout aufbaut, das so intensiv Gebrauch von der Eigenschaft `float` macht und dabei browserübergreifend so stabil ist. Herausragend ist auch die Dokumentation[2]: Vor allem den Artikel »Die Funktionsweise von floats im Detail«[3] sollten Sie sich zur weiteren Vertiefung zu Gemüte führen.

7.7 Positionierung

7.7.1 Positionierung: Die Eigenschaft position

DEFINITION

*Die **Eigenschaft** `position` legt fest, auf welche Art und Weise die Position einer Box berechnet wird.*

2 *http://www.yaml.de/de/dokumentation.html*
3 *http://www.yaml.de/de/dokumentation/grundlagen/funktionsweise-von-floats.html*

Werte	`static`, `relative`, `absolute`, `fixed`, `inherit`
Initialwert	`static`
Vererbung	nein
Anwendung	global

Die Werte der Eigenschaft `position` sind am einfachsten anhand von Beispielen zu erklären.

Statische Positionierung: `static`

Bei der Box handelt es sich um eine normale Box gemäß dem gewöhnlichen Fluss. `static` ist der Initialwert der Eigenschaft `position`, was bedeutet, dass jede Box, für die Sie keine andere Positionierungsart definieren, statisch positioniert ist.

Man bezeichnet ein Element als positioniert, wenn dessen Eigenschaft `position` einen anderen Wert als `static` annimmt.

Relative Positionierung: `relative`

Die Box wird *relativ zu ihrer statischen Position* verschoben.

Das positionierte `div`-Element in folgendem Beispiel wird im Verhältnis zu seiner gewöhnlichen Position 50 Pixel nach unten und 70 Pixel nach rechts positioniert. Die dem positionierten Element nachfolgenden Elemente werden so berechnet, als wäre das Element nicht positioniert worden. Daher kommt es in diesem Beispiel zu einer Überlappung beider Boxen.

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

<html xmlns="http://www.w3.org/1999/xhtml"
  lang="de" xml:lang="de">
  <head>
    <meta http-equiv="Content-Type"
      content="text/html; charset=utf-8" />
    <title>Relative Positionierung</title>

    <style type="text/css">
      /* <![CDATA[ */
      div {
        width: 200px;
        height: 100px;
        padding: .5em;
        border: 2px solid black;
```

```
        color: black;
        background-color: #ffd
      }
      div#box1 {
        background-color: #ffb;
        position: relative;
        top: 50px;
        left: 70px;
      }
      /* ]]> */
    </style>
  </head>

  <body>
    <div id="box1">Box 1</div>

    <div id="box2">Box 2</div>
  </body>
</html>
```

Listing 7.4: Relative Positionierung

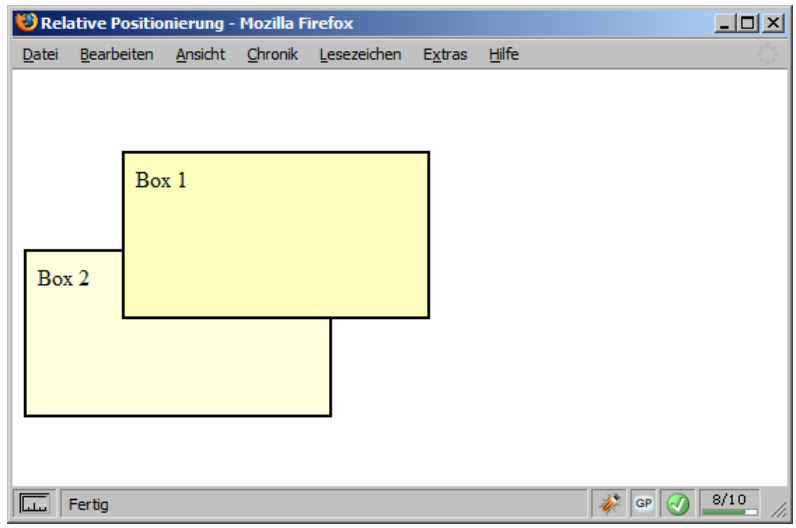


Abbildung 7.23: Relativ positionierte Elemente können andere Elemente überlappen

Absolute Positionierung: absolute

Die Box wird *relativ zu ihrem umschließenden Block* verschoben. Dabei gilt es Folgendes zu beachten: Der umschließende Block für eine absolut positionierte Box wird durch das nächste positionierte Elternelement oder, falls es kein sol-

ches gibt, durch den äußersten umschließenden Block gebildet. Im Allgemeinen können Sie davon ausgehen, dass ein Element relativ zum Viewport, also zum Browserfenster, ausgerichtet wird, wenn kein positioniertes Vorfahrenelement gefunden wird.

Beispiel

Anhand von Listing 7.5 ist dies gut zu erkennen. Die dritte und die vierte Box sind absolut positioniert. Die dritte Box befindet sich innerhalb der relativ positionierten zweiten Box und wird daher relativ zu dieser ausgerichtet. Die vierte Box hat kein positioniertes Vorfahrenelement und wird daher relativ zum Viewport positioniert, also anhand der rechten unteren Ecke des Browserfensters.

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

<html xmlns="http://www.w3.org/1999/xhtml"
  lang="de" xml:lang="de">
  <head>
    <meta http-equiv="Content-Type"
      content="text/html; charset=utf-8" />
    <title>Absolute Positionierung</title>

    <style type="text/css">
      /* <![CDATA[ */
      div {
        width: 200px;
        height: 100px;
        padding: .5em;
        border: 2px solid black;
        color: black;
      }
      #box1 {
        background-color: #ffd;
        /* statisch positioniert */
      }
      #box2 {
        background-color: #ffb;
        /* relativ positioniert */
        position: relative;
        top: 50px;
        left: 50px;
      }
      #box3 {
        background-color: #ff6;
        /* absolut positioniert */
        position: absolute;
        top: 50px;
        left: 70px;
```

```
        }
        #box4 {
          background-color: #ff3;
          /* absolut positioniert */
          position: absolute;
          bottom: 25px;
          right: 25px;
        }
        /* ]]> */
      </style>
    </head>

    <body>
      <div id="box1">Box 1 (<code>static</code>)</div>
      <div id="box2">
        Box 2 (<code>relative</code>)
        <div id="box3">Box 3 (<code>absolute</code>)</div>
      </div>
      <div id="box4">Box 4 (<code>absolute</code>)</div>
    </body>
  </html>
```

Listing 7.5: Absolute Positionierung

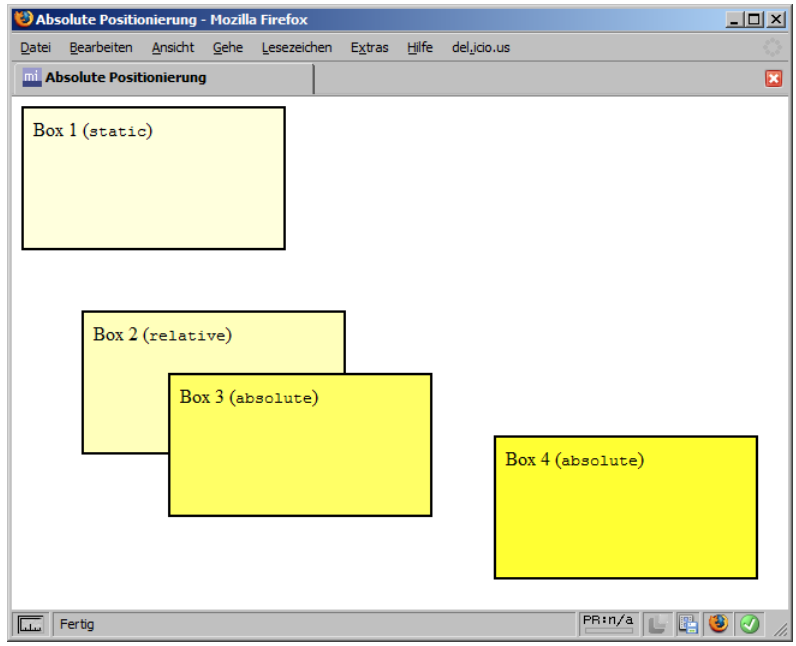


Abbildung 7.24: Absolute Positionierung bietet viele Möglichkeiten

Absolut positionierte Elemente werden aus dem gewöhnlichen Elementfluss herausgenommen, deren Boxen haben also keinerlei Einfluss auf die Darstellung der nachfolgenden gleichrangigen Elemente; Überlappungen sind damit vorprogrammiert.

Feste Positionierung: fixed

Die Box wird *absolut positioniert*, also relativ zu ihrem umschließenden Block verschoben, und ist darüber hinaus *feststehend hinsichtlich einer bestimmten Referenz*. Auf dem Bildschirm bedeutet dies, dass die Box beim Scrollen der Seite stehen bleibt; beim Druck wird das Element auf jeder gedruckten Seite an der gleichen Stelle wiederholt.

Workaround für veraltete Browser

In aktuellen Browsern wird die feste Positionierung mittlerweile sehr gut unterstützt, in alten Browsern jedoch nicht oder sehr fehlerhaft (siehe unten). Hier hilft ein schneller Griff in die CSS-Trickkiste: Webautoren können sich zunutze machen, dass Browser, die Probleme mit fester Positionierung haben, den Attributselektor nicht verstehen. Das Element mit der ID `box1` wird zunächst absolut positioniert, würde aber mit dem Rest der Seite wegscrollen. Für alle Browser, die den Attributselektor korrekt interpretieren, überschreiben Sie `position: absolute;` mit `position: fixed;`. Nicht standardkonformen Browsern wird somit die Deklaration, die sie ohnehin nicht verstehen, vorenthalten, sodass sie das Element zumindest korrekt positionieren, während standardkonforme Browser das Element nun fest positionieren.

```
#box1 {
  position: absolute;
  top: 10px;
  right: 10px;
}
#box1[id] {
  position: fixed;
}
```

Browserunterstützung

Die Eigenschaft `position` wird von allen modernen Browsern unterstützt mit folgenden Einschränkungen:

IE bis Version 6 erkennt den Wert `fixed` nur am Element `body`. An allen anderen Elementen angewendet, kann dieser Wert unvorhersehbare Auswirkungen haben.

7.7.2 Boxabstände: Die Eigenschaften top, right, bottom und left

DEFINITION

*Die **Eigenschaften** `top`, `right`, `bottom` und `left` geben an, wie weit die obere beziehungsweise rechte, untere oder linke Kante einer Box von der entsprechenden Kante des umschließenden Blocks entfernt ist. Positive Werte zählen vom Rand des umschließenden Blocks nach innen, negative Werte zählen nach außen.*

Werte	`auto`, `<Länge>`, `<Prozent>`, `inherit`
Initialwert	`auto`
Vererbung	nein
Anwendung	positionierte Elemente

Standardmäßig wird die Position der entsprechenden Kante automatisch berechnet (`auto`). Die Berechnung ist von mehreren anderen Angaben abhängig, die die vertikalen beziehungsweise horizontalen Abstände und Dimensionen betreffen:

- Die Summe aus `top`, `margin-top`, `border-top-width`, `padding-top`, `height`, `padding-bottom`, `border-bottom-width`, `margin-bottom` und `bottom` muss gleich der Höhe des umgebenden Blocks sein.
- Die Summe aus `left`, `margin-left`, `border-left-width`, `padding-left`, `width`, `padding-right`, `border-right-width`, `margin-right` und `right` muss gleich der Breite des umgebenden Blocks sein.

Wenn Sie eine Länge angeben, wird das positionierte Element exakt um den entsprechenden Wert verschoben.

Prozentwerte werden bei `left` und `right`

- für absolut und fest positionierte Elemente relativ zur Breite des umschließenden Blocks und
- für relativ positionierte Elemente relativ zur Breite des Elements selbst

berechnet. Für `top` und `bottom` gilt dies entsprechend relativ zur Höhe.

Browserunterstützung

Die Eigenschaften `top`, `left`, `bottom` und `right` werden von allen modernen Browsern unterstützt mit folgender Einschränkung:

Wird ein Prozentwert deklariert, verschiebt IE bis Version 6 die Box unter Umständen um einen Betrag, der größer ist als der aus den Abmessungen des umschließenden Blocks und des gegebenen Werts errechnete. Darüber hinaus wird der Wert `inherit` ignoriert.

7.7.3 Angabe der Ebene: Die Eigenschaft z-index

Stapelebene

In CSS hat jede Box drei Dimensionen, von denen im Rahmen dieser Einführung bislang nur zwei betrachtet wurden: die horizontale und die vertikale Position (x-Achse und y-Achse). Darüber hinaus liegen Boxen jedoch auch entlang einer z-Achse über- oder untereinander gestapelt. So wie transparente Folien auf einem Projektor Schicht für Schicht übereinandergelegt werden können, so können Elemente in verschiedenen Stapelebenen liegen und sich so gegenseitig überlappen oder überdecken. Wenn es weiter oben heißt, absolut positionierte Elemente würden aus dem Elementfluss herausgehoben, so bedeutet dies, sie stehen in einer höheren Stapelebene.

Jede Box in einem bestimmten Stapelkontext hat eine ganzzahlige Stapelebene, die ihre Position auf der z-Achse relativ zu anderen Boxen angibt. Boxen in einer höheren Stapelebene überlappen Boxen in einer niedrigeren Stapelebene. Boxen auf derselben Stapelebene werden von unten nach oben der Reihenfolge im Dokumentbaum entsprechend gestapelt.

DEFINITION

*Die **Eigenschaft** `z-index` gibt die Stapelebene positionierter Boxen im Stapelkontext an und ob die Box einen lokalen Stapelkontext einrichtet.*

Werte	`auto`, `<Integer>`, `inherit`
Initialwert	`auto`
Vererbung	nein
Anwendung	positionierte Elemente

Standardmäßig weist der Browser Elementen nach oben beschriebenen Regeln automatisch (`auto`) einen Wert zu. Wenn Sie ein `<Integer>` angeben, errechnet der Browser den aktuellen und den lokalen Stapelkontext neu.

Stapelkontext

- Der **aktuelle Stapelkontext**, das heißt der Stapelkontext der Box, für die `z-index` angegeben ist, wird berechnet, wobei Elemente mit höherem `z-index` Elemente mit niedrigerem `z-index` überlappen. Negative Werte sind erlaubt.
- Wenn nicht nur gleichrangige Elemente mit einem `z-index` versehen sind, sondern auch deren Nachkommenelemente, stehen diese in einem **lokalen Stapelkontext**. Zunächst werden also die gleichrangigen Elemente entsprechend ihrem `z-index` ausgerichtet und erst anschließend ihre Nachkommenelemente. Anders ausgedrückt: Wenn ein Element aufgrund seines `z-index` über seinem Nachbarelement dargestellt wird, werden auch alle seine Nachkommenelemente über den Nachkommenelementen des Nachbarelements dargestellt.

Beispiel

In Listing 7.6 sind Regeln für sechs `div`-Elemente definiert, an denen Sie deutlich das Prinzip des lokalen Schichtungskontexts erkennen können. Box A hat einen größeren `z-index` als Box B und überlappt diese und alle ihre Nachfahren daher. Zusätzlich stehen die Boxen 1 und 2 als Kinder der Box A ebenfalls vor Box B und allen ihren Kindern, auch wenn Box 1 einen geringeren `z-index` aufweist als die Boxen B, 3 und 4.

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

<html xmlns="http://www.w3.org/1999/xhtml"
  lang="de" xml:lang="de">
  <head>
    <meta http-equiv="Content-Type"
      content="text/html; charset=utf-8" />
    <title>Die Eigenschaft 'z-index'</title>

    <style type="text/css">
      /* <![CDATA[ */
      div {
        width: 237px;
        height: 150px;
        padding: .5em;
        border: 2px solid black;
        color: black;
        background-color: #ffd;
        position: absolute;
      }
      div div {
        width: 95px;
        height: 90px;
        border: 1px solid black;
        background-color: #ffb;
      }
      #boxA {
```

```
          top: 100px;
          left: 80px;
          z-index: 6;
        }
        #box1 {
          top: 35px;
          left: 10px;
          z-index: 1;
        }
        #box2 {
          top: 35px;
          left: 130px;
          z-index: 5;
        }
        #boxB {
          top: 10px;
          left: 10px;
          z-index: 3;
        }
        #box3 {
          top: 35px;
          left: 10px;
          z-index: 2;
        }
        #box4 {
          top: 35px;
          left: 130px;
          z-index: 4;
        }
        /* ]]> */
      </style>
    </head>

    <body>
      <div id="boxA">
        Box A (<code>z-index: 6</code>)
        <div id="box1">Box 1 (<code>z-index: 1</code>)</div>
        <div id="box2">Box 2 (<code>z-index: 5</code>)</div>
      </div>

      <div id="boxB">
        Box B (z-index: 3)
        <div id="box3">Box 3 (<code>z-index: 2</code>)</div>
        <div id="box4">Box 4 (<code>z-index: 4</code>)</div>
      </div>
    </body>
  </html>
```

Listing 7.6: Die Eigenschaft `z-index`

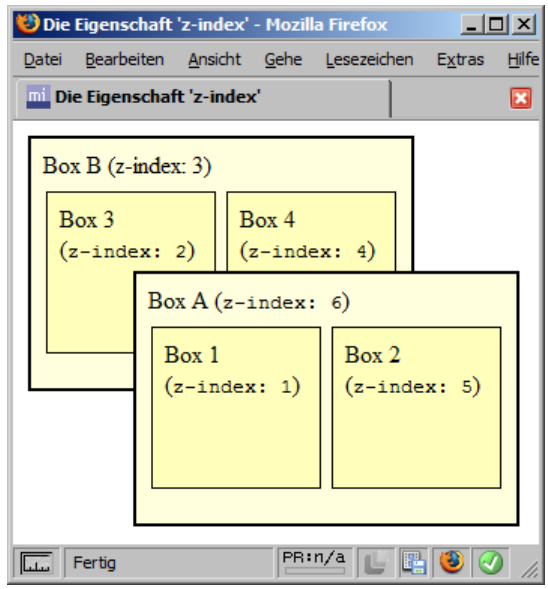


Abbildung 7.25: Wenn ein Element aufgrund seines `z-index` über seinem Nachbarelement dargestellt wird, werden auch alle seine Nachkommenelemente über den Nachkommenelementen des Nachbarelements dargestellt

Browserunterstützung

Alle Browser – Die Eigenschaft `z-index` wird von allen modernen Browsern unterstützt.

7.8 Breite und Höhe

7.8.1 Content-Breite: Die Eigenschaft width

DEFINITION

Die ***Eigenschaft*** `width` *gibt die Breite der Contentbox eines Blockelements oder eines solchen Elements an, dessen Dimensionen in CSS bereits bekannt sind (sogenannte* replaced elements *wie* `object`, `textarea` *oder* `img`*).*

Werte	`auto`, `<Länge>`, `<Prozent>`, `inherit`
Initialwert	`auto`
Vererbung	nein
Anwendung	alle Elemente außer Inline-Elemente, Tabellenzeilen und Zeilengruppen

Sie können die Breite eines Elements durch Angabe einer Länge oder eines Prozentwerts bestimmen. Der Prozentwert wird relativ zur Breite des umschließenden Blocks berechnet, wenn diese explizit angegeben ist, ansonsten wird der Wert als `auto` interpretiert und in Abhängigkeit von verschiedenen Faktoren berechnet, die im Folgenden erläutert werden.

Browserunterstützung

Die Eigenschaft `width` wird von allen modernen Browsern unterstützt mit folgenden Einschränkungen:

Wenn ein Prozentwert deklariert wurde, vererbt die Gecko-Engine den deklarierten Wert, nicht den daraus resultierenden berechneten Wert.

IE vererbt bis Version 7 ebenfalls keine Prozentwerte und kann darüber hinaus den Wert `inherit` nicht erkennen.

Breite und Ränder berechnen

Die tatsächliche Breite eines Elements ist nicht nur von der Eigenschaft `width` abhängig, sondern auch von der Art der Box, den Eigenschaften `padding-right`, `padding-left`, `border-right`, `border-left`, `margin-right` und `margin-left` sowie von der Standardkonformität des verwendeten Browsers.

Grundsätzlich gilt, dass Blockelemente

Blockelemente

- stets den vollen horizontal zur Verfügung stehenden Raum einnehmen, wenn nichts anderes deklariert wurde, und
- sich ihre Gesamtbreite aus Content-Breite plus linke und rechte Padding-, Border- und Margin-Bereiche zusammensetzt.

Folgende Regel sorgt dafür, dass `div`-Elemente einen Raum von 502 Pixeln einnehmen:

Beispiel

```
div {
  width: 400px;
  border: 1px solid; /* 1px oben, rechts, unten und links */
  padding-left: 20px;
  padding-right: 20px;
  margin-left: 30px;
  margin-right: 30px;
}
```

Inline-Elemente verhalten sich anders als Blockelemente, auch in Bezug auf die berechnete Breite. Die Eigenschaft `width` wird auf Inline-Elemente *nicht* angewendet. Die berechnete Breite ist somit 0 beziehungsweise so groß, wie der Inhalt des Elements es erfordert. Die Angabe

Inline-Elemente

```
span { width: 300px; }
```

bewirkt überhaupt nichts. Eine Ausnahme bilden sogenannte ersetzte Inline-Elemente, beispielsweise Bilder (`img`) oder Objekte (`object`), die Angaben zu Breite und Höhe bereits enthalten. Auf ersetzte Inline-Elemente wird die Eigenschaft `width` angewandt. Die Breite eines Bilds lässt sich mit

```
img { width: 280px; }
```

durchaus beeinflussen.

Blockelemente horizontal zentrieren

Vorüberlegungen

Wenn Sie über die Eigenschaft `width` eine Breite festlegen, die kleiner ist als der zur Verfügung stehende horizontale Raum, und zugleich `margin-left` und `margin-right` auf den Wert `auto` setzen, wird das Element horizontal zentriert. Dieses Verhalten ist einleuchtend, wenn Sie sich zwei Dinge vor Augen führen:

- Ein Blockelement füllt standardmäßig den vollen zur Verfügung stehenden Raum aus. Beträgt dieser beispielsweise 1.000 Pixel und setzen Sie die Breite eines Elements auf 400 Pixel, bleiben – davon ausgehend, dass `border` und `padding` auf 0 stehen – 600 Pixel Randbereich übrig.
- Mit der Eigenschaft `auto` für den Wert `margin-left` oder `margin-right` legen Sie fest, dass der Randbereich den Rest des zur Verfügung stehenden Raums einnehmen soll. Wenn beide Eigenschaften auf `auto` gesetzt werden, teilen sich linker und rechter Rand diesen Raum, in diesem Fall jeweils 300 Pixel.

Workaround für veraltete Browser

Es wird Sie wohl kaum überraschen zu lesen, dass nicht standardkonforme Browser, also auch Browser im Quirks Mode, den Wert `auto` ignorieren. Dafür wenden sie die Eigenschaft `text-align` fälschlicherweise auch auf Blockelemente an, sodass Webautoren ein wenig tricksen können. In Listing 7.7 wird für `div`-Elemente eine Breite von 30 Prozent festgelegt und `margin` zugleich auf `auto` gesetzt. Für standardkonforme Browser würden diese Angaben reichen, um `div`-Elemente horizontal zu zentrieren. Fehlerhafte Browser hingegen benötigen die Angabe `text-align: center;` für den umschließenden Block – in diesem Beispiel das `body`-Element. Um zu verhindern, dass nun auch die Textinhalte zentriert werden, wird der Universalselektor dazu verwendet, alle Kinder von `body` (eigentlich: die Textinhalte aller Kinder von `body`) linksbündig zu setzen.

```
div {
  color: black;
  background-color: lime;
  width: 30%;
  margin: auto; /* Reicht eigentlich */
}
```

```
/* Für kaputte Browser */
body {
  /* Alle Inhalte horizontal zentrieren */
  text-align: center;
}
body * {
  /* Textausrichtung korrigieren */
  text-align: left;
}
```

Listing 7.7: Blockelemente horizontal zentrieren

> > > HINWEIS

Es ist durchaus legitim, auf fehlerhafte, veraltete Browser keine Rücksicht zu nehmen, solange der Nutzer dadurch nicht merklich eingeschränkt wird. Ob Blöcke nun zentriert dargestellt werden oder nicht, ist eine kosmetische Frage und hat keinen Einfluss auf die Gebrauchstauglichkeit der Seite.

7.8.2 Mindestbreite: Die Eigenschaft min-width

• • • DEFINITION

*Die **Eigenschaft** `min-width` legt die Mindestbreite der Contentbox eines Blockelements oder eines solchen Elements fest, dessen Dimensionen in CSS bereits bekannt sind.*

Werte	`<Länge>`, `<Prozent>`, `inherit`
Initialwert	browserabhängig
Vererbung	nein
Anwendung	alle Elemente außer Inline- und Tabellenelemente

Prozentwerte werden relativ zur Breite des umschließenden Blocks berechnet.

Browserunterstützung

Die Eigenschaft `min-width` wird von allen modernen Browsern unterstützt mit folgenden Einschränkungen:

Der IE kennt diese Eigenschaft bis Version 6 nicht.

IE 7 missachtet `min-width`, wenn gleichzeitig zu breite `margin` und `width: auto` gegeben sind.

Der Apple-Browser stellt diese Eigenschaften nur für Elemente, die nicht positioniert sind, korrekt dar.

7.8.3 Maximalbreite: Die Eigenschaft max-width

DEFINITION

*Die **Eigenschaft** `max-width` legt die Maximalbreite der Contentbox eines Blockelements oder eines solchen Elements fest, dessen Dimensionen in CSS bereits bekannt sind.*

Werte	`<Länge>`, `<Prozent>`, `none`, `inherit`
Initialwert	browserabhängig
Vererbung	nein
Anwendung	alle Elemente außer Inline- und Tabellenelemente

Der Initialwert ist browserabhängig. Bei Angabe des Werts `none` wird keine Maximalbreite festgelegt. Prozentwerte werden relativ zur Breite des umschließenden Blocks berechnet.

Browserunterstützung

Die Eigenschaft `max-width` wird von allen modernen Browsern unterstützt mit folgenden Einschränkungen:

Der IE kennt diese Eigenschaft bis Version 6 nicht.

Der Apple-Browser stellt diese Eigenschaften nur für Elemente, die nicht positioniert sind, korrekt dar.

7.8.4 Content-Höhe: Die Eigenschaft height

DEFINITION

*Die **Eigenschaft** `height` gibt die Höhe der Contentbox eines Blockelements oder eines solchen Elements an, dessen Dimensionen in CSS bereits bekannt sind.*

Werte	`<Länge>`, `<Prozent>`, `auto`, `inherit`
Initialwert	`auto`
Vererbung	nein
Anwendung	alle Elemente außer Inline-Elemente, Tabellenspalten und Spaltengruppen

Sie können die Höhe eines Elements analog zur Breite (siehe die Eigenschaft `width`, Abschnitt 7.8.1) durch Angabe einer Länge oder eines Prozentwerts bestimmen. Standardmäßig wird die Höhe in Abhängigkeit von verschiedenen Faktoren berechnet, die im Folgenden erläutert werden.

Browserunterstützung

Die Eigenschaft `height` wird von allen modernen Browsern unterstützt mit folgenden Einschränkungen:

Der IE kann bis Version 7 nichts mit dem Wert `inherit` anfangen.

Wenn die Höhe des umschließenden Blocks nicht explizit angegeben und die Höhe des aktuellen Elements in Prozent definiert ist, verwendet Opera 8 nicht den Wert `auto`, sondern ermittelt daraus einen absoluten Höhenwert für das aktuelle Element.

Höhe und Ränder berechnen

Vorüberlegungen

Wie die tatsächliche Breite eines Elements ist auch dessen tatsächliche Höhe unter anderem von der Art der Box und deren Padding-, Border- und Margin-Bereichen abhängig. Zunächst ist davon auszugehen, dass Blockelemente

- stets nur so hoch sind, wie es die Boxen, die sie enthalten, erfordern, wobei fließende und absolut positionierte Boxen ignoriert werden, und
- sich ihre Gesamthöhe aus der Content-Höhe zuzüglich der oberen und unteren Padding-, Border- und Margin-Bereiche zusammensetzt.

Beispiel

Die verschiedenen Browser verhalten sich unterschiedlich, wenn Sie die Höhe einer Box mittels `height` einschränken. Im folgenden Beispiel wird die Höhe des `div`-Elements auf 50 Pixel und die Breite zugleich auf 200 Pixel beschränkt. Dadurch ist die Box kleiner, als es ihr Inhalt (bei Normalschriftgröße) erfordert. Während einige Browser wie ältere Versionen des Internet Explorers die Box vertikal auf die Höhe erweitern, die der Elementinhalt benötigt, halten sich andere Browser wie Firefox an die vorgegebenen Werte und lassen den Inhalt aus der Box hinauslaufen (siehe Abbildung 7.26).

```
div {
  color: black;
  background-color: lime;
  width: 200px;
  height: 50px;
}
```

Analog zur Eigenschaft `width` wird `height` auf Inline-Elemente im Allgemeinen nicht angewendet, es sei denn, es handelt sich um ersetzte Inline-Elemente.

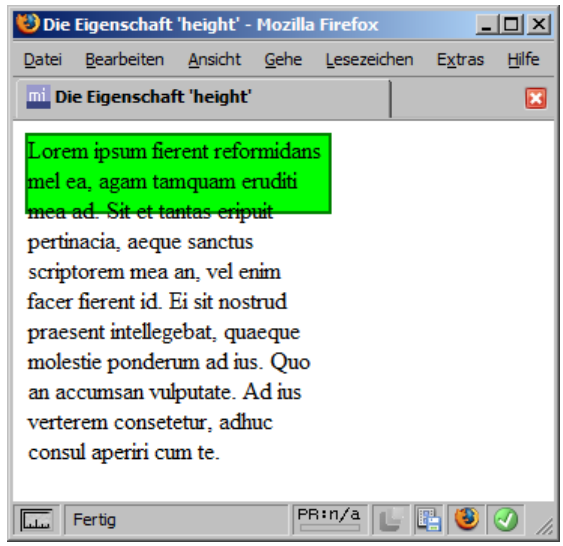


Abbildung 7.26: Herauslaufende Inhalte

7.8.5 Mindesthöhe: Die Eigenschaft min-height

DEFINITION

*Die **Eigenschaft** `min-height` legt die Mindesthöhe der Contentbox eines Blockelements oder eines solchen Elements fest, dessen Dimensionen in CSS bereits bekannt sind.*

Werte	`<Länge>`, `<Prozent>`, `inherit`
Initialwert	0
Vererbung	nein
Anwendung	alle Elemente außer Inline- und Tabellenelemente

Elemente sind standardmäßig so hoch, wie ihr Inhalt es erfordert. Elemente ohne Inhalt sind somit 0 Pixel hoch, was auch der Initialwert für die Eigenschaft `min-height` ist.

Prozentwerte werden relativ zur Höhe des umschließenden Blocks berechnet, wenn diese explizit angegeben ist, ansonsten wird der Wert als 0 interpretiert.

Browserunterstützung

Die Eigenschaft `min-height` wird von allen modernen Browsern unterstützt mit folgenden Einschränkungen:

Der IE kennt diese Eigenschaft bis Version 6 nicht.

Bis Version 7 kennt IE den Wert `inherit` nicht.

Der Apple-Browser unterstützt diese Eigenschaft erst ab Version 2.0.2 fehlerfrei.

7.8.6 Maximalhöhe: Die Eigenschaft max-height

DEFINITION

*Die **Eigenschaft** `max-height` legt die Maximalhöhe der Contentbox eines Blockelements oder eines solchen Elements fest, dessen Dimensionen in CSS bereits bekannt sind.*

Werte	`<Länge>`, `<Prozent>`, `none`, `inherit`
Initialwert	`none`
Vererbung	nein
Anwendung	alle Elemente außer Inline- und Tabellenelemente

Der Initialwert ist browserabhängig. Bei Angabe des Werts `none` wird keine Maximalhöhe festgelegt. Prozentwerte werden relativ zur Höhe des umschließenden Blocks berechnet, wenn diese explizit angegeben ist, ansonsten wird der Wert als `none` interpretiert.

Browserunterstützung

Die Eigenschaft `max-height` wird von allen modernen Browsern unterstützt mit folgenden Einschränkungen:

Der IE kennt diese Eigenschaft bis Version 6 nicht.

Bis Version 7 kennt IE den Wert `inherit` nicht.

Der Apple-Browser unterstützt diese Eigenschaft erst ab Version 2.0.2 fehlerfrei.

7.8.7 Der Boxmodell-Bug

Im Quirks Mode gelten die einfachen Gleichungen »Gesamtbreite gleich Content-Breite plus linke und rechte Padding-, Border- und Margin-Bereiche« und »Gesamthöhe gleich Content-Höhe plus obere und untere Padding-, Border- und Margin-Bereiche« nicht. Während standardkonforme Browser `width` und `height` tatsächlich gemäß den Standards nur auf die Contentbox anwenden und Padding-, Border- und Margin-Bereiche addieren, interpretieren Browser wie der Internet Explorer im Quirks Mode `width` und `height` als Breite beziehungsweise Höhe des Content-, Padding und Border-Bereichs und addieren lediglich den Margin-Bereich. Abbildung 7.27 zeigt den Unterschied.

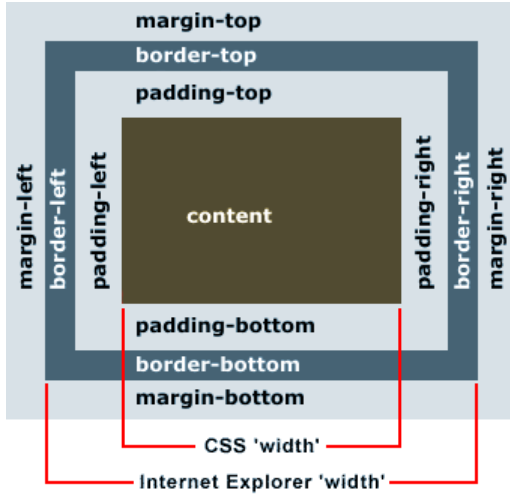


Abbildung 7.27: Der CSS-Boxmodell-Bug in veralteten Browsern

Dies führt in diesen Browsern natürlich schnell zu falsch berechneten Werten. Den klassischen Fall demonstriert folgendes Beispiel. Die Eigenschaft `width` wird auf `100%` gesetzt, hier für ein `div`-Element; in der Praxis geschieht dies jedoch häufig für Elemente, die sich im Gegensatz zu `div`-Elementen standardmäßig nicht über die volle Breite erstrecken, etwa Tabellen. Zusätzlich wird eine Polsterung von 20 Pixeln definiert.

```
div {
  color: black;
  background-color: lime;
  width: 100%;
  padding: 20px;
}
```

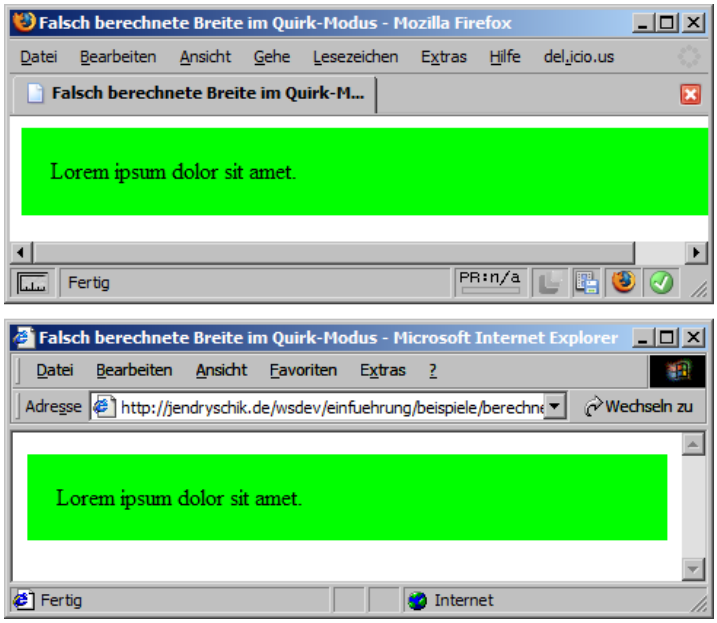


Abbildung 7.28: Korrekte Darstellung im Firefox oben, falsche (wenngleich eher erwartungsgemäß aussehende) Darstellung im Internet Explorer 5 unten

Davon ausgehend, dass `border` und `margin` auf 0 gesetzt sind, spannt das Element somit eigentlich eine Breite von 100 Prozent plus 40 Pixeln auf; in den meisten Fällen kommt es damit entweder zu ungewollten Überlappungen, einem verschobenen Layout oder gar zu horizontalen Scrollbalken, wie in Abbildung 7.28 zu sehen. Im Quirks Mode wird das Dokument scheinbar korrekt angezeigt, das heißt so, wie es ein unerfahrener CSS-Autor vielleicht erwartet.

Um dieses Problem zu umgehen, bleibt Ihnen nur die Möglichkeit, den Abstand auf andere Art zu erzeugen, etwa indem Sie ein Kindelement hinzufügen und `padding` oder `margin` (je nachdem, was im Einzelfall geeigneter ist) für dieses Element setzen.

Der Boxmodell-Bug tritt natürlich auch bei der Verwendung absoluter Längen auf, ist dann aber verhältnismäßig einfach zu beheben. Angenommen, das relevante CSS sähe wie folgt aus:

```
div {
  width: 200px;
  height: 100px;
  padding: 20px;
  margin: 10px;
}
```

Workaround

Die korrekte Gesamtbreite beträgt 260 Pixel, die korrekte Gesamthöhe 160 Pixel. Fehlerhafte Browser stellen das Element jedoch nur 220 Pixel breit und 120 Pixel hoch dar. Um die richtigen Maße zu erreichen, müssen diese Browser also größere Werte für `width` und `height` zugewiesen bekommen, hier `240px` und `140px`. Dabei gehen Sie wie folgt vor: Sie deklarieren zunächst die falschen Werte. Danach machen Sie sich zunutze, dass alle Browser, die das Boxmodell falsch berechnen, mit Maskierung nichts anfangen können, und notieren jeweils in einer eigenen Regel den richtigen Wert, wobei Sie einen Buchstaben der entsprechenden Eigenschaft maskieren. Beachten Sie dabei, dass das Maskierungszeichen in diesem Hack nicht vor den Zeichen »a« bis »f« und »0« bis »9« stehen darf. Fehlerhafte Browser bekommen die korrigierten Werte nicht mit, standardkonforme hingegen schon, womit Sie browserübergreifend die gewünschten Maße erreichen.

```
div {
  width: 240px;
  height: 140px;
  padding: 20px;
  margin: 10px;
}
div {
  w\idth: 200px;
  he\ight: 100px;
}
```

7.8.8 Höhe der Zeilenbox: Die Eigenschaft line-height

Berechnung der Standardhöhe

Die Breite einer Zeilenbox wird durch den umgebenden Block bestimmt. Die Höhe wird wie folgt berechnet:

1. Die Höhe und vertikale Position (siehe Kapitel 7.6.1 »Vertikale Ausrichtung: Die Eigenschaft vertical-align«) jeder Inlinebox in der Zeilenbox wird berechnet.
2. Die Höhe der Zeilenbox ist der Abstand zwischen der oberen Kante der oberen Box und der unteren Kante der untersten Box.

Über die Eigenschaft `line-height` lässt sich die Höhe der Zeilenboxen eines Elements indirekt dadurch beeinflussen, dass die Höhe der Inlineboxen bestimmt wird.

DEFINITION

*Wie die **Eigenschaft** `line-height` auf ein Element angewendet wird, ist abhängig davon, um welche Art von Element es sich handelt. Ist die Eigenschaft für ein Blockelement gesetzt, gibt sie die Mindesthöhe jeder erzeugten Inlinebox an. Ist sie für ein nicht ersetztes Inline-Element gesetzt, gibt sie die exakte Höhe jeder Box an, die von dem Element erzeugt wird.*

Auf diese Weise ist `line-height` für nicht ersetzte Inline-Elemente das Äquivalent zur Eigenschaft `height` für Block- und ersetzte Inline-Elemente.

Die Eigenschaft `line-height` kann als Teil der zusammenfassenden Eigenschaft `font` notiert werden (siehe Kapitel 7.3.6).

Werte	`normal`, `<Länge>`, `<Zahl>`, `<Prozent>`, `inherit`
Initialwert	`normal`
Vererbung	ja
Anwendung	alle Elemente

Die Zeilenhöhe ist browserabhängig. Die meisten Browser verwenden das 1,2-Fache der Schriftgröße oder etwas weniger. Wenn Sie eine Zahl angeben (beispielsweise `line-height: 2;`), ist der berechnete Wert die angegebene Zahl multipliziert mit der Schriftgröße des Elements. Bei einer Prozentangabe wird analog der Prozentwert mit der Schriftgröße des Elements multipliziert.

Die folgenden drei Regeln ergeben dieselbe Zeilenhöhe:

```
div { line-height: 1.5em; } /* Länge   */

div { line-height: 1.5;   } /* Zal     */

div { line-height: 150%;  } /* Prozent */
```

Zeilenhöhe hochsetzen

Der von Browsern voreingestellte Standardwert ist im Allgemeinen zu klein, um eine optimale Lesbarkeit auf dem Bildschirm zu gewährleisten. Ich empfehle Ihnen daher, die Zeilenhöhe bereits für das Element `body` hochzusetzen, am besten auf einen Wert zwischen 150 und 170 Prozent.

Browserunterstützung

Die Eigenschaft `line-height` wird von allen modernen Browsern unterstützt mit folgenden Einschränkungen:

Bei älteren Firefox-Versionen kommt es zu falsch berechneten Werten im Zusammenspiel mit anderen Eigenschaften wie `display`, `padding` und/oder `border`, häufig zu beobachten an mit Listenelementen realisierten Navigationsmenüs. Kommt es zu 1 Pixel großen Lücken zwischen den Listenpunkten, sollten Sie `line-height` auf `normal` setzen und Abstände lieber mit `padding` erzeugen.

Wenn Fließtext ersetzte Inline-Elemente (beispielsweise `img`, `object` oder `input`) enthält, verringert sich die Höhe der entsprechenden Textzeile. Genau genommen fallen der obere Halbdurchschuss der betroffenen Zeile mit dem unteren der vorangegangenen Zeile und der untere Halbdurchschuss mit dem oberen der nachfolgenden Zeile zusammen. Für diesen Fehler gibt es keine überzeugende Lösung. Webautoren können den Star-HTML-Hack heranziehen und mittels `padding-top` und `padding-bottom` für das ersetzte Inline-Element, das den Fehler auslöst, versuchen, die Zeilenhöhe an den Normalwert anzupassen.

7.9 Sichtbarkeit und visuelle Effekte

7.9.1 Sichtbarkeit: Die Eigenschaft visibility

DEFINITION

*Die **Eigenschaft** `visibility` gibt an, ob die durch ein Element erzeugten Boxen angezeigt oder unsichtbar sein sollen.*

Werte	`visible`, `hidden`, `collapse`, `inherit`
Initialwert	`inherit`
Vererbung	nein
Anwendung	alle Elemente

Standardmäßig sind Elemente sichtbar (`visible`). Wenn Sie den Wert `hidden` angeben, wird das Element unsichtbar. Im Gegensatz zur Angabe `display: none;` verschwindet das Element nicht vollständig aus der Darstellung des Dokuments, sondern ist lediglich nicht sichtbar. Stellen Sie sich das so vor: Das Element ist zwar da, aber vollständig transparent, verfügt über eine Breite und eine Höhe und hat dadurch Einfluss auf die Darstellung nachfolgender Elemente.

Ein weiterer Unterschied zwischen `display: none;` und `visibility: hidden;` liegt in der Behandlung der Nachkommenelemente. Nachkommen eines aus der Darstellung des Dokuments herausgenommenen Elements können nicht ausgegeben werden, auch nicht, wenn `display` explizit auf einen anderen Wert als `none` gesetzt wird. Nachkommen von unsichtbaren Elementen können wieder sichtbar gemacht werden, wenn `visibility` auf den Wert `visible` gesetzt wird.

Wird der Wert `collapse` auf eines der Tabellenelemente `tr`, `tbody`, `col` oder `colgroup` angewendet (beziehungsweise auf ein Element, dessen Eigenschaft `display` den Wert `row`, `row-group`, `column` oder `columns-group` hat), fällt das Element in sich zusammen. Damit lassen sich Teile von Tabellen dynamisch aus der Anzeige entfernen, ohne dass sich am Elementfluss etwas ändert. Bei anderen Elementen hat dieser Wert dieselbe Bedeutung wie `hidden`.

Browserunterstützung

Die Eigenschaft `visibility` wird von allen modernen Browsern unterstützt mit folgenden Einschränkungen:

- Der IE erkennt bis Version 7 die Werte `inherit` und `collapse` an normalen Elementen nicht und reagiert innerhalb von Tabellen gar nicht auf diese Eigenschaft.
- In Tabellen erkennt Opera bis Version 9.2 den Wert `collapse` nur für Zeilen und Zeilengruppen, die aber nicht zusammenfallen, sondern lediglich unsichtbar dargestellt werden.
- Safari behandelt bis Version 2 den Wert `collapse` genau so wie `hidden`, das heißt, das Element wird zwar unsichtbar, aber nicht ganz entfernt.
- Konqueror bis Version 2.2 erkennt den Wert `collapse` nicht.
- Ebenso wie Opera erkennt Konqueror 3.5 in Tabellen den Wert `collapse` nur für die Elemente `row` und `row-group`, die aber nicht zusammenfallen, sondern lediglich unsichtbar dargestellt werden.

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

<html xmlns="http://www.w3.org/1999/xhtml"
  lang="de" xml:lang="de">
  <head>
    <meta http-equiv="Content-Type"
      content="text/html; charset=utf-8" />
    <title>Die Eigenschaft 'visibility'</title>

    <style type="text/css">
      /* <![CDATA[ */
      div {
        padding: .5em;
        margin: .5em 0;
        border: 2px solid black;
        color: black;
        background-color: #ffd;
      }
      table {
        border: 2px solid black;
```

```
          color: black;
          background-color: #ffd;
        }
        td {
          padding: .5em;
          border: 1px solid black;
          color: black;
          background-color: #ffb;
        }
        /* ]]> */
      </style>
    </head>

    <body>
      <div>Box 1</div>
      <div style="visibility: hidden;">Box 2</div>
      <div>Box 3</div>

      <table>
        <tr>
          <td>Zelle 1</td>
          <td style="visibility: collapse;">Zelle 2</td>
          <td>Zelle 3</td>
        </tr>
      </table>
    </body>
  </html>
```

Listing 7.8: Die Eigenschaft `visibility`

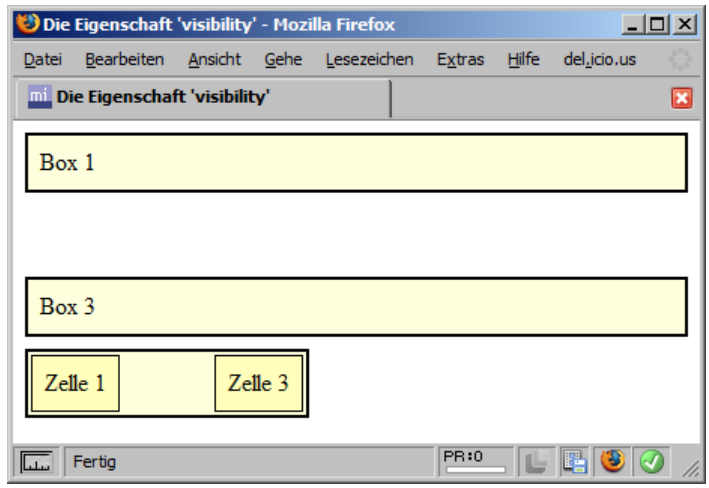


Abbildung 7.29: Unsichtbare Elemente

7.9.2 Überlauf: Die Eigenschaft overflow

Im Allgemeinen ist der Inhalt einer Box höchstens so breit, wie der Inhalt der Box es erfordert. Das bedeutet, Inhalte werden so umbrochen beziehungsweise die Box wird so erweitert (zumeist in der Höhe), dass der Inhalt hineinpasst. In einigen Fällen kann es jedoch dazu kommen, dass der Inhalt einer Box aus dieser herausragt:

- Eine Zeile kann oder soll nicht (beispielsweise weil `white-space: nowrap;` deklariert wurde) umbrochen werden, sodass die Zeilenbox breiter als die Contentbox ist.
- Eine Blockbox ist breiter als ihr umschließender Block. Im folgenden Beispiel würde Box 2 aus Box 1 herausstehen:

```
<div class="box1" style="width: 300px;">
  Box 1
  <div class="box2" style="width: 500px">Box 2</div>
</div>
```

- Die notwendige Höhe des Elementinhalts überschreitet die dem Block explizit (mittels der Eigenschaft `height`) zugeordnete Höhe, beziehungsweise die Höhe einer Box überschreitet die dem umschließenden Block explizit zugeordnete Höhe. Abbildung 7.26 zeigt ein entsprechendes Beispiel.
- Eine Box ist derart positioniert, dass Teile außerhalb des umschließenden Blocks liegen. Oder anders ausgedrückt: Eine Box wurde aus ihrem umgebenden Block herauspositioniert.
- Eine Box hat negative Randabstände, also negative Werte für die Eigenschaft `margin`, sodass Teile außerhalb ihres umgebenden Blocks liegen.

DEFINITION

*Die **Eigenschaft** `overflow` gibt an, wie überlaufender Inhalt von Nachkommenelementen dargestellt wird, also Inhalt, der aus der aktuellen Box hinauslaufen würde. `overflow` kann nur auf Blockelemente und ersetzte Elemente angewendet werden.*

Werte	`visible`, `hidden`, `scroll`, `auto`, `inherit`
Initialwert	`visible`
Vererbung	nein
Anwendung	Blockelemente

Der Standardwert ist `visible`; überlaufender Inhalt wird vollständig dargestellt.

Beim Wert `hidden` wird der überlaufende Inhalt nicht dargestellt; das Nachkommenelement wird einfach abgeschnitten, sobald es die Content-Grenze erreicht.

Der Wert `scroll` sorgt dafür, dass der überlaufende Inhalt abgeschnitten wird, jedoch über Rollbalken erreichbar ist, die zwischen Padding- und Border-Bereich eingefügt und bei der Berechnung der Dimensionen der Box mit berücksichtigt werden. Rollbalken werden auch dann dargestellt, wenn sie nicht notwendig sind, es also kein Nachkommenelement gibt, das über die Content-Grenze hinausragen würde.

Der Wert `auto` schließlich überlässt es dem Browser zu entscheiden, wie überlaufender Inhalt dargestellt wird. In den meisten Fällen werden Rollbalken angeboten, sobald sie benötigt werden.

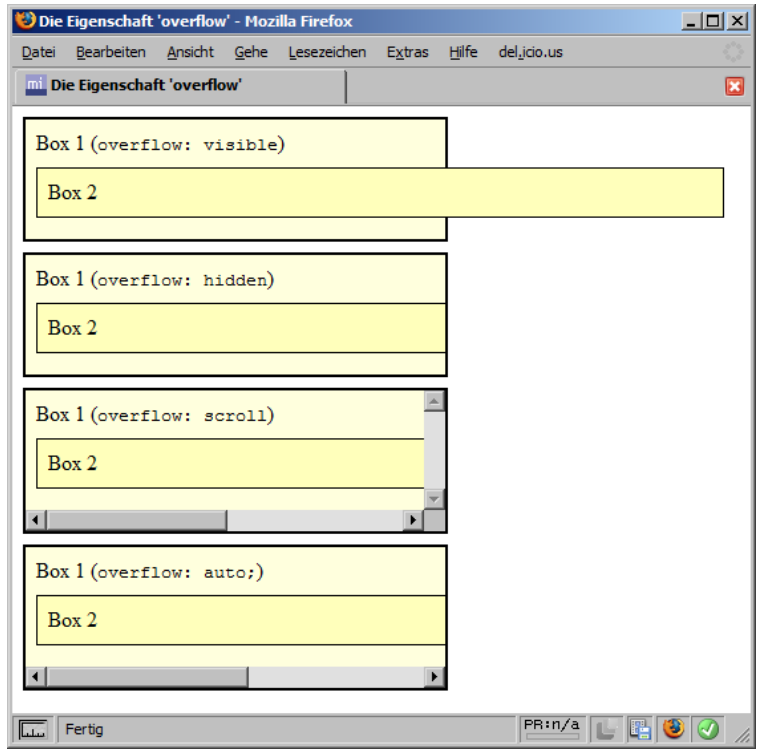


Abbildung 7.30: Die Eigenschaft `overflow`

Floats einschließen

In Abschnitt 7.6.2 habe ich erklärt, dass floatierte Boxen aus dem normalen Elementfluss herausgenommen werden. Wenn der umgebende Block eine geringere Höhe hat als die darin floatierte Box, ragt diese unten aus dem Block heraus. Um dies zu vermeiden, können Sie die Eigenschaft `overflow` einsetzen. Dieser Workaround funktioniert in allen Browsern, auch im Internet Explorer, wenn der umgebende Block »Layout« hat (siehe Kapitel 6.8.3). Die Kombination `overflow: hidden;` zusammen mit `height: 1%;` hat sich in diesem Zusammenhang bewährt.

Browserunterstützung

Die Eigenschaft `overflow` wird von allen modernen Browsern unterstützt mit folgenden Einschränkungen:

Überfließende Inhalte werden vollständig und richtig dargestellt, jedoch wird, abweichend von der Spezifikation, das umschließende Element dazu ausgedehnt. Das gilt auch für den Wert `visible` als Ausgangswert. Der Wert `inherit` wird nicht erkannt.

Der Wert `inherit` wird nicht erkannt.

Erkennt den Wert `scroll` nicht.

7.9.3 Deckkraft: Die Eigenschaft opacity

DEFINITION

*Die **Eigenschaft** `opacity` gibt die Deckkraft eines Elements an.*

Werte	`<Zahl>`, `inherit`
Initialwert	`1`
Vererbung	nein
Anwendung	alle Elemente

Die Deckkraft wird mit einem Wert von `0.0` bis `1.0` angegeben. Je höher der Wert, desto größer ist die Deckkraft, mit anderen Worten: Je geringer der Wert, desto höher ist die Transparenz des Elements, sodass überdeckte Elemente durchscheinen. Der Initialwert ist `1.0`.

Ausblick auf CSS 3

Die Eigenschaft `opacity` ist nicht Bestandteil von CSS bis Version 2.1, sondern Teil der noch nicht verabschiedeten W3C-Empfehlung *CSS3 Color Module*[4], die

4 *http://www.w3.org/TR/css3-color/*

mit hoher Wahrscheinlichkeit Teil des CSS 3-Standards werden wird. `opacity` gehört zu den neuen CSS-Eigenschaften, die bereits von einigen Browsern unterstützt werden und daher schon verwendet werden können.

IE-Alpha-Filter

Auch für den Internet Explorer können Sie Transparenz mittels des proprietären Alpha-Filters erreichen. Dieser Filter ist allerdings in keiner CSS-Empfehlung enthalten; dessen Verwendung verstößt somit eindeutig gegen die Bedingungen der BITV und funktioniert auch nur im Internet Explorer. An dieser Stelle wird daher nur gezeigt, wie Sie den gleichen Effekt erzielen können wie mit `opacity`. Weitere Informationen zu Filtern erhalten Sie bei Microsoft[5] oder SELFHTML[6].

Mit der Eigenschaft `filter` und dem Wert `Alpha()` können Sie die Transparenz von Elementen beeinflussen, indem Sie für den Parameter `opacity` einen Wert von `0` bis `100` angeben. Auch hierbei gilt: je geringer der Wert, desto höher die Transparenz.

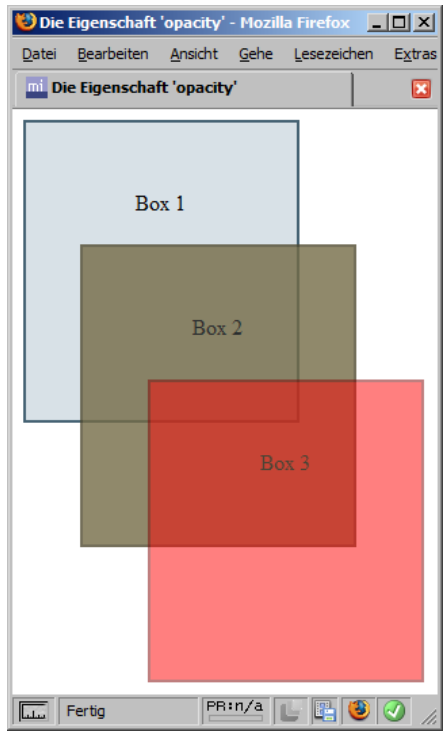


Abbildung 7.31: Die Eigenschaft `opacity`

5 *http://msdn.microsoft.com/workshop/author/filter/filters.asp*
6 *http://de.selfhtml.org/css/eigenschaften/filter.htm*

Browserunterstützung

Die Eigenschaft `opacity` wird derzeit von aktuellen Gecko-Browsern und Opera 9 unterstützt. Ältere Gecko-Browser haben `opacity` in der Schreibweise `-moz-opacity` implementiert.

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

<html xmlns="http://www.w3.org/1999/xhtml"
  lang="de" xml:lang="de">
  <head>
    <meta http-equiv="Content-Type"
      content="text/html; charset=utf-8" />
    <title>Die Eigenschaft 'opacity'</title>

    <style type="text/css">
      /* <![CDATA[ */
      div {
        color: black;
        padding: 50px 50px 150px 50px;
        width: 100px;
        text-align: center;
        position: absolute;
      }
      #box1 {
        border: 2px solid #466374;
        background-color: #D8E1E7;
      }
      #box2 {
        border: 2px solid #514B31;
        background-color: #746B46;
        top: 100px;
        left: 50px;
        opacity: .8;
        filter: Alpha(opacity = 80);
      }
      #box3 {
        border: 2px solid maroon;
        background-color: red;
        top: 200px;
        left: 100px;
        opacity: .5;
        filter: Alpha(opacity = 50);
      }
      /* ]]> */
    </style>
  </head>
```

```
    <body>
      <div id="box1">Box 1</div>
      <div id="box2">Box 2</div>
      <div id="box3">Box 3</div>
    </body>
  </html>
```

Listing 7.9: Die Eigenschaft `opacity`

7.9.4 Mauszeiger: Die Eigenschaft cursor

DEFINITION

*Die **Eigenschaft** `cursor` gibt den Typ des Mauszeigers an, der dargestellt wird, wenn ein Element mit der Maus berührt wird.*

Werte	`auto`, `crosshair`, `default`, `pointer`, `move`, `e-resize`, `ne-resize`, `nw-resize`, `n-resize`, `se-resize`, `sw-resize`, `s-resize`, `w-resize`, `text`, `wait`, `progress`, `help`, `<URI>`, `inherit`
Initialwert	`auto`
Vererbung	ja
Anwendung	alle Elemente

- `auto`: Welcher Mauszeiger angezeigt wird, ist abhängig vom Kontext.
- `crosshair`: Ein einfaches Fadenkreuz.
- `default`: Der Standardmauszeiger, häufig ein nach links geneigter Pfeil.
- `pointer`: Ein Zeiger, häufig eine Hand mit ausgestrecktem Zeigefinger.
- `move`: Kennzeichnet eine Bewegung.
- `e-resize`, `ne-resize`, `nw-resize`, `n-resize`, `se-resize`, `sw-resize`, `s-resize`, `w-resize`: Zeiger, die beim Verschieben einer Fensterkante beziehungsweise beim Vergrößern oder Verkleinern eines Fensters angezeigt werden. `n`, `e`, `s` und `w` stehen für die vier Richtungen Nord (*north*), Ost (*east*), Süd (*south*) und West (*west*).
- `text`: Gibt an, dass Text markiert werden kann.
- `wait`: Zeigt einen Wartevorgang an, häufig in Form einer Sanduhr.
- `progress`: Zeigt einen Wartevorgang an, bei dem der Benutzer noch mit dem Programm interagieren kann.
- `help`: Zeigt ein Hilfesymbol an.

- <URI>: Es kann eine Liste eigener Mauszeiger per URI eingebunden werden, wobei der Browser wie bei Schriftdeklarationen von links nach rechts vorgeht und den Zeiger dargestellt, den er verarbeiten kann. Am Ende der Liste sollte eines der oben genannten Schlüsselwörter stehen. Beispiel: `E { cursor: url(foo.cur), url(bar.csr), auto; }`

Browserunterstützung

Die Eigenschaft `cursor` wird von allen modernen Browsern unterstützt mit folgenden Einschränkungen:

Alle Browser – Nur aktuelle Browser unterstützen den Wert `url()`.

Erst ab Version 7 interpretiert der IE die Eigenschaft, versteht jedoch den Wert `inherit` nicht.

Opera unterstützt ebenfalls nicht den Wert `inherit` bis Version 8.

7.10 Gestaltung von Listen

7.10.1 Stil der Listenelementmarkierung: Die Eigenschaft list-style-type

DEFINITION

*Die **Eigenschaft** `list-style-type` gibt das Erscheinungsbild der Listenelementmarkierung geordneter und ungeordneter Listen an beziehungsweise die Art der Nummerierung geordneter Listen.*

Werte	`none`, `inherit`, `disc`, `circle`, `square`, `decimal`, `decimal-leading-zero`, `lower-roman`, `upper-roman`, `hebrew`, `georgian`, `armenian`, `cjk-ideographic`, `hiragana`, `katakana`, `hiragana-iroha`, `katakana-iroha`, `lower-latin`, `lower-alpha`, `upper-latin`, `upper-alpha`, `lower-greek`
Initialwert	`disc` bei ungeordneten Listen (`ul`), `decimal` bei geordneten (`ol`)
Vererbung	ja
Anwendung	Listenpunkte

Beim Wert `none` wird keine Listenelementmarkierung angezeigt.

Ungeordnete Listen

Die Werte `disc`, `circle` und `square` bestimmen die Art der Listenelementmarkierung bei ungeordneten Listen. Die Darstellung ist browserabhängig, allerdings wird `disc` im Allgemeinen als gefüllter Kreis, `circle` als leerer Kreis und `square` als gefülltes Rechteck interpretiert. Die Farbe der Listenelementmarkierung entspricht der Schriftfarbe.

Geordnete Listen

Die folgenden Werte sollten nur bei geordneten Listen verwendet werden. Viele dieser Werte werden nicht von allen Browsern interpretiert. Die Ausrichtung der Listenelementmarkierungen ist browserabhängig.

- `decimal`: Dezimale Nummerierung.
- `decimal-leading-zero`: Dezimale Nummerierung mit führender Null, das heißt, den Ziffern 1 bis 9 wird eine Null vorangestellt (01, 02, ..., 09, 10, 11, ...).
- `lower-roman`: Nummerierung in römischen Kleinbuchstaben.
- `upper-roman`: Nummerierung in römischen Großbuchstaben.
- `hebrew`: Nummerierung mit traditionellen hebräischen Buchstaben.
- `georgian`: Nummerierung mit traditionellen georgischen Buchstaben.
- `armenian`: Nummerierung mit traditionellen armenischen Buchstaben.
- `cjk-ideographic`: Nummerierung mit ideografischen Zeichen.
- `hiragana`, `katakana`, `hiragana-iroha` und `katakana-iroha`: Nummerierungen mit japanischen Glyphen.
- `lower-latin` oder `lower-alpha`: Nummerierung in kleinen ASCII-Buchstaben.
- `upper-latin` oder `upper-alpha`: Nummerierung in großen ASCII-Buchstaben.
- `lower-greek`: Nummerierung in griechischen Kleinbuchstaben.

Nummerierung nach Ende des Alphabets

Es ist in CSS nicht geregelt, wie bei alphabetischer Nummerierung nach Ende des Buchstabenvorrats des Alphabets, zum Beispiel bei `lower-latin` nach 26 Listenpunkten, weiternummeriert werden soll. Viele Browser gehen im Allgemeinen so vor, dass sie analog zu Zahlensystemen eine weitere »Ziffer« anfügen und hochzählen, beispielsweise bei `lower-latin`: a, b, ..., y, z, aa, ab, ..., ay, az, ba, bb, ...

Browserunterstützung

Die **Eigenschaft list-style-type** wird von den meisten modernen Browsern unterstützt mit folgenden Einschränkungen:

Der IE kann nur Ziffern und Zeichen des lateinischen Alphabets verwenden, auch `decimal-leading-zero` wird nicht erkannt. Für alle nicht erkannten Typen wird das dezimale Äquivalent eingesetzt. `inherit` wird nicht interpretiert.

Erkennt ab Version 8.5 alle Listentypen mit Ausnahme der fernöstlichen und dem Wert `hebrew`.

Safari 1.3 kennt den Wert `decimal-leading-zero` nicht.

Der KDE-Browser erkennt alle Listentypen mit Ausnahme der fernöstlichen.

7.10.2 Grafiken als Listenelementmarkierung: Die Eigenschaft list-style-image

DEFINITION

*Die **Eigenschaft** `list-style-image` ermöglicht es, eine Grafik festzulegen, die anstelle der gewöhnlichen Listenelementmarkierung angezeigt wird.*

Werte	`none`, `<URI>`, `inherit`
Initialwert	`none`
Vererbung	ja
Anwendung	Listenpunkte

Die Schreibweise des Werts `<URI>` entspricht der bei Einbindung eines Hintergrundbilds über die Eigenschaft `background-image`.

HINWEIS

Im Allgemeinen wird das Bild nicht exakt mittig vor das Listenelement gesetzt, sondern nach oben ausgerichtet. Dadurch wirkt die Listenelementmarkierung verschoben, was Sie ausgleichen könnten, indem Sie die Grafik so erstellen, dass sie über einen gewissen Abstand nach oben verfügt. Vielen Webautoren ist dies zu umständlich, sodass sie auf die Eigenschaft `list-style-image` verzichten und eine Listenelementmarkierung stattdessen über ein Hintergrundbild nachstellen, dessen (vertikale) Position sich im Gegensatz zu der Position von Listenelementmarkierungen kontrollieren lässt.

Browserunterstützung

Alle Browser – Die Eigenschaft `list-style-image` wird von allen modernen Browsern ohne Einschränkung unterstützt.

7.10.3 Position der Listenelementmarkierung: Die Eigenschaft list-style-position

DEFINITION

*Die **Eigenschaft** `list-style-position` gibt die Position der Listenelementmarkierung an.*

Werte	`outside`, `inside`, `inherit`
Initialwert	`outside`
Vererbung	ja
Anwendung	Listenpunkte

Die Listenelementmarkierung kann entweder außerhalb (`outside`) oder innerhalb (`inside`) der Listenelementbox liegen.

Browserunterstützung

Alle Browser – Die Eigenschaft `list-style-position` wird von allen modernen Browsern ohne Einschränkung unterstützt.

7.10.4 Listenelementmarkierungen: Die zusammenfassende Eigenschaft list-style

DEFINITION

*Die **Eigenschaft** `list-style` fasst die Einstellungen der einzelnen Listeneigenschaften (`list-style-type`, `list-style-image`, `list-style-position`) zusammen. Sie setzt zunächst die einzelnen Eigenschaften auf ihre Initialwerte und weist ihnen dann die explizit in der Deklaration angegebenen Werte zu.*

Die Regel

```
ol.inhaltsverzeichnis {
  list-style-type: decimal-leading-zero;
  list-style-position: inside;
}
```

können Sie auch wie folgt schreiben:

```
ol.inhaltsverzeichnis {
  list-style: decimal-leading-zero inside;
}
```

Browserunterstützung

Alle Browser – Die Eigenschaft `list-style` wird von allen modernen Browsern unterstützt. Auf Einschränkungen wurde bei den Einzeleigenschaften hingewiesen.

7.11 Das CSS-Tabellenmodell

Tabellen und Tabellenelemente verhalten sich anders als andere Elemente. Das Standard-Boxmodell gilt nicht für Tabellen. Die rechteckigen Boxen mit Inhalt, Polsterung, Rahmen und Randabstand werden stattdessen durch das CSS-Tabellenmodell beschrieben, das in diesem Abschnitt Schritt für Schritt erläutert wird.

Das CSS-Tabellenmodell beeinflusst die Darstellung folgender Eigenschaften an Tabellenelementen:

- `background` und `border` (sowie natürlich alle »untergeordneten« Eigenschaften wie `background-color`, `border-width`, `border-left-color` und so weiter) werden nach speziellen Regeln berechnet.
- `margin` ist nur auf Tabellen selbst anwendbar, das heißt auf Elemente, deren `display`-Eigenschaft den Wert `table` oder `inline-table` hat.
- `padding` beeinflusst nur Tabellen und Tabellenzellen, also Elemente, deren `display`-Eigenschaft den Wert `table`, `inline-table` oder `table-cell` hat.

Darüber hinaus gibt es Unterschiede in der Berechnung der Breite und Höhe.

7.11.1 Virtuelle Tabellenschichten und Transparenz

Rufen Sie sich noch einmal den Aufbau von Tabellen in Erinnerung. Wenn man wollte, könnte man sagen, dass zwischen dem Container für die Auszeichnung von Tabellen `table` und den eigentlichen Tabellenzellen `td` und `th` viele weitere Boxen liegen, konkret ausgezeichnete Boxen und berechnete, »virtuelle« Boxen:

Ausgezeichnete Boxen

Konkret ausgezeichnet sind folgende Boxen:

- Es gibt Zeilen, ausgezeichnet durch `tr`, innerhalb deren sich die Zellen befinden.

- Es gibt Zeilengruppen, ausgezeichnet durch `thead` (Tabellenkopf), `tfoot` (Tabellenfuß) und `tbody` (Tabellenrumpf).

»Virtuelle« Boxen

Aber es gibt noch weitere Boxen, die die Zellen zwar nicht im XHTML-Quelltext, aber letztendlich im Tabellenmodell umfassen:

- Die Tabelle enthält so viele Spalten wie
  - durch `col`- oder `colgroup`-Elemente in Verbindung mit entsprechenden `span`-Attributen definiert wurden oder
  - es maximal Zellen in einer Zeile gibt.
- Spalten können zu Spaltengruppen, ausgezeichnet durch `colgroup`, zusammengefasst werden.

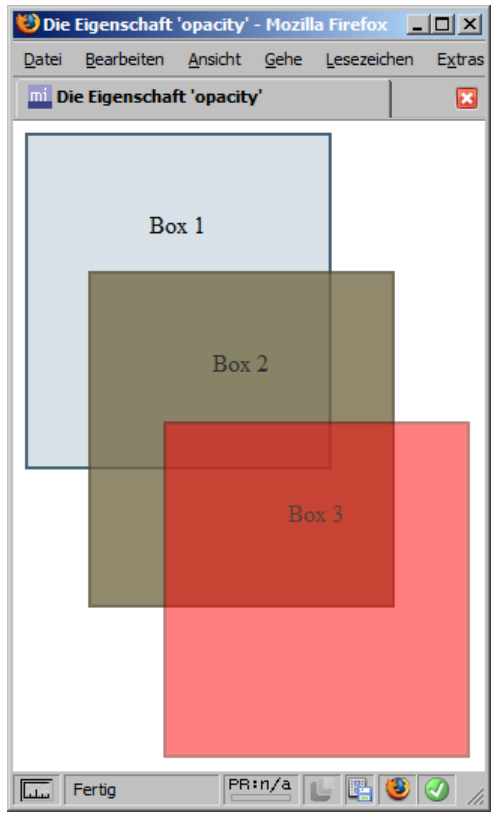


Abbildung 7.32: Die sechs Tabellenschichten, von unten nach oben: Tabellencontainer, Spaltengruppen, Spalten, Zeilengruppen, Zeilen, Zellen

Schichten

Aus diesen Überlegungen ergeben sich sechs Schichten, die im CSS-Tabellenmodell übereinanderliegen, je eine Schicht für den Tabellencontainer, die Spaltengruppen, die Spalten, die Zeilengruppen, die Zeilen und die Zellen. Jede dieser Schichten können Sie sich als transparente Folie vorstellen, die wie auf einem Lichtprojektor die darunterliegende Schicht überdeckt, wobei diese jedoch durchscheint. Abbildung 7.32 illustriert diese Schichten.

Hintergrundfarben

Das Schichtmodell ist wichtig, um herauszufinden, welche Hintergrundfarbe ein bestimmter Teil einer Tabelle hat. Für jedes Tabellenelement können Sie eine eigene Hintergrundfarbe oder ein eigenes Hintergrundbild deklarieren. Der Hintergrund, der für ein Element in einer der Schichten gesetzt wird, ist jedoch nur dann sichtbar, wenn die darüberliegenden Schichten einen transparenten Hintergrund haben. Oder anders ausgedrückt: Wenn Sie für Elemente einer Schicht einen Hintergrund deklarieren, überdeckt dieser den Hintergrund der darunterliegenden Schichten.

7.11.2 Algorithmen für die Tabellenbreite: Die Eigenschaft table-layout

DEFINITION

*Die **Eigenschaft** `table-layout` gibt an, welcher Algorithmus für die Berechnung der Breite einer Tabelle verwendet wird.*

Werte	`auto`, `fixed`, `inherit`
Initialwert	`auto`
Vererbung	ja
Anwendung	Tabellen

Das **automatische Tabellenlayout** (`auto`) ist das für den Browser deutlich aufwendigere der beiden. Die Tabellenbreite wird durch die Breite ihrer Spalten und der dazwischenliegenden Rahmen festgelegt, wobei die Spaltenbreite sich an den Inhalten orientiert. Die Tabelle muss daher vollständig heruntergeladen sein, bevor sie berechnet und dargestellt werden kann, vor allem dann, wenn die Spaltenbreite nicht durch die Elemente `colgroup` und `col` festgelegt ist, sondern sich aus der Breite der einzelnen Tabellenzellen ergibt. Für Webautoren ist dieser Algorithmus häufig der komfortablere, da im Allgemeinen alle Zelleninhalte vollständig dargestellt werden; aus diesem Grund ist `auto` der Initialwert.

Beim **festen Tabellenlayout** (`fixed`) ist die Tabellenbreite nur von den Vorgaben für die Tabellenspalten mittels `col` und `colgroup` und den Breiten der Zel-

len der ersten Zeile abhängig. Breitenangaben haben folglich Vorrang vor den Inhalten. So kann der Browser bereits mit dem Aufbau der Tabelle auf dem Bildschirm beginnen, wenn er die Tabelle noch nicht vollständig heruntergeladen hat. Sollte der Inhalt einer der folgenden Zellen breiter sein, als die Tabellenbreiten beziehungsweise die Zellen der ersten Zeile festlegen, hängt es vom Wert der Eigenschaft `overflow` ab, wie der überfließende Teil behandelt wird.

Berechnung der Tabellenbreite

Die genaue Berechnung der Tabellenbreite geht in vier Schritten vor sich:

1. Falls die Breite von einer oder mehreren Spalten explizit deklariert ist, wird diese direkt verwendet, andernfalls bestimmt jede Zelle der ersten Zeile die Breite der zugehörigen Spalte.
2. Wird für eine oder mehrere Spalten keine Breite definiert, wird der noch verfügbare Platz auf diese aufgeteilt.
3. Die Summe aller Werte für Spaltenbreiten, Zwischenräume und Rahmen ergibt die Gesamtbreite der Tabelle.
4. Wurde der Tabelle selbst eine Breite zugewiesen, die über die berechnete Breite hinausgeht, wird die Differenz beider Werte gleichmäßig auf die Spalten verteilt.

Festes Tabellenlayout eignet sich nicht für jede Form von Tabellen, sondern nur für Tabellen, deren Inhalte beziehungsweise deren Inhaltsbreiten bereits im Vorfeld überschaubar und innerhalb einer Spalte ähnlich sind.

Browserunterstützung

Alle Browser – Leider kann bis heute kein Browser das feste Tabellenlayout richtig interpretieren. Die Spaltenbreiten spielen im Allgemeinen keine Rolle; sie werden unabhängig vom verwendeten Algorithmus ignoriert, sobald Zelleninhalte mehr Platz erfordern, und die Zellen werden auf die erforderliche Breite ausgedehnt. Der verwendete Algorithmus spielt nur dann eine Rolle, wenn der Tabelle selbst eine explizite Breite zugewiesen wird, die auch relativ sein kann. In diesem Fall verbreitern die Zellen sich nicht, auch wenn deren Inhalte mehr Platz benötigen.

7.11.3 Algorithmus für die Tabellenhöhe

Die Höhe einer Tabelle wird in drei Schritten berechnet:

1. Zunächst wird die Höhe jeder einzelnen Zelle bestimmt. Die Mindesthöhe ist abhängig vom Zelleninhalt und der vertikalen Ausrichtung. Falls eine Höhe mittels der Eigenschaft `height` für `td` oder `th` vorgegeben wurde, wird geprüft, welcher der beiden Höhenwerte der größere ist. Dieser Wert wird dann verwendet.

2. Anschließend wird die Höhe jeder Zeile bestimmt. Die Mindesthöhe wird durch die höchste Zelle der Zeile bestimmt. Auch hier gilt: Falls eine Höhe mittels der Eigenschaft `height` für `tr` vorgegeben wurde, wird geprüft, welcher der beiden Höhenwerte der größere ist, und dieser dann verwendet. Zudem werden auch alle Zellen innerhalb der Zeile auf diesen Wert gesetzt, damit alle Zellen gleich hoch sind.
3. Die Gesamthöhe der Tabelle ergibt sich aus den einzelnen Zeilenhöhen zuzüglich Rahmen und Abständen. Wird die Tabellenhöhe mittels der Eigenschaft `height` für `table` explizit festgelegt, wird dieser Wert auch hier nur dann verwendet, wenn er größer als der berechnete Wert ist. Der zusätzliche Platz wird dann gleichmäßig auf die Zeilen (und somit auf die Zellen) aufgeteilt.

7.11.4 Rahmenmodell: Die Eigenschaft border-collapse

Für jedes Tabellenelement können Sie einen Rahmen definieren, wodurch es schnell zu Konflikten kommen kann. Wenn Sie verschiedenfarbige und unterschiedlich dicke Rahmen für `table`, für `tbody` und für `td` definieren, wie soll das Gitter, das daraus entsteht, aussehen? CSS bietet zwei Darstellungsmöglichkeiten an: Beim Modell der getrennten Rahmen werden Rahmen nebeneinander gezeichnet, beim Modell der zusammenfallenden Rahmen überlagern sich Rahmen benachbarter Elemente nach bestimmten Regeln.

Welches Rahmenmodell Sie verwenden, ist letztendlich Geschmackssache.

DEFINITION

*Die **Eigenschaft** `border-collapse` legt fest, welches Rahmenmodell für die Darstellung einer Tabelle verwendet werden soll.*

Werte	`separate`, `collapse`, `inherit`
Initialwert	`separate`
Vererbung	ja
Anwendung	Tabellen

Beim Wert `separate` wird wird das **Modell der getrennten Rahmen** verwendet. Es erlaubt Rahmen nur für die Elemente `table`, `td` und `th`, die mit einem kleinen Abstand voneinander dargestellt werden. Die Größe des Abstands können Sie über die Eigenschaft `border-spacing` beeinflussen. Leere Zellen werden in Abhängigkeit von der Eigenschaft `empty-cells` dargestellt.

Beim Wert `collapse` wird das **Modell der zusammenfallenden Rahmen** verwendet. Die Deklaration eines Rahmens ist für alle Tabellenelemente erlaubt, wobei mithilfe folgender Regeln entschieden wird, wie der Rahmen dargestellt werden soll, wenn Deklarationen für unterschiedliche Tabellenelemente miteinander in Konflikt geraten (etwa Deklarationen für `tr` und `td`):

Regelfolge

1. Die Deklaration `border-style: hidden;` hat Priorität vor allen anderen Angaben.
2. Breite Rahmenstärken (`border-width`) werden gegenüber schmalen bevorzugt.
3. Bei gleich breiten Rahmenstärken entscheidet die Rahmenart (`border-style`) in folgender Reihenfolge: Die höchste Priorität hat `double`, anschließend folgen `solid`, `dashed`, `dotted`, `ridge`, `outset`, `groove` und zuletzt `inset`.
4. Unterscheiden Rahmen sich nur in der Farbe, wird nach dem Element gewichtet, zu dem der Rahmen gehört. Dabei wird das Modell der Tabellenschichten herangezogen: Höchste Priorität hat die Angabe für eine Zelle (`td`, `th`), anschließend für eine Zeile (`tr`), für eine Zeilengruppe (`thead`, `tfoot`, `tbody`), eine Spalte (`col`), eine Spaltengruppe (`colgroup`) und schließlich für den Tabellencontainer (`table`) selbst.
5. Die Deklaration `border-style: none;` hat die niedrigste Priorität.

Browserunterstützung

Die Eigenschaft `border-collapse` wird von allen modernen Browsern unterstützt mit folgenden Einschränkungen:

IE 6 erkennt das CSS-Tabellenmodell nur teilweise. Die Darstellung der Rahmen hängt davon ab, ob die Zellen gefüllt sind oder nicht.

IE 7 unterscheidet zwischen den Werten `collapse` und `separate`, hält sich aber bei zusammenfallenden Rahmen nicht an die geforderten Prioritäten.

Opera kann Schwierigkeiten mit der korrekten Darstellung dieser Eigenschaft haben, die auch vom Kontext im Webdokument und von anderen Eigenschaften desselben Elements herrühren können.

Beispiel für den Aufbau eines Rahmenkonzepts

Um den Unterschied zwischen beiden Rahmenmodellen zu illustrieren, folgt ein Beispiel in mehreren Schritten. Ich versehe eine Tabelle mit Öffnungszeiten Schritt für Schritt mit Rahmen, wobei ich jeweils zeige, wie sich die einzelnen Deklarationen auf das Modell der getrennten und das Modell der zusammenfallenden Rahmen auswirken.

Erster Schritt

Im ersten Schritt deklariere ich für dieselbe Tabelle ein Mal `border-collapse: separate;` und ein Mal `border-collapse: collapse;`. Zusätzlich ziehe ich einen 3 Pixel breiten dunkelblauen Rahmen um die Tabelle. Sie sehen sofort, dass die obere Tabelle breiter ist als die untere, da Browser zwischen den Zellen (besser gesagt: zwischen den Border-Bereichen um die Zellen) einen kleinen Abstand darstellen.

```
/* Rahmen um den Tabellencontainer */
table { border: 3px solid navy; }
```

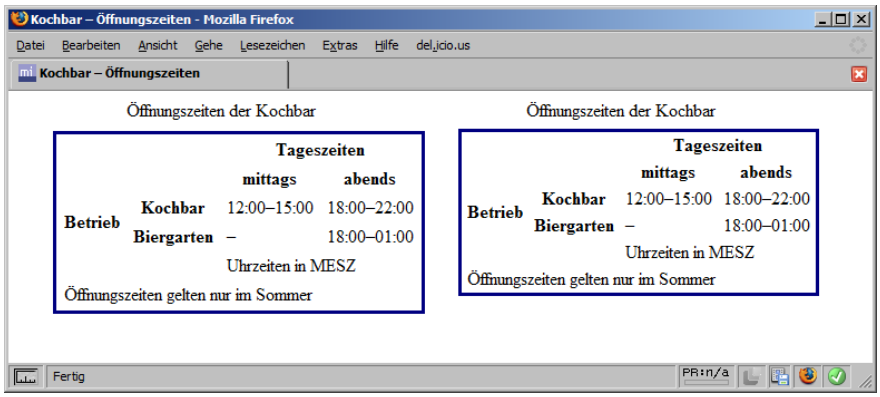


Abbildung 7.33: Erster Schritt: Rahmen um die Tabelle

Zweiter Schritt

Im zweiten Schritt deklariere ich Rahmen für die Spaltengruppen.

Zum einen ziehe ich einen 3 Pixel breiten grünen Rahmen um den Tabellenkopf und den Tabellenfuß. Dieser überschreibt den Rahmen um die Tabelle aufgrund Regel 4, da gleichartige Deklarationen für Zeilengruppen höher priorisiert sind als für Tabellencontainer.

Zugleich setze ich den linken und den rechten Rahmen für den Tabellenrumpf auf `hidden`. Diese Angabe hat Priorität vor allen anderen Angaben (Regel 1) und überschreibt somit die Rahmendeklaration für `table`.

Auf die Darstellung der ersten Tabelle haben diese Regeln erwartungsgemäß keine Auswirkung.

```
/* Rahmen um Tabellenkopf und -fuß */
thead, tfoot { border: 3px solid green; }

/* Rahmen um Tabellenrumpf */
tbody {
  border-left: hidden;
  border-right: hidden;
}
```

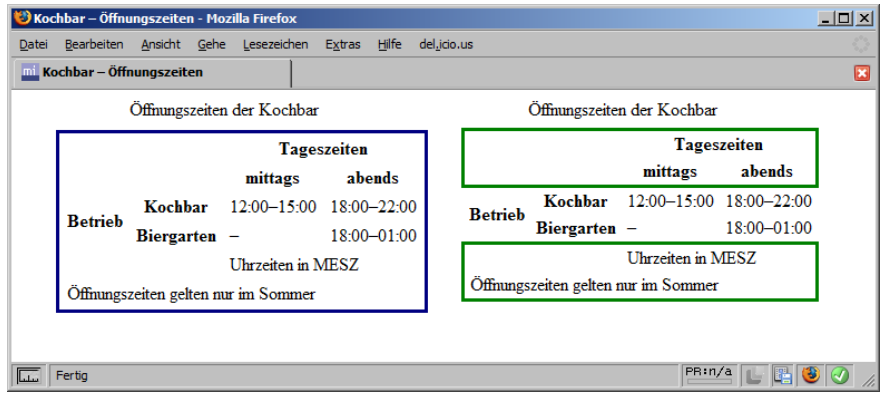


Abbildung 7.34: Zweiter Schritt: Rahmen um Spaltengruppen

Dritter Schritt

Als Nächstes erhalten die Tabellenzeilen einen Rahmen. Die Schicht, in der die Zeilen liegen, hat zwar eine höhere Priorität als alle bisher angesprochenen Schichten, jedoch ist die Rahmendicke geringer (Regel 2), sodass der blaue Rahmen nur dort zu sehen ist, wo es zu keinem Konflikt mit den bisher vorgenommenen Rahmendeklarationen kommt.

Auch diese Regel hat auf die Darstellung der ersten Tabelle keinen Einfluss.

```
/* Rahmen um Tabellenzeilen */
tr { border: 2px solid blue; }
```

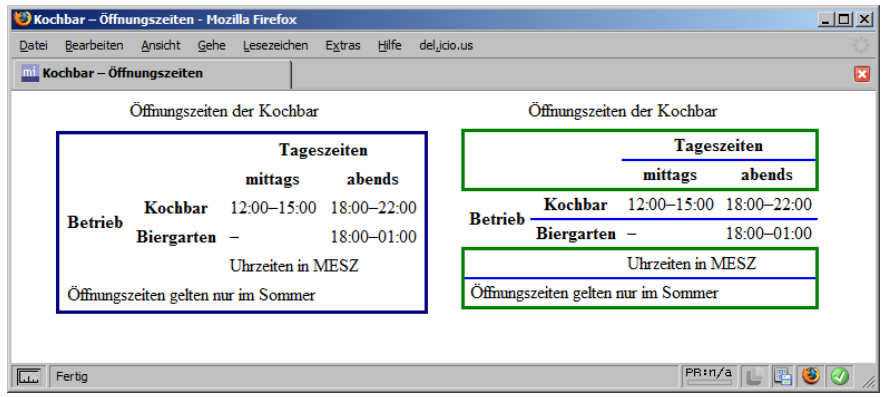


Abbildung 7.35: Dritter Schritt: Rahmen um Zeilen

Vierter Schritt

Der nächste Schritt ist nun besonders interessant. Datenzellen erhalten in diesem Schritt einen 1 Pixel dicken schwarzen Rahmen, der bei Kopfzellen durchgezogen und bei Datenzellen punktiert dargestellt werden soll. Aufgrund der geringen Strichdicke (Regel 2) werden diese Rahmen nur dort gezeichnet, wo

es nicht zu Konflikten mit bisherigen Rahmen kommt. Interessant ist nun die Frage, wie der Rahmen zwischen Kopf- und Datenzellen aussieht: durchgezogen oder punktiert? Antwort gibt Regel 3: Es gewinnt der durchgezogene Rahmen aufgrund seiner höheren Priorität.

Beide Deklarationen beeinflussen nun auch die Darstellung der ersten Tabelle. Jedoch kann es bei getrennten Rahmen nicht zu Konflikten kommen: Alle Rahmen werden vollständig gezeichnet. Nun sehen Sie auch den Abstand, der zwischen den Zellen dargestellt wird.

```
/* Rahmen um Tabellenzellen */
th { border: 1px solid black; }

td { border: 1px dotted black; }
```

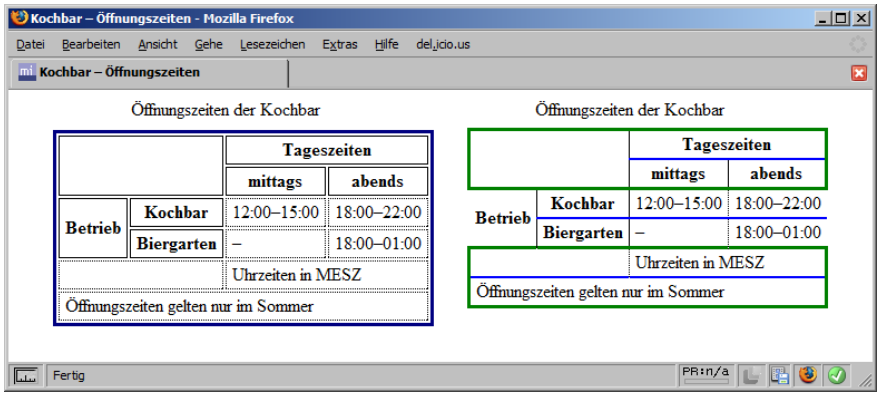


Abbildung 7.36: Dritter Schritt: Rahmen um Zellen

7.11.5 Rahmenabstand: Die Eigenschaft border-spacing

DEFINITION

*Die **Eigenschaft** `border-spacing` gibt beim Tabellenmodell der getrennten Rahmen den Abstand zwischen Zellen an. Die Zwischenräume werden mit den für `table` definierten Hintergrundeigenschaften gefüllt. Beim Modell mit zusammenfallenden Rahmen hat diese Eigenschaft keine Auswirkung.*

Werte	`<Länge> <Länge>` **bzw.** `<Länge>`, `inherit`
Initialwert	0
Vererbung	ja
Anwendung	Tabellen

Wenn Sie zwei Längen angeben, bestimmt die erste den linken und rechten und die zweite den oberen und unteren Zellenabstand. Geben Sie nur eine Länge an, bezieht sie sich auf alle vier Richtungen.

Obwohl der Initialwert `0` ist, hält sich kaum ein moderner Browser daran. Im Allgemeinen wird ein Zellenabstand von 2 Pixeln dargestellt. Wenn Sie diesen Abstand unterdrücken möchten, müssen Sie explizit `border-spacing: 0;` deklarieren.

```
table {
  border-collapse: separate;
  border-spacing: 8px 4px;
  border: 2px solid black;
}
td { border: 1px solid black; }

th { border: 2px solid black; }
```

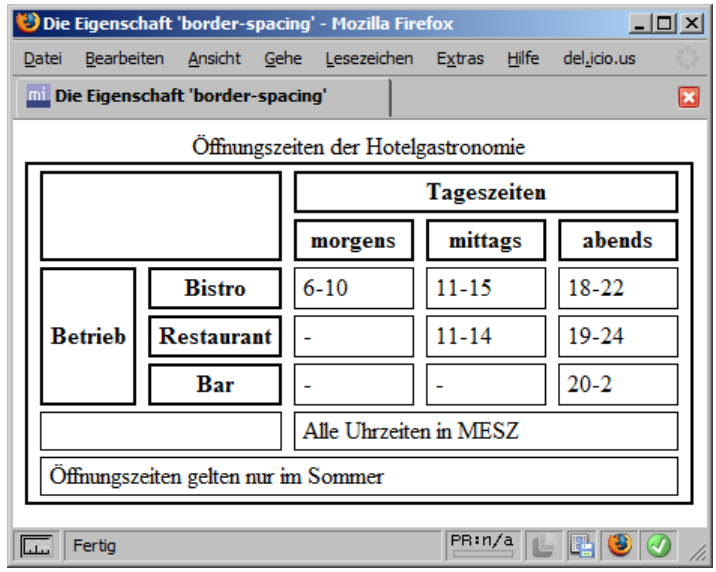


Abbildung 7.37: Die Eigenschaft `border-spacing`

Browserunterstützung

Die Eigenschaft `border-spacing` wird von allen modernen Browsern unterstützt mit folgenden Einschränkungen:

Der IE bis Version 7 erkennt das CSS-Tabellenmodell nur teilweise und stellt deshalb Rahmen an Tabellenzellen immer gleich dar beziehungsweise abhängig davon, ob die Zellen gefüllt sind oder nicht.

7.11.6 Rahmen um leere Zellen: Die Eigenschaft empty-cells

DEFINITION

*Die **Eigenschaft** `empty-cells` gibt beim Tabellenmodell der getrennten Rahmen an, ob leere Zellen dargestellt werden sollen oder nicht. Leer sind Zellen, die keinen Inhalt haben, deren Eigenschaft `visibility` auf `hidden` gesetzt ist oder deren Inhalt durch `display: none`; unterdrückt wird. Beim Modell mit zusammenfallenden Rahmen hat diese Eigenschaft keine Auswirkung.*

Werte	`show`, `hide`, `inherit`
Initialwert	`show`
Vererbung	ja
Anwendung	Tabellenzellen

Standardmäßig werden leere Zellen dargestellt (`show`); ihr Rahmen und ihre Hintergrundeigenschaften werden angezeigt.

Beim Wert `hide` werden leere Zellen nicht dargestellt, das heißt, ihr Rahmen und ihre Hintergrundeigenschaften beeinflussen die Darstellung der Tabelle nicht. Wenn alle Zellen einer Zeile nicht dargestellt werden, hat die Zeile eine Höhe `0`; die Eigenschaft `border-spacing` bezieht sich darüber hinaus auf nur eine Seite der Zeile.

Browserunterstützung

Die Eigenschaft `empty-cells` wird von allen modernen Browsern unterstützt mit folgenden Einschränkungen:

Der IE erkennt das CSS-Tabellenmodell nur teilweise und bis Version 6 deshalb auch nicht die Eigenschaft `empty-cells`. Leere Zellen werden grundsätzlich ohne Rahmen dargestellt (Abhilfe durch ` `), die Eigenschaft `empty-cells` wird nur dann beachtet, wenn eine ganze Reihe der Tabelle leer ist.

IE 7 erfüllt den Wert `hidden` nur, wenn er für alle Zellen einer Tabellenreihe deklariert ist. Ist das nicht der Fall, missachtet er ihn völlig.

Opera lässt bei der Deklaration des Werts `hide` den Hintergrund der Tabellenzelle stehen.

7.11.7 Ausrichtung von Tabellenüberschriften: Die Eigenschaft caption-side

Tabellen generieren eine anonyme, unsichtbare (virtuelle) Box, die sowohl die Box des Tabellencontainers `table` als auch die Box der Tabellenüberschrift `caption` enthält. Die Box der Tabellenüberschrift ist eine Blockbox, die wie eine normale Blockbox innerhalb der anonymen Tabellenbox dargestellt wird und alle vererbbaren Eigenschaften der Tabelle übernimmt.

DEFINITION

*Die **Eigenschaft** `caption-side` gibt an, wie die Tabellenüberschrift im Verhältnis zur Tabelle positioniert werden soll.*

Werte	`top`, `bottom`, `inherit`
Initialwert	`top`
Vererbung	ja
Anwendung	Tabellenüberschriften

Wenn Sie `top` angeben, steht die Tabellenüberschrift über der Tabelle, geben Sie `bottom` an, steht sie darunter.

HINWEIS

In CSS 2.0 gab es die Werte `left` und `right`, die jedoch nicht in CSS 2.1 übernommen wurden. Sie werden zwar von einigen Browsern unterstützt, dennoch sollten Sie sie nicht verwenden. CSS 2.1-konforme Browser (und solche, die mit `left` und `right` ohnehin noch nie etwas anfangen konnten) ersetzen beide Werte durch den Initialwert `top`.

Beispiel

Im folgenden Beispiel wird die Tabellenüberschrift unter die Tabelle positioniert (sie stellt damit eher eine Tabellenunterschrift dar). Da die verschiedenen Browser den Abstand zwischen Tabelle und Tabellenüberschrift (sprich: die Margin-Bereiche) unterschiedlich behandeln, wird `margin` auf `0` gesetzt und der Abstand stattdessen mit `padding` erzeugt.

```
table {
  caption-side: bottom;
  border: 2px solid black;
}
td, th {
  border: 1px solid black;
  padding: 2px 5px;
}
th { border: 2px solid black; }
```

```
caption {
  margin: 0;
  padding: 1em 0;
  text-align: center;
}
```

Browserunterstützung

Die Eigenschaft `caption-side` wird nicht von allen modernen Browsern unterstützt:

Der IE bis Version 7 ignoriert diese Eigenschaft völlig beziehungsweise stellt immer den Initialwert dar.

Auch Konqueror ignoriert diese Eigenschaft völlig beziehungsweise stellt immer den Initialwert dar.

7.12 Generierter Content

In einigen Fällen kann es sinnvoll sein, Dokumentinhalte durch Inhalte zu ergänzen, die nicht explizit im Dokument notiert und somit nicht Teil des Dokumentstammbaums sind. Bei Listen beispielsweise generiert der Browser die Listenelementmarkierung automatisch in Form eines Aufzählungszeichens vor ungeordneten und einer Nummerierung bei Aufzählungslisten, aber spezielle Zeichen oder Zeichenketten könnten auch in anderen Fällen automatisch erzeugt werden, etwa die Zeichenkette »Abbildung: « vor Bildunterschriften oder Nordostpfeile vor externen Links. Wie sich derartige Inhalte per CSS generieren lassen, beschreibe ich in diesem Abschnitt.

7.12.1 Generierter Inhalt: Die Eigenschaft content

DEFINITION

*Die **Eigenschaft** `content` kann zusammen mit den Pseudoelementen `:before` und `:after` dafür verwendet werden, Inhalt in einem Dokument zu erzeugen, der nicht aus dem Dokumentstammbaum kommt. Dabei gibt die Eigenschaft `display` an, mit welcher Art von Box der erzeugte Inhalt dargestellt werden soll.*

Werte	`<Zeichenkette>`, `<URI>`, `<Zähler>`, `open-quote`, `close-quote`, `no-open-quote`, `no-close-quote`, `attr(X)`, `none`, `normal`, `inherit`
Initialwert	leere Zeichenkette (`content: "";`)
Vererbung	nein
Anwendung	Pseudoelemente `:before` und `:after`

Die einzelnen Werte der Eigenschaft `content` erfordern jeweils eine kurze Erklärung.

- `<Zeichenkette>`: Es wird eine Zeichenkette ausgegeben, die innerhalb des Werts in einfache oder doppelte Anführungszeichen eingeschlossen werden muss. Im folgenden Beispiel wird vor externen Links (genauer: vor Elementen vom Typ `a`, deren `href`-Attribut mit `http://` oder `https://` beginnt) ein Nordostpfeil und ein geschütztes Leerzeichen erzeugt (siehe auch Kapitel 6.4.2 »Maskierung«).

```
a[href^="http://"]:before,
a[href^="https://"]:before {
  content: "\2197\00A0";
}
```

- `<URI>`: Es wird eine externe Ressource eingebunden, im Allgemeinen ein Bild. Folgendes Beispiel zeigt die Einbindung eines grafischen Symbols vor externen Links.

```
a[href^="http://"]:before,
a[href^="https://"]:before {
  content: url(/images/externe_links.gif);
}
```

- `<Zähler>`: Die Werte `counter()` und `counters()` generieren zusammen mit den Eigenschaften `counter-increment` und `counter-reset` Zähler ähnlich denen bei geordneten Listen. Die Anwendung wird weiter unten genauer erläutert.

- `open-quote` und `close-quote`, `no-open-quote` und `no-close-quote`: `open-quote` und `close-quote` fügen Anführungszeichen ein, die mittels der Eigenschaft `quotes` definiert wurden. `no-open-quote` und `no-close-quote` geben explizit an, dass keine Anführungszeichen dargestellt werden sollen.

  Ich gehe auf diese Werte wie auch auf die Eigenschaft `quotes` in diesem Buch nicht weiter ein. Die Generierung von Anführungszeichen per CSS sowie die Verwendung von Elementen wie `q`, die dies von selbst tun beziehungsweise tun sollten, ist keine gute Idee und funktioniert browserübergreifend auch nicht besonders gut.

- `attr(X)`: Das Attribut `X` des Elements wird vor oder nach dem Element ausgegeben. Wenn das angegebene Attribut nicht existiert, wird eine leere Zeichenkette ausgegeben. Im folgenden Beispiel wird nach externen Links mit einer Kombination aus generierten Strings und Attributfunktion der URL der Website ausgegeben. Da dies nur bei Printmedien sinnvoll ist, in denen es nicht möglich ist, Verweisen direkt zu folgen, werden die entsprechenden Regeln auf das Medium `print` beschränkt.

```
@media print {
  a {
    color: black;
    background-color: white;
    text-decoration: none;
  }
  a[href^="http://"]:after,
  a[href^="https://"]:after {
    content: " (" attr(href) ")";
  }
}
```

- `none`: Der in der Regel beschriebene zu generierende Inhalt wird *nicht* erzeugt.
- `normal`: Der in der Regel beschriebene zu generierende Inhalt wird *nicht* erzeugt. Dieser Wert wurde in CSS 2.1 mit Blick auf CSS 3 ergänzt, da er sich dort von `none` deutlich unterscheiden soll. Allerdings bleibt abzuwarten, wie sich dieses Modul von CSS 3 entwickeln wird; der »aktuelle« Arbeitsentwurf ist mittlerweile seit über fünf Jahren unverändert.

Browserunterstützung

Die Eigenschaft `content` wird nur lückenhaft von den modernen Browsern unterstützt:

Der IE unterstützt diese Eigenschaft bis Version 7 nicht.

Die Gecko-Engine erkennt den Wert `url()` nicht.

Der Apple-Browser erkennt nur Textstrings und Schlüsselwörter. Die Werte `url()` und `attr()` erkennt er nicht.

Opera unterstützt ab Version 7 alle Werte mit Ausnahme von `url()`.

7.12.2 Zähler: Die Eigenschaften counter-increment und counter-reset

DEFINITION

*Die **Eigenschaft** `counter-increment` dient dazu, einen oder mehrere Zähler um einen bestimmten Wert hochzuzählen.*

Werte	`none`, beliebig viele Paare aus einem eindeutigen Namen und einem `<Integer>`, `inherit`
Initialwert	`none`
Vererbung	nein
Anwendung	alle Elemente

Alle aufgeführten Zähler werden um die zugeordnete Zahl hochgezählt. Wird keine Zahl angegeben, wird der entsprechende Zähler um den Wert 1 hochgezählt. Die definierten Zähler werden in Kombinationen mit den Funktionen `counter()` und `counters()` der Eigenschaft `content` verwendet.

DEFINITION

*Die **Eigenschaft** `counter-reset` dient dazu, einen oder mehrere Zähler auf einen bestimmten Wert einzustellen.*

Werte	`none`, beliebig viele Paare aus einem eindeutigen Namen und einem `<Integer>`, `inherit`
Initialwert	`none`
Vererbung	nein
Anwendung	alle Elemente

Alle aufgeführten Zähler werden auf die zugeordnete Zahl gesetzt. Wird keine Zahl angegeben, wird der entsprechende Zähler auf den Wert 0 gesetzt.

Beispiel

Im folgenden Beispiel werden alle Überschriften erster Ordnung eines Dokuments durchnummeriert. Dazu wird der Zähler zunächst definiert: Der Name des Zählers lautet `abschnitt`, der Inkrementwert, also die Zahl, um die hochgezählt wird, ist nicht vorgegeben und folglich laut Spezifikation 1. Nachfolgend wird der Zähler und anschließend ein Leerzeichen über die `content`-Eigenschaft ausgegeben. Opera-Browser nummerieren die Überschriften nun korrekt.

Gecko-Browser nummerieren jedoch jede Überschrift mit 1. Um dieses Verhalten zu »reparieren«, muss der Zähler am umschließenden Block, in diesem Fall `body`, voreingestellt werden, hier auf den Standardwert 0.

Beim ersten Auftreten des Elements `h1` wird dieses nun um den Schritt 1 auf den Wert 1 hochgezählt, beim zweiten Auftreten erneut um 1 auf 2, beim dritten Auftreten auf 3 und so weiter.

```
h1:before {
  counter-increment: abschnitt;
  content: counter(abschnitt) " ";
}
body { counter-reset: abschnitt; }
```

Standardmäßig werden Zähler mit Dezimalzahlen dargestellt, allerdings ist es auch möglich, die Werte der Eigenschaft `list-style-type` als Zählerdarstellung zu definieren. Um die Überschriften im vorangegangenen Beispiel mit römischen Zahlen zu nummerieren, müssen Sie die entsprechende Regel wie folgt ergänzen:

```
h1:before {
  content: counter(abschnitt, upper-roman) " ";
  counter-increment: abschnitt;
}
```

Zähler sind »selbstverschachtelnd«, das heißt, eine Wiederverwendung eines Zählers in einem untergeordneten Element erzeugt automatisch eine neue Instanz dieses Zählers. Durch diese Festlegung ist es beispielsweise möglich, denselben Zähler für verschachtelte Listen zu verwenden.

Beispiel

Folgendes Beispiel ändert das Zählverhalten verschachtelter geordneter Listen, deren Nummerierung standardmäßig bei 1 beginnt, unabhängig von der Verschachtelungstiefe. Um die Listen in Abhängigkeit von ihrer Verschachtelung zu nummerieren, wird die automatische Nummerierung mittels `list-style-type: none;` abgeschaltet und stattdessen ein eigener Zähler definiert. Die Vorgehensweise ist dieselbe wie bei der Nummerierung der Überschriften. Der einzige Unterschied liegt in der Funktion `counters(zaehler, ".")`, die eine Zeichenkette generiert, die die Werte aller Zähler enthält, getrennt durch eine vorgegebene Zeichenkette, in diesem Fall einen Punkt.

```
ol {
  list-style-type: none;
  counter-reset: zaehler;
}
li:before {
  counter-increment: zaehler;
  content: counters(zaehler, ".");
  margin-right: .5em;
}
```

Browserunterstützung

Die Unterstützung für Zähler ist derzeit leider noch äußerst bescheiden. Nur Opera ab Version 7 und aktuelle Firefox-Browser verstehen die automatische Nummerierung, jedoch unterscheiden sich ihre Implementierungen im Detail.

8 Eine Beispielwebsite: Die Kochbar

Wenn Sie dieses Buch gewissenhaft durchgearbeitet haben, wissen Sie alles, was Sie wissen müssen, um statische Websites zu erstellen. Erinnern Sie sich noch daran, was Ihnen dieses Buch alles beibringen wollte? Die Versprechen beginnen auf Seite 17. Dort sichere ich zu, Ihnen zu zeigen, »wie Sie zugängliche, suchmaschinenfreundliche XHTML-Dokumente erstellen, die so strukturiert sind, dass sie über ein Stylesheet in die gewünschte visuelle Form gebracht werden können«. Weiter geht es einige Abschnitte später, wo Sie die Versprechung finden: »Sie werden lernen, wie Sie Ihre sauber strukturierten und semantisch reichhaltigen XHTML-Dokumente visuell aufbereiten und wie CSS Sie dabei unterstützen kann, ansprechend aussehende Webseiten zu erstellen.« Wohlgemerkt, Sie *werden* lernen, nicht: Sie *sollen* lernen.

Konnte ich meine Versprechen einhalten? Wenn nicht, bringen Sie das Buch zurück (falls Ihr Buchhändler es noch annimmt) oder besser: Lesen Sie aufmerksam dieses Kapitel, in dem ich Ihnen dabei helfen werde (wieder ein Versprechen), Ihr Wissen über XHTML und CSS praktisch anzuwenden. Ich laden Sie ein, mit

mir zusammen eine vollständige, kleine Website aufzubauen, von der Planung über die Umsetzung bis zur Qualitätssicherung.

8.1 Einleitung

Vorbedingungen

Wir stellen uns vor: Irgendwo in einer deutschen Stadt, nennen wir sie Musterstadt, in der Musterstraße gibt es ein kleines Restaurant, die »Kochbar«, das sich fortan auch im Web präsentieren möchte. Wir stellen uns weiter vor, dass es einen zugehörigen Biergarten gibt, dessen Öffnungszeiten sich von denen des Restaurants unterscheiden. Wir gehen ferner davon aus, dass der Besitzer der Kochbar dem Projekt zunächst ein wenig verhalten gegenübersteht und seine Kosten daher möglichst gering halten möchte. Das bedeutet, die Website sollte sich auf das Wesentliche beschränken und nur die Informationen präsentieren, die wirklich notwendig sind. Natürlich soll es möglich sein, die Website später im Erfolgsfall durch weitere Informationen zu ergänzen, etwa zur Geschichte des Hauses, zur Einrichtung, zum Personal oder zu besonderen Veranstaltungen wie Mitternachtsbüfetts, Sommerveranstaltungen im Biergarten oder Silvesterdinner. Als das einzige besondere Angebot soll das Rezept des Tages zum Nachkochen auf der Site präsentiert werden.

Bevor wir anfangen, ist es notwendig, einige Vorüberlegungen anzustellen. Der Systemarchitekt Jens Meiert drückt dies treffend so aus:

> *»Ein gutes Informationsangebot fällt nicht einfach vom Himmel, und es lässt sich nicht schnell zusammenklicken. Nein. Es stellt Ansprüche an Ziele, Inhalte, Struktur, Design, Programmierung, Pflege. Es erfordert Expertise, und dies permanent.«* ***[Meiert 2006-1]***

Planung

Die Website soll potenzielle Gäste ansprechen, die sich über das Restaurantangebot in erreichbarer Nähe informieren möchten. Sie soll den Gast über die Kochbar informieren, ihm die Möglichkeit geben, die Speisekarte und die Öffnungszeiten einzusehen und einen Tisch zu reservieren. Das Rezept des Tages soll Hobbyköche und solche, die es werden wollen, zum regelmäßigen Besuch der Site animieren; denn wer des Öfteren vorbeischaut, der bestellt vielleicht eher mal einen Tisch.

Das schlanke Konzept stellt keine hohen Ansprüche an die Informationsarchitektur. Werfen wir einen Blick darauf, was wir an Anforderungen zusammenstellen können:

- Die Startseite stellt die Kochbar kurz vor und ermöglicht den Zugriff auf die weiteren Informationen.

- Um die Startseite nicht zu sehr mit Informationen zu überladen, gibt es eine weitere Seite »Kochbar«, die einen Blick »in die Töpfe« erlaubt.
- Die Speisekarte informiert den Gast im Vorfeld darüber, welche lukullischen Genüsse ihn erwarten.
- Die Angabe der Öffnungszeiten sorgt dafür, dass der Gast nicht vor verschlossener Tür steht.
- Ein Kontaktformular erlaubt Rückmeldungen und Fragen und ermöglicht die Reservierung eines Tischs.
- Das Rezept des Tages, auf jeder Seite an auffälliger Stelle verlinkt, macht aus jedem Besucher einen potenziellen Meisterkoch.
- Darüber hinaus benötigt jede Website ein Impressum[1], auch die Kochbar.

Screendesign

Wie das inhaltliche Konzept, soll auch das Screendesign schlank und übersichtlich, aber »appetitanregend« sein. Abbildung 8.1 zeigt den Photoshop-Entwurf, auf dem im Wesentlichen vier Bereiche zu erkennen sind:

- ein Kopfbereich mit dem Kochbar-Schriftzug (Logo) und der Navigation,
- eine Spalte auf der linken Seite, im Folgenden »Sidebar« genannt, die Kontaktinformationen und den Verweis auf das Rezept des Tages enthält,
- der Inhaltsbereich rechts und
- die Fußzeile, die Metainformationen über die Seite enthält (hier einen Hinweis auf den Beispielcharakter der Site im Rahmen dieser Einführung sowie die beteiligten Personen).

Neben dem Logo gibt es weitere grafische Elemente:

- Die oberste Zeile ist mit einer Art Laminat-Muster gekachelt.
- Eine sich wiederholende Reihe von Gabeln, die Leckereien aufspießen, bildet einen geschwungenen Übergang vom Kopf- zum Inhaltsbereich.
- Aufgespießtes Gemüse dekoriert auch das Kontaktfeld.
- Das Rezept des Tages wirkt fast wie handgeschrieben durch die aus Schulheften bekannte Linierung und die Unterstreichung der Überschrift.
- Im Inhaltsbereich sorgt verstreutes Gemüse für Frische und Farbigkeit.

1 Informationen zur Impressumpflicht finden Sie bei ***[Ott 2008]***.

Die Farbgebung der Site ist sehr lebhaft[2]. Warme Farben sorgen für eine freundliche Stimmung; so wie man es sich für die Atmosphäre eines Restaurants wünscht. Andere Elemente greifen das Graublau der Gabeln auf, beispielsweise die Navigation und die Fußzeile, kühlen das Design wiederum ein wenig ab und sorgen so für ein angenehmes Gleichgewicht.

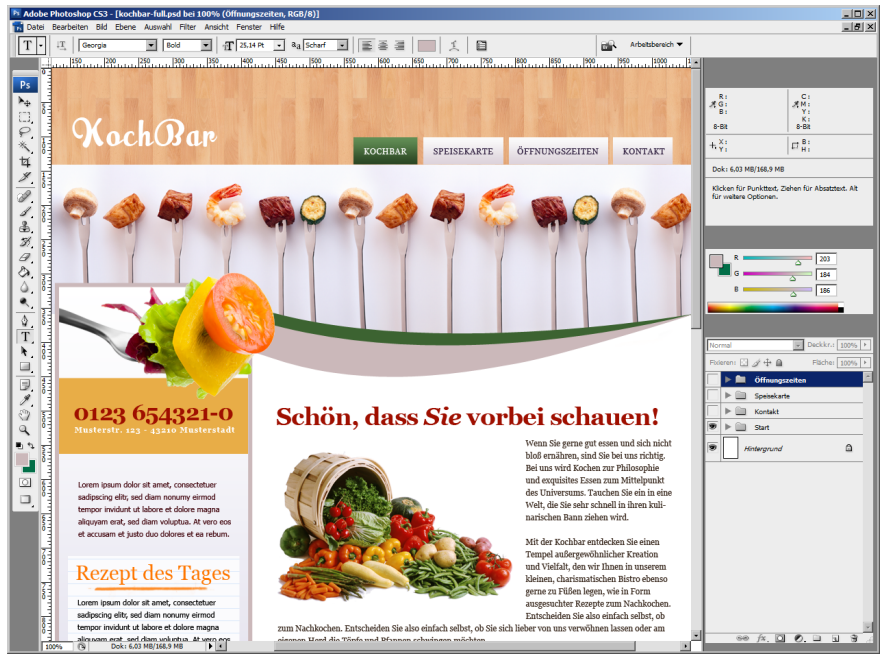


Abbildung 8.1: Startseite

Auf den weiteren Screenshots, Abbildung 8.2 bis Abbildung 8.4, sehen Sie die Entwürfe des Kontaktformulars, der Speisekarte und der Öffnungszeiten jeweils in Form einer Tabelle.

2 Im Schwarz-Weiß-Druck können Sie das leider nicht erkennen. Laden Sie sich daher die Originalabbildungen von der Seite *http://jendryschik.de/wsdev/einfuehrung/buch/* herunter.

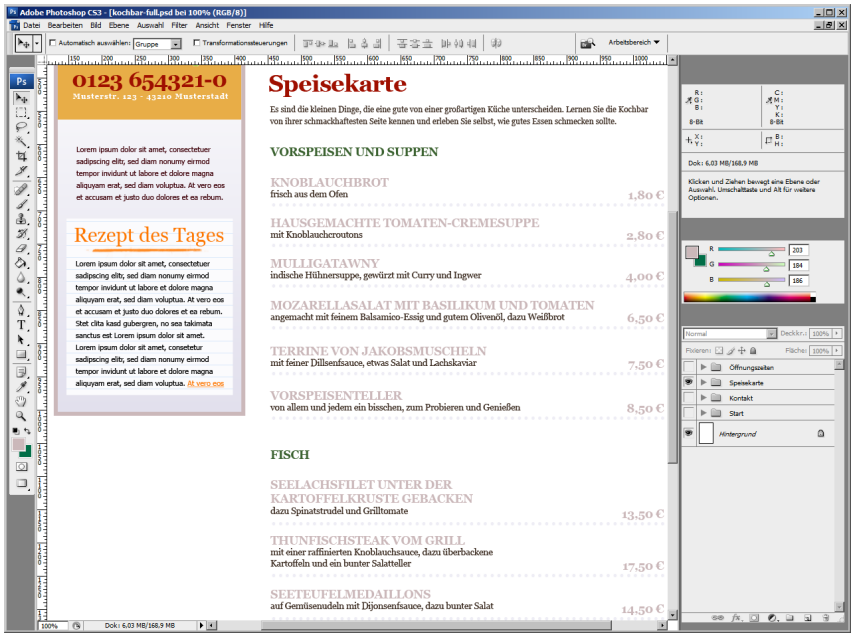


Abbildung 8.2: Speisekarte

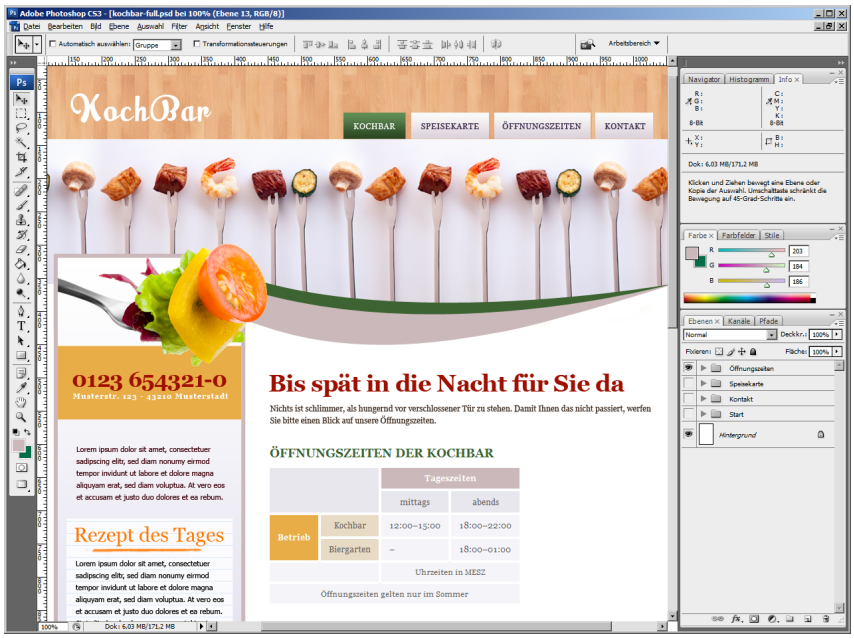


Abbildung 8.3: Öffnungszeiten

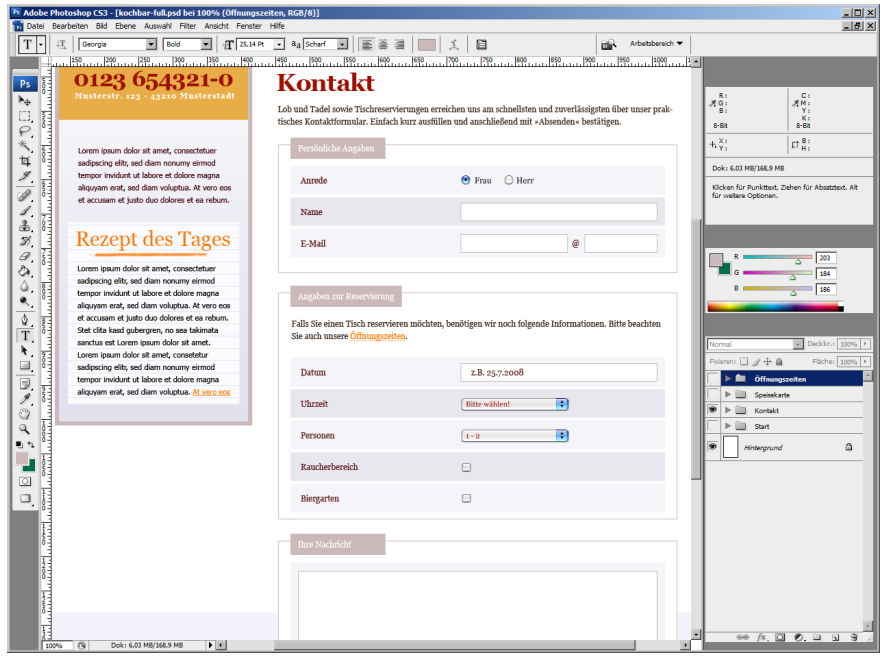


Abbildung 8.4: Kontaktformular mit der Möglichkeit, einen Tisch zu reservieren

Realisierung

Da Sie gerade ein Lehrbuch für XHTML und CSS in Händen halten, sind dies auch die Techniken, mit denen wir die Site umsetzen. Wir erstellen eine kleine, in sich geschlossene Website, die auch auf lokalen Datenträgern laufen wird. Der Fokus liegt ganz klar auf der Textauszeichnung im ersten und auf der Umsetzung des Designs im zweiten Teil.

Qualitätssicherung

Vor Fertigstellung der Kochbar-Website überprüfen wir die technische **Validität** und Konformität aller Ressourcen, sowohl des Markups als auch des Stylesheets.

8.2 Vorbereitung

Wenn Sie maximalen Nutzen aus dem Kochbar-Beispiel ziehen möchten, sollten Sie es Schritt für Schritt nachvollziehen. Bauen Sie die Kochbar parallel zur Lektüre dieses Kapitels auf Ihrem Rechner mit Ihren Werkzeugen per Hand nach! Was benötigen Sie, um anzufangen?

ÜBUNG

Arbeitsumgebung einrichten

1. *Laden Sie sich das Downloadpaket von der Seite http://jendryschik.de/wsdev/einfuehrung/buch/ herunter. Darin finden Sie das vollständige Kochbar-Beispiel, darunter alle Grafiken, die Sie benötigen.*
2. *Erstellen Sie irgendwo auf Ihrem Rechner einen Ordner. Nennen Sie ihn, wie Sie möchten, am besten vielleicht »Kochbar«.*
3. *Öffnen Sie den Texteditor Ihrer Wahl (siehe Kapitel 3.1.2). Ich würde UltraEdit öffnen, Sie vielleicht einen anderen Editor.*
4. *Erstellen Sie ein neues leeres Dokument und speichern Sie es in Ihrem Ordner als »index.html«. Geben Sie dabei UTF-8 als Kodierung an. Abbildung 8.5 zeigt den Speichervorgang mit UltraEdit.*
5. *Öffnen Sie die* `index.html` *in einem modernen Browser Ihrer Wahl.*

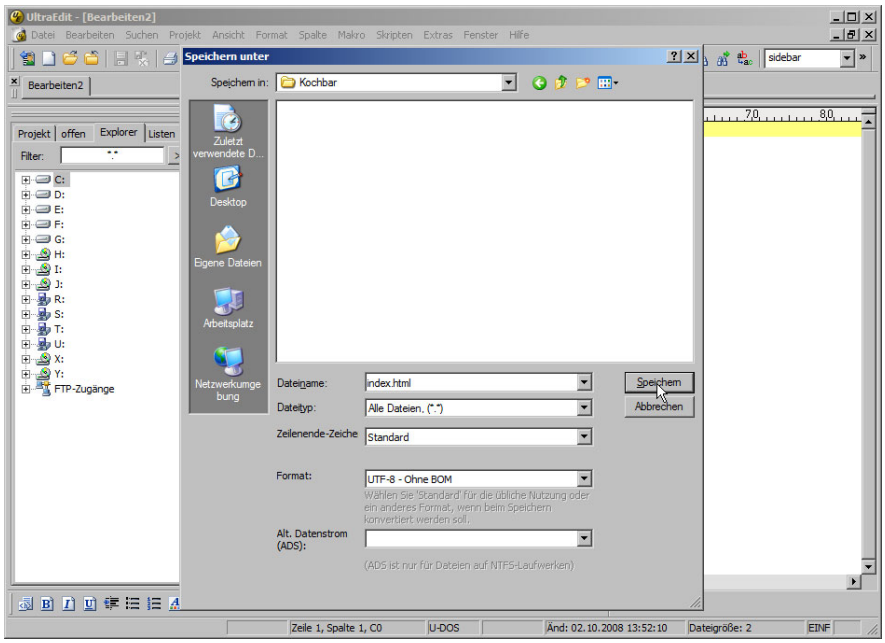


Abbildung 8.5: Leere Datei als `index.html` speichern

Ich werde im Folgenden Firefox verwenden. Ich empfehle Ihnen, stets in einem aktuellen Browser zu testen, solange Sie die Website aufbauen. Firefox hat den Vorteil, dass Sie all die nützlichen Browsererweiterungen verwenden können, die ich Ihnen in Kapitel 3.5 vorgestellt habe. Erst am Ende werden wir uns das Ergebnis in anderen Browsern anschauen und dann gegebenenfalls den einen oder anderen Bug fixen; vor allem in veralteten Versionen des Internet Explorers.

Nun können Sie loslegen! Im Editor nehmen Sie Änderungen vor, speichern diese und aktualisieren die Anzeige im Browser, um direkt die Ergebnisse zu sehen. Fangen wir mit dem XHTML-Grundgerüst der Kochbar an.

8.3 Schritt 1: XHTML-Grundgerüst

Die Kochbar realisieren wir mit XHTML 1.0 Strict. Der Entwurf ist einfach genug, um eine bestmögliche Trennung von Struktur und Darstellung zu ermöglichen.

Die globale Struktur eines XHTML-Dokuments beschreibe ich in Kapitel 4.2. Darauf aufbauend sieht das XHTML-Grundgerüst der Homepage (und aller weiteren Seiten) wie folgt aus:

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

<html xmlns="http://www.w3.org/1999/xhtml"
  xml:lang="de" lang="de">
  <head>
    <title>Kochbar</title>
  </head>

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

Listing 8.1: Grundgerüst der Kochbar-Homepage

Nun fehlen noch wichtige Metaangaben (siehe Kapitel 4.5). Wir müssen die Zeichenkodierung angeben und sollten die Möglichkeit nutzen, Informationen für Suchroboter und Suchmaschinen hinzuzufügen.

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

<html xmlns="http://www.w3.org/1999/xhtml"
  xml:lang="de" lang="de">
  <head>
    <meta http-equiv="Content-Type"
      content="text/html; charset=utf-8" />

    <title>Kochbar</title>

    <meta name="author-personal"
      content="Michael Jendryschik" />
    <meta name="author-mail"
      content="mailto:michael@jendryschik.de" />
```

```
    <meta name="description" content="Vorstellung der Kochbar in
      Musterstadt mit Informationen zur Küche, dem Biergarten
      sowie dem Rezept des Tages." />

    <meta name="language" content="de" />
    <meta name="robots" content="index, follow" />
  </head>

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

Listing 8.2: Grundgerüst mit Metaangaben

Auf weitere Metaangaben verzichten wir zunächst. Bisher haben wir kein Stylesheet eingebunden, also brauchen wir auch keine Stylesheetsprache anzugeben. Schlagwörter können wir bei Bedarf später hinzufügen.

ÜBUNG

XHTML-Grundgerüst erstellen

1. *Fügen Sie das erweiterte Grundgerüst in Ihre* `index.html` *ein, wobei Sie natürlich Ihren Namen und Ihre E-Mail-Adresse bei den Metaangaben notieren. Sie können sich die Listings von der Seite http://jendryschik.de/wsdev/einfuehrung/buch/ herunterladen und den Quelltext daraus kopieren, aber der Lerneffekt ist größer, wenn Sie ihn abschreiben.*

   Bitte übertragen Sie auch alle Leerzeilen und Einrückungen. Gewöhnen Sie sich schon zu Beginn an, den Quelltext sauber und übersichtlich zu strukturieren. Das wird für Sie nützlich sein, wenn Sie sich nach einiger Zeit wieder Ihrem Quelltext widmen, aber auch für Kollegen, die Ihrer Arbeit ergänzen oder weiterführen wollen und sich in Ihren Dokumenten zurechtfinden müssen.

   Es gibt unterschiedliche Wege, Quelltextzeilen einzurücken. Manche Webautoren rücken mit Tabulatoren ein, andere mit Leerzeichen. Ich bevorzuge die Einrückung mittels zweier Leerzeichen und habe meinen Editor entsprechend konfiguriert. In UltraEdit finden Sie die Einstellung in den Optionen im Unterpunkt EDITOR > ZEILENUMBRUCH/TABULATOREN. *Aktivieren Sie die Option »Leerzeichen anstelle von Tabulatoren verwenden« und vergeben Sie den Tabstoppwert 2 sowie 2 bei »Leerzeichen für Einzug«. Falls Sie einen anderen Editor verwenden, schauen Sie, ob Sie ihn ähnlich konfigurieren können.*
2. *Speichern Sie die Änderungen.*
3. *Betrachten Sie das Ergebnis im Browser. Sie sehen eine Darstellung wie in Abbildung 8.6: ein leeres Browserfenster mit dem Titel »Kochbar«.*

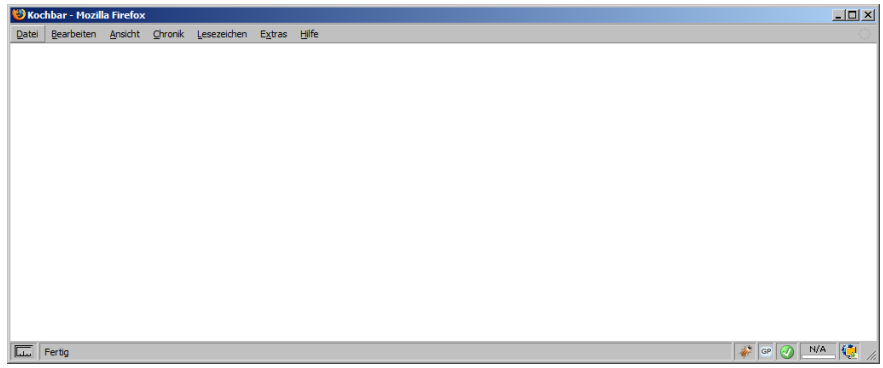


Abbildung 8.6: Schritt 1

Unser Grundgerüst steht. Nun können wir anfangen, uns um die Auszeichnung der Inhalte zu kümmern. Dabei gehen wir einfach von oben nach unten vor: Fangen wir also an mit dem Logo, dem Kochbar-Schriftzug.

8.4 Schritt 2: Logo

Zur Auszeichnung von Grafiken gibt es das `img`-Element. Wir fügen es direkt nach dem öffnenden `<body>`-Tag in unser Grundgerüst ein. Um Sie nicht mit Redundanz zu langweilen und um unsere Wälder zu schonen, verzichte ich bei den folgenden Listings darauf, den gesamten Quelltext anzuführen, sondern beschränke mich auf die Darstellung des `body`-Elements.

```
<body>
  <img src="images/logo.png" alt="Kochbar"
    width="288" height="100" />
</body>
```

Listing 8.3: Auszeichnung des Logos

Abbildung 8.7: Logo

Beachten Sie den Inhalt des alternativen Attributs: Der Text »Kochbar« ist genau der, der auf der Grafik zu lesen ist.

ÜBUNG

Logo auszeichnen

1. *Es würde zu weit führen, Ihnen in diesem Buch über XHTML und CSS hinaus zu vermitteln, wie Sie mittels eines Grafikbearbeitungsprogramms die benötigten Grafiken aus den Vorlagen herausschneiden. Aus diesem Grund finden Sie alle Grafiken in der richtigen Größe in den entsprechenden Ordnern im Downloadpaket[3], sodass Sie diese direkt verwenden können. Erstellen Sie in Ihrem Arbeitsordner einen Unterordner »images« und kopieren Sie das Bild hinein.*
2. *Ergänzen Sie Ihre* `index.html` *mit der Auszeichnung des Logos.*
3. *Speichern Sie und betrachten Sie das Ergebnis im Browser.*

Haben Sie die Web Developer Toolbar installiert? Dann validieren Sie Ihre lokale Datei (Menüpunkt TOOLS > VALIDATE LOCALE HTML). Sie werden eine Fehlermeldung angezeigt bekommen:

> *Line 20, Column 90: document type does not allow element "img" here; missing one of "p", "h1", "h2", "h3", "h4", "h5", "h6", "div", "address", "fieldset", "ins", "del" start-tag*

Was bedeutet das?

Das Element `body` darf laut Spezifikation nur Blockelemente enthalten. `img` ist ein Inline-Element und folglich an dieser Stelle ohne umgebendes Blockelement nicht erlaubt. Der Validator gibt Ihnen bereits an, welche Elemente Sie verwenden dürfen. Die Frage ist nun: Welches dieser Elemente eignet sich als Container für das Logo? `div` oder `p` wären geeignete Kandidaten, ich bevorzuge jedoch das Element `h1`. Ich denke, der Schriftzug »Kochbar« ist eine gute Hauptüberschrift. Wie weit wir mit diesem Ansatz kommen und ob es gegebenenfalls weitere Hauptüberschriften gibt, sehen wir im Folgenden. Also:

```
<body>
  <h1><img src="images/logo.png" alt="Kochbar"
    width="288" height="100" /></h1>
</body>
```

Listing 8.4: Es ist eine gute Idee, das Logo als Hauptüberschrift auszuzeichnen

3 *http://jendryschik.de/wsdev/einfuehrung/buch/*

ÜBUNG

Logo aus Hauptüberschrift

1. *Ergänzen Sie Ihre* `index.html`.
2. *Speichern Sie und betrachten Sie das Ergebnis im Browser. Die Seite sollte so aussehen wie die in Abbildung 8.8 gezeigte.*

   Im Firefox können Sie über die Web Developer Toolbar die Grafiken durch ihre Alternativtexte ersetzen (IMAGES > REPLACE IMAGES WITH ALT ATTRIBUTES). Auf diese Weise sehen Sie, dass »Kochbar« tatsächlich wie eine Überschrift aussieht.

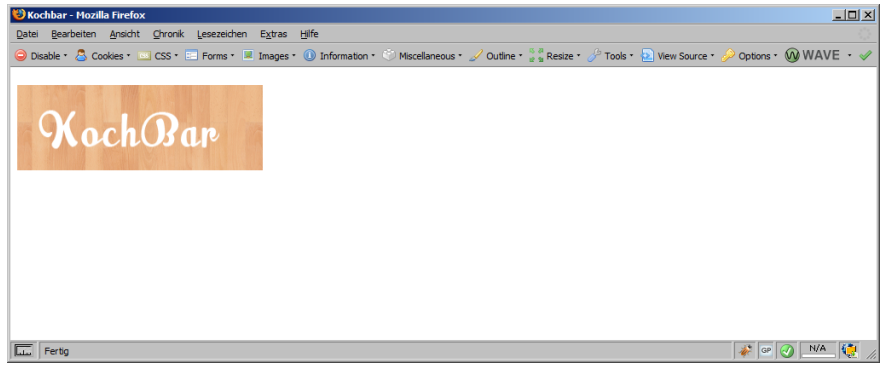


Abbildung 8.8: Schritt 2

Als Nächstes kommen wir zur Hauptnavigation und – nach der Auszeichnung des Logos als Überschrift erster Ordnung – zur nächsten Überraschung: Wir werden die Navigation als ungeordnete Liste (`ul`) auszeichnen!

8.5 Schritt 3: Hauptnavigation

Wenn Sie sich das Screendesign der Kochbar anschauen, sehen Sie fünf nebeneinanderstehende Navigationspunkte; der jeweils aktive Punkt zeigt sich in weißer Schrift auf grünem Grund, alle anderen Dunkel- auf Hellgrau. Es liegen also fünf gleichrangige Einträge vor. Das semantisch richtige Element zur deren Auszeichnung ist eine ungeordnete Liste, also »eine Sammlung verschiedener Elemente, in der es keine Reihenfolge gibt« (vgl. Kapitel 5.6.1). Lassen Sie sich nicht dadurch verwirren, dass sich die Listenpunkte nebeneinander und nicht, wie die meisten Listen, untereinander befinden.

Die Navigation sieht also wie folgt aus:

```
<body>
  <h1><img src="images/logo.png" alt="Kochbar"
    width="288" height="100" /></h1>

  <ul>
    <li><a href="index.html">Startseite</a></li>
    <li><a href="ueber-uns.html">Kochbar</a></li>
    <li><a href="speisekarte.html">Speisekarte</a></li>
    <li><a href="oeffnungszeiten.html">Öffnungszeiten</a></li>
    <li><a href="kontakt.html">Kontakt</a></li>
  </ul>
</body>
```

Listing 8.5: Navigationen können Sie fast immer als ungeordnete Liste auszeichnen

Der Quelltext ist noch nicht ganz perfekt: Eine Navigation sollte niemals einen Link auf sich selbst enthalten. Das bedeutet, wenn wir uns auf der Startseite befinden, sollte der Navigationspunkt »Startseite« nicht verlinkt sein, um Nutzern zu ersparen, dem Link zu folgen, um sich dann wundern zu müssen, dass sich offensichtlich nichts geändert hat. Das Element `strong` mit einer entsprechenden Anmerkung über das Universalattribut `title` ist hierbei das Element unserer Wahl.

```
<body>
  <h1><img src="images/logo.png" alt="Kochbar"
    width="288" height="100" /></h1>

  <ul>
    <li><strong title="Hier befinden Sie sich
      gerade.">Startseite</strong></li>
    <li><a href="ueber-uns.html">Kochbar</a></li>
    <li><a href="speisekarte.html">Speisekarte</a></li>
    <li><a href="oeffnungszeiten.html">Öffnungszeiten</a></li>
    <li><a href="kontakt.html">Kontakt</a></li>
  </ul>
</body>
```

Listing 8.6: Navigationslinks sollten niemals auf sich selbst verweisen

ÜBUNG

Hauptnavigation als Liste

1. *Ergänzen Sie Ihre* `index.html`.
2. *Speichern Sie und betrachten Sie das Ergebnis im Browser. Die Seite sollte so aussehen wie Abbildung 8.9.*
3. *Fahren Sie mit der Maus über den hervorgehobenen Navigationslink und schauen Sie sich den Tooltipp an.*

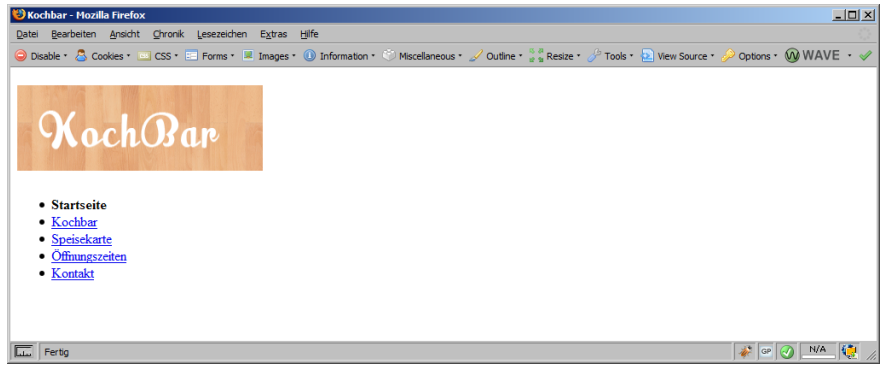


Abbildung 8.9: Schritt 3

An dieser Stelle sei noch mal gesagt: Lassen Sie sich nicht davon beeindrucken, dass die Navigation derzeit noch überhaupt nicht so aussieht wie im Screendesign. Wir haben bisher alles richtig gemacht. Der Rest folgt, wenn wir uns um das Styling der Site per CSS kümmern.

Mit dem Kopfbereich sind wir fertig: Logo und Navigation haben wir ausgezeichnet. Sie sind vielleicht versucht, sich nun mit den Grafiken zu beschäftigen, die den Übergang zwischen Kopf- und Inhaltsbereich bilden, aber die lassen wir zunächst außen vor. Es ist einfacher, wenn wir uns zunächst nur den tatsächlichen Inhalten widmen, also die weiteren Texte auszeichnen.

Kommen wir zur Sidebar.

8.6 Schritt 4: Sidebar

Die Sidebar beinhaltet zum einen das Kontaktfeld, zum anderen den Teaser auf das Rezept des Tages.

Das Kontaktfeld ist eine gute Übung, Darstellung von Semantik zu unterscheiden. Nicht immer ist das, was wie eine Überschrift aussieht (groß und fett), auch tatsächlich eine.

Werfen wir einen Blick auf den Text, der im Kontaktfeld steht, losgelöst von Formatierungen wie Schriftgröße oder -gewicht. Das hilft uns, ihn semantisch einzuordnen.

Kontaktdaten
0123 654321-0
Musterstr. 123 – 43210 Musterstadt

»Kontaktdaten« ist offensichtlich die Überschrift für die darunterstehenden Kontaktinformationen. Aber welche Art von Überschrift? In Kapitel 5.1.1 weise ich darauf hin, dass Überschriften eine Hierarchie bilden sollen. Eine Überschrift erster Ordnung gibt es bereits, das Logo, also müssen wir nun das Element `h2` verwenden.

Wie die Telefonnummer und die Adresse auszuzeichnen sind, darüber lässt sich streiten. Als eine ungeordnete Liste von Kontaktdaten (`ul`)? Das erscheint mir »übertrieben semantisch« zu sein. Mit `div`? Eher zu wenig Semantik. Hier mag `p` der richtige Elementtyp sein.

```
<body>
  <h1><img src="images/logo.png" alt="Kochbar"
    width="288" height="100" /></h1>

  <ul>
    <li><strong title="Hier befinden Sie sich
      gerade.">Startseite</strong></li>
    <li><a href="ueber-uns.html">Kochbar</a></li>
    <li><a href="speisekarte.html">Speisekarte</a></li>
    <li><a href="oeffnungszeiten.html">Öffnungszeiten</a></li>
    <li><a href="kontakt.html">Kontakt</a></li>
  </ul>

  <h2>Kontaktdaten</h2>

  <p>0123 654321-0</p>
  <p>Musterstr. 123 - 43210 Musterstadt</p>
</body>
```

Listing 8.7: Auszeichnung der Kontaktinformationen

ÜBUNG

Kontaktdaten und Teaser

1. *Ergänzen Sie Ihre* `index.html`.
2. *Überlegen Sie sich nun, wie Sie den Teaser auf das Rezept des Tages auszeichnen wollen. Gibt es Überschriften? Falls ja, welche?*

Sind Sie auch zu folgendem Ergebnis gekommen?

```
<body>
  <h1><img src="images/logo.png" alt="Kochbar"
    width="288" height="100" /></h1>

  <ul>
    <li><strong title="Hier befinden Sie sich
      gerade.">Startseite</strong></li>
    <li><a href="ueber-uns.html">Kochbar</a></li>
    <li><a href="speisekarte.html">Speisekarte</a></li>
    <li><a href="oeffnungszeiten.html">Öffnungszeiten</a></li>
    <li><a href="kontakt.html">Kontakt</a></li>
  </ul>

  <h2>Kontaktdaten</h2>

  <p>0123 654321-0</p>
  <p>Musterstr. 123 - 43210 Musterstadt</p>

  <h2>Rezept des Tages</h2>

  <h3>Mulligatawny</h3>

  <p>Eine ursprünglich sehr scharfe indische Hühnersuppe,
    gewürzt mit Curry und Ingwer. Wir empfehlen die sanfte,
    englische Version. <a href="rezepte/mulligatawny.html">
    Rezept ansehen</a></p>
</body>
```

Listing 8.8: Auszeichnung des Teasers auf das Rezept des Tages

»Rezept des Tages« ist eindeutig eine Überschrift. Sie ist gleichrangig mit der Überschrift »Kontaktdaten«, also ebenfalls eine Überschrift zweiter Ordnung. »Mulligatawny« ist dieser Überschrift unter- und dem darunterstehenden Absatz übergeordnet: Eine Auszeichnung als Überschrift dritter Ordnung bietet sich an.

Folgende Auszeichnung hat übrigens ebenfalls einen gewissen Charme:

```
<dl>
  <dt>Mulligatawny</dt>
  <dd>Eine ursprünglich sehr scharfe indische Hühnersuppe,
    gewürzt mit Curry und Ingwer. Wir empfehlen die sanfte,
    englische Version. <a href="rezepte/mulligatawny.html">
    Rezept ansehen</a><dd>
</dl>
```

Allerdings bin ich kein Freund von Listen, die nur aus einem Listenpunkt bestehen. Die Auszeichnung mit `h3` und `p` gefällt mir besser – belassen wir es dabei.

Mit der Auszeichnung der Sidebar-Inhalte sind wir zwar fertig, allerdings ist das Ergebnis noch nicht ganz zufriedenstellend. Was haben wir erreicht? Eine lose Abfolge von Überschriften und Absätzen. Es ist nicht erkennbar, dass es sich um die Sidebar handelt, die zwei voneinander getrennte Inhalte umfasst. Diese Informationen sollten wir abschließend mittels einiger `div`-Elemente hinzufügen.

```
<body>
  <h1><img src="images/logo.png" alt="Kochbar"
    width="288" height="100" /></h1>

  <ul>
    <li><strong title="Hier befinden Sie sich
      gerade.">Startseite</strong></li>
    <li><a href="ueber-uns.html">Kochbar</a></li>
    <li><a href="speisekarte.html">Speisekarte</a></li>
    <li><a href="oeffnungszeiten.html">Öffnungszeiten</a></li>
    <li><a href="kontakt.html">Kontakt</a></li>
  </ul>

  <div id="sidebar">
    <div id="contact">
      <h2>Kontaktdaten</h2>

      <p>0123 654321-0</p>
      <p>Musterstr. 123 - 43210 Musterstadt</p>
    </div>

    <div id="teaser">
      <h2>Rezept des Tages</h2>

      <h3>Mulligatawny</h3>

      <p>Eine ursprünglich sehr scharfe indische Hühnersuppe,
        gewürzt mit Curry und Ingwer. Wir empfehlen die sanfte,
        englische Version. <a href="rezepte/mulligatawny.html">
```

```
            Rezept ansehen</a></p>
        </div>
      </div>
    </body>
```

Listing 8.9: `div`-Elemente fassen zusammen, was zusammengehört

Um Elemente zu Gruppen zusammenzufassen, hätten die `div`-Elemente allein genügt; die Bezeichner `sidebar`, `contact` und `teaser` hätten wir nicht hinzufügen müssen. Allerdings erläutern diese den Zweck der `div`-Elemente für jene, die einen Blick in den Quelltext werfen, und werden uns später zunutze sein, wenn es um das Styling der Website geht.

ÜBUNG

Feinstrukturierung der Sidebar

1. *Ergänzen Sie Ihre* `index.html`.
2. *Speichern Sie und betrachten Sie das Ergebnis im Browser. Vergleichen Sie es mit Abbildung 8.10.*
3. *Entfernen Sie die* `div`*-Elemente oder fügen Sie weitere hinzu. Ändern Sie die Namen der Bezeichner. Sie werden feststellen, dass die Feinstrukturierung keinen Einfluss auf die Darstellung des Dokuments hat – zumindest so lange, wie wir noch kein Stylesheet eingebunden haben.*

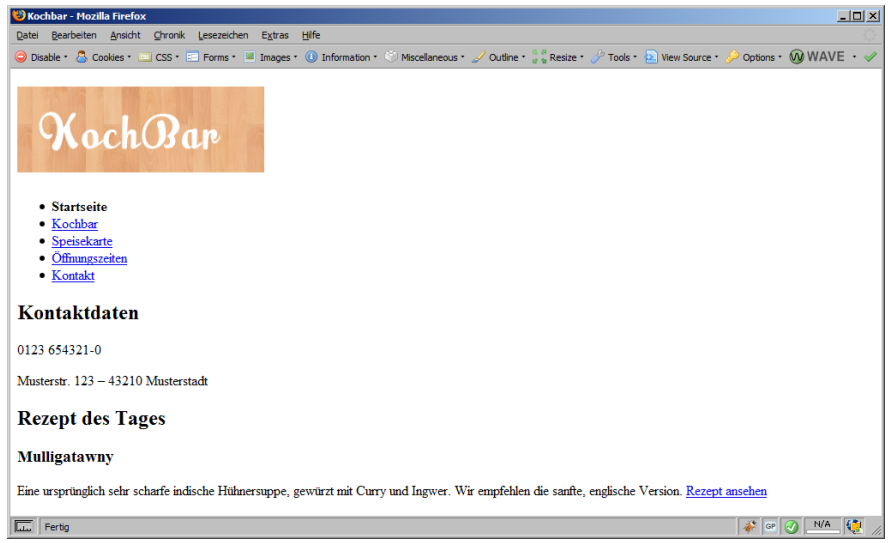


Abbildung 8.10: Schritt 4

Kümmern wir uns als Nächstes um den Inhaltsbereich.

8.7 Schritt 5: Content und Fußzeile

Der Content besteht aus einer Überschrift, einer Illustration und einigen Absätzen mit Links. Wir zeichnen alles der Reihe nach aus und packen die Elemente in einen `div`-Container.

Es ist allgemein üblich, die Content-Überschrift als Überschrift erster Ordnung auszuzeichnen.

```
<body>
  <h1><img src="images/logo.png" alt="Kochbar"
    width="288" height="100" /></h1>

  <ul>
    <li><strong title="Hier befinden Sie sich
      gerade.">Startseite</strong></li>
    <li><a href="ueber-uns.html">Kochbar</a></li>
    <li><a href="speisekarte.html">Speisekarte</a></li>
    <li><a href="oeffnungszeiten.html">Öffnungszeiten</a></li>
    <li><a href="kontakt.html">Kontakt</a></li>
  </ul>

  <div id="sidebar">
    <div id="contact">
      <h2>Kontaktdaten</h2>

      <p>0123 654321-0</p>
      <p>Musterstr. 123 - 43210 Musterstadt</p>
    </div>

    <div id="teaser">
      <h2>Rezept des Tages</h2>

      <h3>Mulligatawny</h3>

      <p>Eine ursprünglich sehr scharfe indische Hühnersuppe,
        gewürzt mit Curry und Ingwer. Wir empfehlen die sanfte,
        englische Version. <a href="rezepte/mulligatawny.html">
        Rezept ansehen</a></p>
    </div>
  </div>

  <div id="content">
    <h1>Schön, dass Sie vorbei schauen!</h1>

    <p><img src="images/gemuese.jpg" width="354"
      height="233" alt="" /></p>

    <p>Wenn Sie gerne gut essen und sich nicht bloß (...)</p>
```

```
      <p>Mit der Kochbar entdecken Sie einen Tempel (...)</p>

      <p>Auf unserer Website erfahren Sie
        <a href="ueber-uns.html">mehr über die Kochbar</a>, können
        einen kleinen Blick auf unsere
        <a href="speisekarte.html">Speisekarte</a> werfen, eine
        <a href="kontakt.html">Online-Reservierung</a> Ihres
        Tisches bei uns vornehmen sowie das eine oder andere
        interessante Rezept entdecken. Sollten Sie überdies noch
        Fragen haben, finden Sie unsere Kontaktdaten ebenso
        schnell wie ein praktisches Formular zum
        <a href="kontakt.html">Versenden einer Nachricht an
        uns</a>.</p>

      <p>Viel Spaß beim Stöbern und Ausprobieren und hoffentlich
        bis bald bei uns in der Kochbar!</p>
    </div>
  </body>
```

Listing 8.10: Content

Die Auszeichnung der Fußzeile stellt uns vor keine neuen Herausforderungen: Sie besteht aus zwei Absätzen, deren Zusammengehörigkeit wir mittels eines umschließenden `div`-Elements ausdrücken.

```
  <body>
    <h1><img src="images/logo.png" alt="Kochbar"
      width="288" height="100" /></h1>

    <ul>
      <li><strong title="Hier befinden Sie sich
        gerade.">Startseite</strong></li>
      <li><a href="ueber-uns.html">Kochbar</a></li>
      <li><a href="speisekarte.html">Speisekarte</a></li>
      <li><a href="oeffnungszeiten.html">Öffnungszeiten</a></li>
      <li><a href="kontakt.html">Kontakt</a></li>
    </ul>

    <div id="sidebar">
      <!-- Sidebar -->
    </div>

    <div id="content">
      <!-- Inhalte -->
    </div>

    <div id="footer">
    <p>Beispiel-Website zur
      <a href="http://jendryschik.de/wsdev/einfuehrung/">
```

```
      Einführung in XHTML, CSS und Webdesign</a> von
      <a href="http://jendryschik.de/michael/">Michael
      Jendryschik</a> - © 2000-2008 -
      <a href="impressum.html">Impressum</a></p>

    <p>Design von <a href="http://www.pixelgraphix.de/">Manuela
      Hoffmann</a>, Texte von <a href="http://www.lindah.de/">Ralf
      Koyro</a>, Speisekarte von
      <a href="http://www.roterochs.de/">Konrad Scheller</a></p>
    </div>
  </body>
```

Listing 8.11: Fußzeile

ÜBUNG

Auszeichnung der Inhalte

1. *Ergänzen Sie Ihre* `index.html` *um den Content und die Fußzeile.*
2. *Speichern Sie und betrachten Sie das Ergebnis im Browser. Es sollte so aussehen, wie in Abbildung 8.11 dargestellt.*
3. *Validieren Sie die Seite. Wenn Sie alles richtig übertragen haben, sollte das Dokument fehlerfrei sein (siehe Abbildung 8.12).*

Abbildung 8.11: Schritt 5

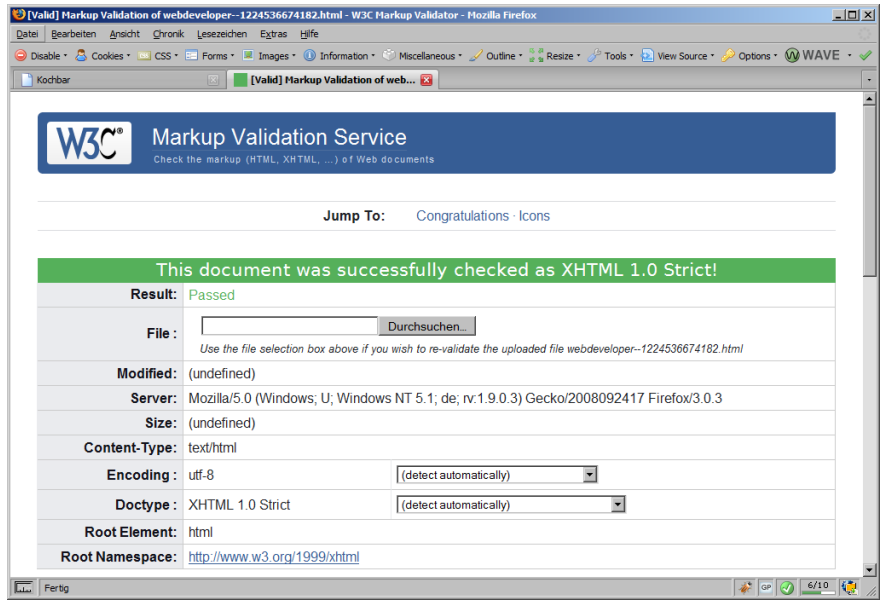


Abbildung 8.12: Der W3C-Validator bescheinigt uns, dass wir keine Fehler gemacht haben

Wir haben nun alle Inhalte der Startseite der Kochbar ausgezeichnet. Um die Unterseiten werden wir uns später kümmern. Widmen wir uns nun dem Styling.

8.8 Schritt 6: Styling der Startseite

In diesem Abschnitt werden wir die Kochbar-Startseite mit rund 250 Zeilen CSS und einigen Grafiken so gestalten, dass ihre Darstellung in modernen Browsern dem Entwurf entspricht. Dabei gehen wir Schritt für Schritt vor.

Wenn Sie die erste Auflage meines Buchs oder die dritte Version meiner Onlineeinführung gelesen haben, erinnern Sie sich vielleicht daran, dass das Kochbar-Beispiel kein eigenes Kapitel war, sondern im Verlauf des Buchs in vielen Einzelschritten aufgebaut wurde. Diese Schritte waren »eigenschaftenorientiert«. Am Ende jedes Abschnitts der CSS-Kapitel habe ich die soeben vorgestellten Eigenschaften angewendet, das heißt, ich habe zunächst Farben und Hintergründe der Kochbar definiert, anschließend Schrift und Texte, danach Abstände und Rahmen und so weiter. Die Leser konnten so direkt in der Praxis ausprobieren, was sie gerade gelernt haben. Ich glaubte damals, das sei eine gute Idee, allerdings habe ich etwas Wichtiges übersehen: Kein Webautor geht so vor!

Die meisten Webautoren schreiben ihre Stylesheets vom Allgemeinen zum Speziellen. Das Stylesheet beginnt mit globalen Einstellungen wie Browser-

Reset, Bugfixes oder Schriftskalierung. Dann folgen Regeln, die allgemeine Vorgaben enthalten; Bezeichner oder Klassennamen sollten dabei ebenso wenig auftauchen wie Kontextselektoren. Formatiert werden Links, Überschriften, Absätze, Tabellen und andere Elementtypen. Dann folgen die spezifischeren Regeln für die einzelnen Layoutelemente: Kopfbereich, Navigation, Logo, Kontakt, Inhalt, Fußzeile. Abschließend werden Regeln für Elemente aufgeführt, die nicht auf allen Seiten der Website vorkommen. So kann es beispielsweise Produkttabellen geben, die sich von gewöhnlichen Datentabellen unterscheiden, oder Sitemaps, die mit verschachtelten Listen realisiert werden und anders aussehen sollen als einfache Aufzählungslisten.

In diesem Buch lernen Sie, Stylesheets so zu schreiben, wie echte Profis es tun – schließlich möchte ich, dass Sie irgendwann selbst ein Profi werden!

ÜBUNG

Stylesheet erstellen und einbinden

1. *Erstellen Sie ein neues leeres Dokument und speichern Sie es in Ihrem Arbeitsordner als »default.css«. Geben Sie dabei UTF-8 als Kodierung an.*
2. *Verknüpfen Sie die* `index.html` *mit dem Stylesheet, indem Sie folgendes* `link`*-Element direkt vor dem schließenden* `</head>` *einfügen:*

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

8.8.1 Vorgehensweise und Coderichtlinien beim Schreiben eines Stylesheets

Es ist sinnvoll, beim Schreiben von CSS nach bestimmten Regeln vorzugehen und sich konsequent an diese Regeln zu halten. So werden Ihre Stylesheets einheitlich und übersichtlich und Ihre Arbeit dadurch effizienter. Andere Autoren, die dieselben Richtlinien anwenden, finden sich in Ihrem Stylesheet sofort zurecht – und umgekehrt.

Gedanken dazu, wie Webautoren ihre Stylesheets sinnvoll aufbauen können, haben sich schon viele Autoren gemacht. Klaus Langenberg beschreibt, wie Sie ein Stylesheet immer nach denselben Prinzipien aufbauen und sich stets strikt an eine einmal etablierte Ordnung halten ***[Langenberg]***. Dieser Artikel ist ebenso lesenswert wie ***[Meiert 2006-2]***.

Wir halten uns an folgende Richtlinien:

- **Deklarationen sind aus Gründen der Übersichtlichkeit einzurücken.** Ich bevorzuge eine Einrückung mit zwei Leerzeichen. Selektoren und schlie-

ßende Klammern werden nicht eingerückt. Besteht die Regel aus nur einer Deklaration, kann sie in einer Zeile notiert werden.

```
/* Nicht gut */
p.warnung {
color: red;
}

/* Besser */
p.warnung {
  color: red;
}
p.warnung { color: red; }
```

- **Die öffnende Klammer steht immer am Ende des Selektors oder der At-Regel; die schließende Klammer befindet sich stets in einer eigenen Zeile.**

```
/* Nicht gut */
p.warnung
{
  color: red; }
```

- **Innerhalb einer Deklaration steht nach dem Doppelpunkt ein Leerzeichen.**

```
p.warnung {
  color:red; /* Nicht gut */
  background-color: white; /* Besser */
}
```

- **Jede Deklaration wird mit einem Semikolon abgeschlossen.** Das gilt auch und ganz besonders für die letzte Deklaration innerhalb eines Deklarationsblocks.

```
/* Nicht gut */
p.warnung {
  color: red;
  background-color: white
}

/* Besser */
p.warnung {
  color: red;
  background-color: white;
}
```

- **Regeln, die zusammengehören, werden zusammen notiert.** Während andere Autoren eine alphabetische Sortierung bevorzugen und sich damit unnötige Arbeit aufhalsen, bevorzuge ich es, die Regeln so zu notieren, wie

es sich im Laufe der Arbeit an dem Stylesheet ergibt. Ich achte einzig darauf, logisch zusammengehörige Regeln zu gruppieren, beispielsweise `position`, `top` und `left` oder `width` und `height`.

- **Stylesheets sind nach den Regeln von CSSDOC (siehe Kapitel 6.4.3) zu kommentieren.** Vor allem Workarounds und Browserhacks sind zu kommentieren, ebenso besonders kritische Stellen des Stylesheets, an denen jede Änderung zu absehbaren Problemen führen kann.

8.8.2 Globale Einstellungen

Nun können wir damit anfangen, das Stylesheet aufzubauen. Als Erstes deklarieren wir die Zeichenkodierung. Anschließend folgt der CSSDOC-Dateikommentar, der einige Informationen über die Website, den Autor sowie Versionsinformationen enthält.

```
@charset "utf-8";
/**
 * Einführung in XHTML, CSS und Webdesign. Kochbar.
 *
 * @link    http://kochbar.jendryschik.de
 * @package kochbar
 * @version 2.0
 */
```

Listing 8.12: At-Regel und Dateikommentar

Kümmern wir uns nun um die globalen Einstellungen und die Behandlung von Browserbugs. Wir beginnen mit einem kleinen Trick:

```
/**
 * Globale Einstellungen und Bugfixes
 *
 * @section Browser-Reset
 */

/* Zurücksetzen der Innen- und Außenabstände */
* {
  margin: 0;
  padding: 0;
}
```

Listing 8.13: Browser-Reset

Dadurch wird der standardmäßig vorgegebene Außen- und Innenabstand aller Elemente auf 0 gesetzt. Das ist sinnvoll, denn verschiedene Browser verwenden unterschiedliche Standardvorgaben für Margin und Padding, vor allem bei Formularelementen und Listen. Durch das Zurücksetzen der Abstände erhalten wir eine konsistente, browserübergreifende Darstellung.

> > > HINWEIS

Wenn Sie sich die Stylesheets handwerklich guter Websites anschauen, werden Sie häufig auf Eric Meyers Browser-Reset[4] oder Tripoli[5] stoßen. Dabei handelt es sich um eine Handvoll CSS-Regeln, die alle herstellerspezifischen Einstellungen zurücksetzen, nicht nur Außen- und Innenabstände. Die Darstellung aller Elemente wird so vereinheitlicht; Sie beginnen browserübergreifend mit denselben Voraussetzungen. Viele Webautoren finden es bequem und zeitsparend, diese oder andere Browser-Resets einzusetzen. Wir verzichten im Rahmen dieses Beispiels darauf; die Kochbar ist einfach genug, um ohne CSS Dritter zurechtzukommen.

ÜBUNG

Browser-Reset

1. *Fügen Sie das bis hierhin aufgeführte CSS in Ihre noch leere* `default.css` *ein.*
2. *Speichern Sie das Stylesheet und betrachten Sie die Kochbar-Startseite im Browser. Vergleichen Sie Abbildung 8.11 und Abbildung 8.13 miteinander und schauen Sie sich an, wie die Elemente ohne Außen- und Innenabstände »zusammenfallen«.*
3. *Wenn Sie möchten, probieren Sie aus, wie die Seite aussieht, wenn Sie die vorgestellten Browser-Reset-Stylesheets einbinden. Binden Sie nur diese Stylesheets ein oder kommentieren Sie unseren kurzen Browser-Reset vorübergehend aus.*

Schriftart und Schriftgröße

Nun kümmern wir uns um Schriftart und -größe. Die durch den Entwurf vorgegebene Schrift ist Georgia; als Alternative vorgesehen sind Palatino und Times New Roman – alles Serifenschriftarten. In der Photoshop-Datei können wir nachmessen, dass die Schrift 12 px groß ist. Die meisten Browser haben eine voreingestellte Schriftgröße von 16 px. Wir möchten aber, dass die Schrift nur drei Viertel dieser Größe einnimmt, also reduzieren wir die Schriftgröße auf 75 Prozent. Die Zeilenhöhe beträgt 20 px, was etwa 167 Prozent entspricht – genauer brauchen wir nicht zu sein. Wir machen Gebrauch von dem in Kapitel 7.3.5 vorgestellten Trick und deklarieren die Schriftgröße in zwei Schritten:

```
/**
 * Schriftgröße kontrollieren
 */
body, td, th { font: normal 75%/167% Georgia, Palatino, "Times
New Roman", serif; }
td, th { f\ont-size: 100%; }
```

Listing 8.14: Deklaration von Schriftart und -größe

4 *http://meyerweb.com/eric/thoughts/2007/05/01/reset-reloaded/*
5 *http://devkick.com/lab/tripoli/*

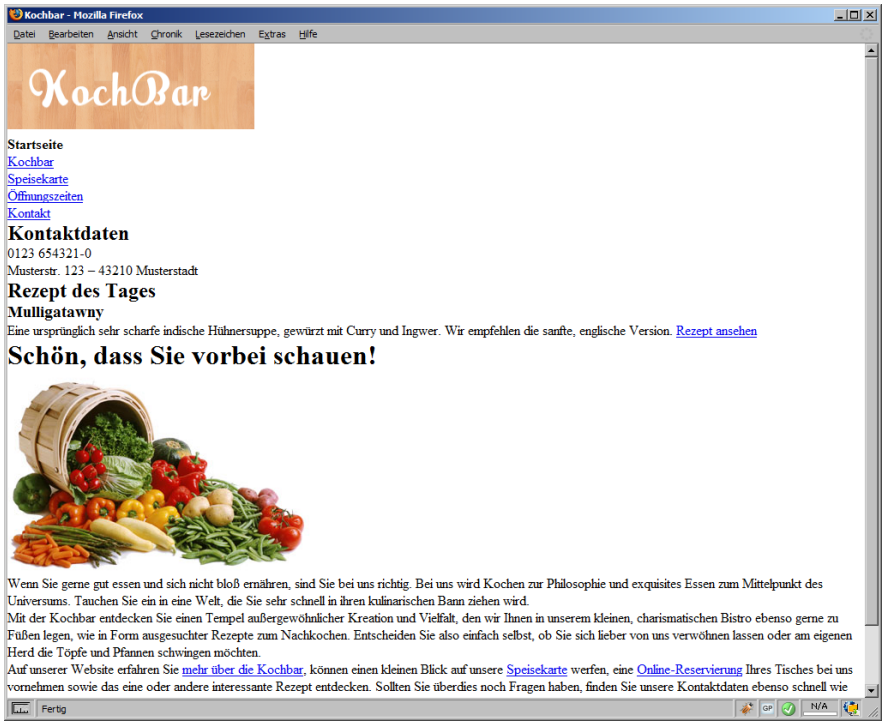


Abbildung 8.13: Ohne Außen- und Innenabstände wirkt die Seite wie zusammengestaucht

ÜBUNG

Schriftart und -größe deklarieren

1. *Ergänzen Sie Ihre* `default.css`.
2. *Speichern Sie und betrachten Sie die Kochbar im Browser.*
3. *Verändern Sie die Werte für Schriftgröße und Zeilenhöhe. Entwickeln Sie ein Gefühl für die Schriftskalierung. Ändern Sie auch mal die Schriftfamilie: Geben Sie beispielsweise Verdana an. Wirkt die Schrift dadurch größer oder kleiner?*

In den meisten Browsern ist Schwarz die Standardeinstellung für die Schriftfarbe und Weiß die Hintergrundfarbe. Hier ist die Schrift nicht ganz schwarz, sondern hat den Farbwert #333. Denken Sie daran: Wenn Sie die Schriftfarbe deklarieren, müssen Sie auch die Hintergrundfarbe explizit angeben, also:

```
/* Farben und Hintergründe setzen */
html, body {
  color: #3b2b20;
  background-color: white;
}
```

Listing 8.15: Globale Schrift- und Hintergrundfarben

ÜBUNG

Schrift- und Hintergrundfarbe

1. *Ergänzen Sie Ihre* `default.css`.
2. *Speichern Sie das Stylesheet und betrachten Sie die Kochbar im Browser.*
3. *Vergleichen Sie mit Abbildung 8.14.*

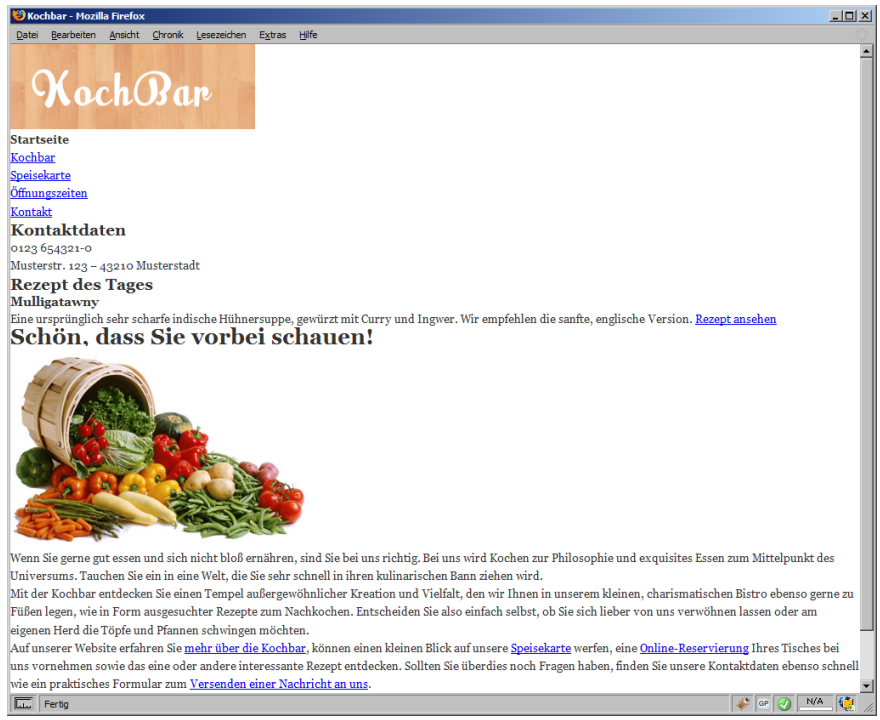


Abbildung 8.14: Schritt 6

Mit den globalen Einstellungen sind wir – zumindest für den Moment – fertig. Kommen wir zum nächsten Schritt.

8.9 Schritt 7: Regeln für allgemeine Elemente

In diesem Teil des Stylesheets legen wir die Standarddarstellung von Elementen auf unserer Website fest. Dazu müssen wir unterscheiden zwischen »normalen« Elementen und »besonderen« Elementen. Wenn wir beispielsweise die Elemente vom Typ `p` betrachten, sehen wir, dass das gar nicht so einfach ist. Es

gibt p-Elemente in der Sidebar, im Content und in der Fußzeile, und das Screendesign sieht vor, dass die Elemente an jeder Stelle anders aussehen. Was davon ist die Standarddarstellung?

Webautoren gehen üblicherweise so vor: Die Darstellung der Elemente im Content ist die Standarddarstellung. Das ist sinnvoll, wenn Sie sich noch einmal bewusst machen, dass der Content üblicherweise der größte (und wichtigste!) Bereich auf einer Webseite ist. Mit dem Content steht und fällt die ganze Seite. Ist er schlecht ausgezeichnet und mies gestaltet, wird die Website dauerhaft nur wenige Besucher anlocken.

Überschriften und Absätze

Beginnen wir mit den Überschriften und den Absätzen. Wir übernehmen Farben und Schriftgrößen aus dem Entwurf und schaffen ein wenig mehr Raum zwischen den einzelnen Elementen durch obere und untere Außenabstände.

```
/**
 * Regeln für allgemeine Elemente
 *
 * @section Überschriften
 */
h1 {
  color: #9F0F00;
  background-color: white;
  font-size: 3em;
  line-height: 1.1;
  margin: .8em 0;
}
h2 {
  color: #3C6331;
  background-color: white;
  font-size: 1.5em;
  line-height: 1.1;
  margin: .8em 0;
  text-transform: uppercase;
}

/**
 * @section Absätze
 */
p { margin: .8em 0; }
```

Listing 8.16: Deklarationen für Überschriften und Absätze

Ist Ihnen aufgefallen, dass wir sowohl den Margin-Bereich als auch die Schriftgrößen in em angegeben haben, also in einer relativen Einheit? Ich habe nachgemessen, wie groß die Überschriften in der Photoshop-Datei sind: Überschriften erster Ordnung 36 px, Überschriften zweiter Ordnung 18 px. Das entspricht 300 Prozent und 150 Prozent der Standardschriftgröße 12 px – oder eben 3 em und 1,5 em.

0,8 em ist ein bewährter Standardwert für vertikale Außenabstände, gerade bei Absätzen. Geben wir diesen Wert zunächst einfach an; wir können ihn später immer noch anpassen.

Links

Nun kümmern wir uns ein wenig um die Farben der Links in ihren jeweiligen Zuständen. Unbesuchte Links präsentieren sich in einem dunklen Rot, besuchte Links sind grau. Wird ein Link mit der Maus überfahren oder hat er den Fokus, leuchtet er orange auf. Bei aktiven Links tauschen wir die ursprüngliche Schriftfarbe in die Hintergrundfarbe: Wir deklarieren weiße Schrift auf rotem Hintergrund.

```
/**
 * @section Links und Anker
 */
a:link {
  color: #980c02;
  background-color: transparent;
}
a:visited {
  color: #999;
  background-color: transparent;
}
a:hover, a:focus {
  color: #FF7700;
  background-color: transparent;
}
a:active {
  color: white;
  background-color: #980c02;
}
```

Listing 8.17: Deklarationen für Hyperlinks

Damit sind wir für diesen Schritt fertig. In späteren Schritten werden wir die Regeln für allgemeine Elemente um Regeln für Listen, Tabellen und Formulare ergänzen.

Beginnen wir nun mit der Ausgestaltung der einzelnen Bereiche.

ÜBUNG

Regeln für allgemeine Elemente

1. *Ergänzen Sie Ihre* `default.css`.
2. *Speichern Sie das Stylesheet und betrachten Sie die Kochbar im Browser.*
3. *Vergleichen Sie mit Abbildung 8.15.*

Abbildung 8.15: Schritt 7

8.10 Schritt 8: Styling des Kopfbereichs

Wir starten oben beim Kopfbereich und fügen ein wenig Feinstrukturierung hinzu, indem wir das Logo und die Navigation in einem `div`-Element mit der ID `head` zusammenfassen. Das bietet uns die Möglichkeit, den Kopfbereich als Ganzes zu gestalten und über Kontextselektoren auf das Logo (`#head h1`) und die Navigation (`#head ul`) zuzugreifen.

```
<div id="head">
  <h1><img src="images/logo.png" alt="Kochbar"
    width="288" height="100" /></h1>

  <ul>
    <li><strong title="Hier befinden Sie sich
      gerade.">Startseite</strong></li>
    <li><a href="ueber-uns.html">Kochbar</a></li>
    <li><a href="speisekarte.html">Speisekarte</a></li>
    <li><a href="oeffnungszeiten.html">Öffnungszeiten</a></li>
    <li><a href="kontakt.html">Kontakt</a></li>
  </ul>
</div>
```

Listing 8.18: Feinstrukturierung: Logo und Navigation zusammenfassen

Sowohl die Holzvertäfelung im Kopfbereich als auch die aufgespießten Leckereien werden über die gesamte Fensterbreite gekachelt. Kacheln bedeutet eine Wiederholung desselben Bilds, entweder in Richtung der x-Achse, der y-Achse oder beider Achsen. Das ist in dem in Abbildung 8.1 dargestellten Ausschnitt zu erahnen, aber nicht eindeutig zu erkennen. Abbildung 8.16 zeigt einen verkleinerten Ausschnitt der Photoshop-Vorlage; darin können Sie die Kachelung gut sehen. Wir müssen nun herausfinden, an welchen Stellen sich die Holzvertäfelung und die Gabeln wiederholen und anschließend die Kacheln herausschneiden. Die Frage hierbei ist: Können wir die Vertäfelung und die Gabeln zu einer Kachel zusammenfassen, die sich in der x-Achse wiederholt?

Abbildung 8.16: Die Hintergrundbilder sind durchgängig von Rand zu Rand

Abbildung 8.17 und Abbildung 8.18 zeigen die beiden Kacheln. Wir können auf den ersten Blick erkennen, dass die Kacheln nicht zusammenpassen. Das Bild `vertaefelung.gif` ist 271 px breit, das Bild `gabeln.jpg` 398 px. Wir können Vertäfelung und Gabeln folglich nicht zu einer großen Kachel zusammenfassen. Das ist nicht weiter schlimm; wir kümmern uns zunächst nur um die Holzvertäfelung im Kopfbereich und bauen die Gabeln später ein.

Abbildung 8.17: vertaefelung.gif

Abbildung 8.18: gabeln.jpg

Die Holzvertäfelung weisen wir dem `body`-Element als Hintergrundbild zu.

```
/* Farben und Hintergründe setzen */
html, body {
  color: #3b2b20;
  background-color: white;
}
body {
  background: url(images/vertaefelung.gif) top center repeat-x;
}
```

Listing 8.19: Die Vertäfelung fügen wir als Hintergrundbild des `body`-Elements ein

Für `h1`-Elemente haben wir `margin-bottom` auf `.8em` gesetzt. Diese Einstellung müssen wir für das `h1`-Element innerhalb von `#head` wieder überschreiben, damit der Kopfbereich exakt so hoch ist wie das Logo. Zudem müssen wir die Breite des `h1`-Elements auf die Breite des Logos beschränken, damit es sich nicht über die volle zur Verfügung stehende Breite erstreckt und der standardmäßig gesetzte weiße Hintergrund die gekachelte Vertäfelung überdeckt. Wir setzen die Breite auf 288 px. Eine andere Möglichkeit wäre die Deklaration von `display: inline;`.

```
/**
 * Regeln für besondere Bereiche
 *
```

```
 * @section Kopfbereich
 */
div#head {}

#head h1 {
  margin: 0;
  padding: 0;
  width: 288px;
}
```

Listing 8.20: Die Überschrift erster Ordnung auf die Dimensionen des Logos beschränken

Hier sehen Sie übrigens eine Angewohnheit von mir, die ich Ihnen ebenfalls nahelegen möchte. In jedem Dokument der Kochbar-Website gibt es nur ein Element mit der ID `head`, nämlich das `div`-Element, das den Kopfbereich auszeichnet. Wir brauchen also nicht `div#head` zu schreiben, sondern können auf `#head` abkürzen. Dennoch schreibe ich den Selektor ein Mal vollständig hin, auch wenn – wie in diesem Fall – es für das Element selbst noch überhaupt keine Deklarationen gibt. Damit weiß der Leser des Stylesheets (beispielsweise Sie!), dass es sich bei `#head` um ein `div`-Element handelt; das kann helfen, die eine oder andere Deklaration besser zu verstehen.

ÜBUNG

Vertäfelung einbauen

1. *Laden Sie sich das vollständige Kochbar-Beispiel von der Seite http://jendryschik.de/wsdev/einfuehrung/buch/ herunter. Kopieren Sie die Datei* `vertaefelung.gif` *in Ihren* `images`*-Ordner.*
2. *Ergänzen Sie Ihre* `index.html`*.*
3. *Ergänzen Sie Ihre* `default.css` *um die* `background`*-Deklaration für* `body`*.*
4. *Speichern Sie und betrachten Sie die Kochbar im Browser. Sie werden keinen Unterschied feststellen, da das* `h1`*-Element mit dem Logo den Hintergrundverlauf komplett überdeckt.*
5. *Fügen Sie in Ihre* `default.css` *nun das CSS aus Listing 8.20 ein.*
6. *Speichern Sie das Stylesheet und betrachten Sie die Kochbar im Browser. Nun wird die Vertäfelung über die gesamte Breite gekachelt. Wir haben Glück, dass sich das Muster des Logos exakt in den Hintergrund einfügt und es nicht zu Verschiebungen kommt. Da scheint sich jemand etwas dabei gedacht zu haben.*[6]
7. *Vergleichen Sie mit Abbildung 8.19.*

6 An dieser Stelle noch einmal herzlichen Dank an die Screendesignerin!

Abbildung 8.19: Vertäfelung als Hintergrundbild des `body`-Elements

Der Entwurf gibt vor, dass alle Inhalte der Kochbar auf eine feste Breite beschränkt und zudem horizontal zentriert sind. In der Photoshop-Vorlage können wir die Breite abschätzen; wir gehen im Folgenden von 932 px aus. Wir setzen die Breite des Kopfbereichs auf diesen Wert und zentrieren den Block mittels automatischer Außenabstände (siehe Kapitel 7.8.1).

```
/**
 * Regeln für besondere Bereiche
 *
 * @section Kopfbereich
 */
div#head {
  width: 932px;
  margin: auto;
}

#head h1 {
  margin: 0;
  padding: 0;
  width: 288px;
}
```

Listing 8.21: Breite des Kopfbereichs beschränken und Kopfbereich zentrieren

Nun kümmern wir uns um die Navigation. Sie liegt bündig auf der Unterkante des Kopfbereichs auf, 30 px vom rechten Rand entfernt.

Wie bekommen wir die Navigation an diese Stelle? Wissen Sie die Antwort? Falls nicht, rufen Sie sich bitte Kapitel 7.7 »Positionierung« ins Gedächtnis.

Wir positionieren #head relativ, damit es zum Positionierungskontext für die Navigation wird. Anschließend positionieren wir die Navigationsliste absolut an die ermittelte Stelle.

```
/**
 * Regeln für besondere Bereiche
 *
 * @section Kopfbereich
 */
div#head {
  width: 932px;
  margin: auto;
  position: relative;
}

#head h1 {
  margin: 0;
  padding: 0;
  width: 288px;
}

/**
 * @section Hauptmenü
 */
#head ul {
  position: absolute;
  bottom: 0;
  right: 30px;
}
```

Listing 8.22: Positionierung der Navigation

Um die Navigationspunkte nicht vertikal, sondern horizontal auszurichten, lassen wir sie nach links floatieren. Wir verzichten auf eine Listenelementmarkierung und definieren einen Abstand von 8 px zwischen den Listenpunkten.

```
#head ul li {
  float: left;
  list-style: none;
  margin: 0 8px 0 0;
}
```

Listing 8.23: Horizontale Ausrichtung der Listenpunkte

Als Nächstes kümmern wir uns um die Darstellung der Listenpunkte. Wir deklarieren Schrift- und Hintergrundfarben sowie die Hintergrundverläufe. Der aktive Navigationspunkt, mit strong ausgezeichnet, sieht so aus wie Navigationslinks, die mit der Maus überfahren werden, den Fokus haben oder aktiv sind.

```
#head ul a,
#head ul strong {
  color: #352545;
  background: #FBFBFB url(images/nav_bg.gif)
    bottom left repeat-x;
  display: block;
  padding: .8em;
  text-transform: uppercase;
  text-decoration: none;
  font-weight: bold;
}
#head ul a:hover,
#head ul a:focus,
#head ul a:active,
#head ul strong {
  color: white;
  background: #638F56 url(images/nav_bg_hover.gif)
    bottom left repeat-x;
}
```

Listing 8.24: Styling aktiver und inaktiver Listenpunkte

Abbildung 8.20: nav_bg.gif

Abbildung 8.21: nav_bg_hover.gif

ÜBUNG

Navigation

1. *Kopieren Sie die Dateien* `nav_bg.gif` *und* `nav_bg_hover.gif` *in Ihren* `images`*-Ordner.*
2. *Ergänzen Sie Ihre* `default.css` *um die Regeln für die Navigation.*
3. *Speichern Sie und betrachten Sie die Kochbar im Browser. Vergleichen Sie mit Abbildung 8.22.*

8.11 Schritt 9: Illustrationen

In diesem Schritt werden Sie sehen, dass die Umsetzung eines Screendesigns manchmal einfacher sein kann, als es auf den ersten Blick aussieht. Wir werden uns um die Kachelung der Gabeln kümmern, den geschwungenen Übergang und den oberen Teil der Sidebar mit dem aufgespießten Gemüse, das die obere rechte Ecke des Kastens überlappt.

Abbildung 8.22: Schritt 8

Haben Sie schon eine Vorstellung davon, wie Sie vorgehen würden, um all dies umzusetzen?

Wir fangen erneut an mit einer Erweiterung des Markups. Dieses Mal müssen wir sogar zwei Elemente ineinander verschachteln.

```
<body>
  <div id="head">
    <!-- Logo und Navigation -->
  </div>

  <div id="main">
    <div id="main-inner">
      <div id="sidebar">
        <!-- Sidebar -->
      </div>

      <div id="content">
        <!-- Inhalte -->
      </div>
    <div>
  </div>

  <div id="footer">
    <!-- Absätze -->
  </div>
</body>
```

Listing 8.25: Erweiterung des Markups um zwei Styling-Container

In das Stylesheet schreiben wir:

```
/**
 * @section Hauptbereich
 */
div#main {
  color: inherit;
  background: white url(images/gabeln.jpg) top center repeat-x;
}
#main div#main-inner {
  width: 932px;
  margin: 0 auto;
}
```

Listing 8.26: Hinzufügen der Gabeln, Beschränkung der Breite des Inhalts und horizontale Zentrierung

Die erste Regel sorgt für die Kachelung der Gabeln über die gesamte Breite des `#main`-Elements. Das darin verschachtelte `div`-Element `#main-inner` sorgt analog zum `#head`-Element für die Breitenbeschränkung und horizontale Zentrierung.

Jetzt kommt der Trick: Wir machen aus dem gesamten oberen Teil des inneren Bereichs ein einziges großes Bild (siehe Abbildung 8.23) und weisen es `#main-inner` als Hintergrundbild zu! Wir setzen `padding-top` mit `348px` auf die Höhe des Bilds; alle weiteren Elemente beginnen nun unterhalb der Illustration.

```
/**
 * @section Hauptbereich
 */
div#main {
  color: inherit;
  background: white url(images/gabeln.jpg) top center repeat-x;
}
#main div#main-inner {
  color: inherit;
  background: transparent url(images/illustration.jpg)
    top center no-repeat;
  width: 932px;
  margin: 0 auto;
  padding: 348px 0 0 0;
}
```

Listing 8.27: Hinzufügen aller Illustrationen als Hintergrundbild des inneren Styling-Containers

Das war einfach, oder? Jetzt müssen wir die Sidebar so gestalten, dass sie nahtlos an die Illustration anschließt. Das machen wir im nächsten Schritt.

Abbildung 8.23: Die grafischen Effekte des inneren Bereichs schneiden wir komplett als ein Bild aus dem Entwurf heraus

ÜBUNG

Illustrationen

1. *Kopieren Sie sich die Dateien* `gabeln.jpg` *und* `illustration.jpg` *in Ihren* `images`-*Ordner.*
2. *Ergänzen Sie Ihre* `index.html` *um die beiden* `div`-*Elemente.*
3. *Ergänzen Sie Ihre* `default.css` *um die Regeln für* `#main` *und* `#main-inner`*.*
4. *Speichern Sie und betrachten Sie die Kochbar im Browser. Vergleichen Sie das Ergebnis mit Abbildung 8.24.*

8.12 Schritt 10: Sidebar

Sidebar nennen wir den Bereich links, der das Kontaktfeld und den Teaser auf das Rezept des Tages beinhaltet. Den oberen Teil der Sidebar haben wir durch die großformatige Illustration simuliert – kümmern wir uns nun um den Rest. Dazu überlegen wir uns zunächst, was genau eigentlich zu tun ist.

- Wir müssen die Sidebar nach links floatieren, damit der Content rechts um die Sidebar fließt. Um den Content kümmern wir uns im nächsten Schritt.
- Wir müssen die Breite auf 288 px beschränken.

Abbildung 8.24: Schritt 9

- Wir müssen den grauen Rahmen weiterführen. Der Rahmen ist 6 px dick und hat den Farbwert `#CBB8BA`. Beachten Sie die Berechnung der Breite eines Elements: Die Gesamtbreite setzt sich zusammen aus `margin-left`, `border-left`, `padding-left`, `width`, `padding-right`, `border-right` und `margin-right`. Der beidseitige Rahmen von 6 px reduziert den Wert der Eigenschaft `width` somit auf 276 px (288 px – 6 px – 6 px = 276 px).
- Im Hintergrund der Sidebar liegt ein dezenter Farbverlauf (siehe Abbildung 8.25). Das ist nicht auf den ersten Blick zu erkennen und im Screenshot auf Seite 432 vermutlich überhaupt nicht, aber auf solche Feinheiten müssen aufmerksame Webautoren natürlich achten.
- Wir können im Entwurf erkennen, dass die Sidebar nicht direkt auf der Fußzeile aufliegt, sondern 26 px Raum dazwischen besteht. Darüber hinaus sehen wir, dass das Rezept des Tages 24 px Abstand zum Rahmen der Sidebar hat. Die entsprechenden Angaben müssen wir für die Eigenschaften `margin` und `padding` vornehmen.

Abbildung 8.25: sidebar_bg.gif

Wir erweitern das Stylesheet also wie folgt:

```
/**
 * @section Sidebar
 */
div#sidebar {
  color: inherit;
  background: #F5F5F9 url(images/sidebar_bg.gif)
    bottom left repeat-x;
  float: left;
  border: 6px solid #cbb8ba;
  border-top: 0;
  width: 276px;
  margin: 0 0 26px 0;
  padding: 0 0 24px 0;
}
```

Listing 8.28: Styling der Sidebar

Nun sieht die Sidebar so aus wie im Entwurf. Der Rahmen schließt perfekt an die Illustration an, es gibt keinen sichtbaren Übergang oder Versatz – so soll es sein. Sind Sie mit dem Ergebnis zufrieden?

Ich war es so lange, bis ich mir die Seite im Firefox angeschaut und den Zoomfaktor um eine Stufe erhöht habe. Machen Sie es mir nach: Auf einmal ist ein Versatz zu sehen. Eine Zoomstufe weiter können Sie sogar eine Lücke sehen! Auch bei Verkleinerung passen Rahmen und Illustration nicht mehr zusammen. Wir müssen die Sidebar also umbauen. So ist das manchmal: Überraschungen dieser Art machen unseren Beruf so abwechslungsreich!

Abbildung 8.26: Rahmen und Illustration passen bei anderen Zoomstufen nicht mehr zusammen

Unsere Lösung besteht darin, die vertikalen Rahmen ebenfalls als Bild zu realisieren in der Hoffnung, dass dieses und die Illustration identisch vergrößert und verkleinert werden, wodurch es weder zu Versatz noch zu Lücken kommen sollte. Wir benötigen also eine Grafik (siehe Abbildung 8.27), die wir als Hintergrundbild der Sidebar einbinden. Der Farbverlauf wird zum Hintergrundbild eines neuen Containers, den wir in `#sidebar` verschachteln. Wir geben diesem Container die ID `sidebar-inner`.

Abbildung 8.27: sidebar_rahmen.gif

```
<div id="sidebar">
  <div id="sidebar-inner">
    <div id="contact">
      <h2>Kontaktdaten</h2>

      <p class="tel">0123 654321-0</p>
     <p class="address">Musterstr. 123 - 43210 Musterstadt</p>
    </div>

    <div id="teaser">
      <h2>Rezept des Tages</h2>
```

```
        <h3>Mulligatawny</h3>

        <p>Eine ursprünglich sehr scharfe indische Hühnersuppe,
          gewürzt mit Curry und Ingwer. Wir empfehlen die sanfte,
          englische Version.
          <a href="rezepte/mulligatawny.html">Rezept
          ansehen</a></p>
      </div>
    </div>
  </div>
```

Listing 8.29: Hinzufügen eines zusätzlichen Containers für ein weiteres Hintergrundbild

Nun weisen wir beide Hintergrundbilder zu und verteilen die Deklaration etwas um. Beachten Sie, dass `#sidebar` horizontale Padding-Bereiche in der Breite des »virtuellen« Rahmens benötigt, damit dieser vom Inhalt der Sidebar nicht überdeckt wird.

```
/**
 * @section Sidebar
 */
div#sidebar {
  color: inherit;
  background: #F5F5F9 url(images/sidebar_rahmen.gif)
    top left repeat-y;
  float: left;
  border-bottom: 6px solid #CBB8BA;
  width: 276px;
  padding: 0 6px;
  margin: 0 0 26px 0;
}
div#sidebar-inner {
  color: inherit;
  background: #F5F5F9 url(images/sidebar_bg.gif)
    bottom left repeat-x;
  padding: 0 0 24px 0;
}
```

Listing 8.30: Korrigiertes Styling der Sidebar

Die Nahtstelle zwischen Illustration und Sidebar sieht nun in allen Zoomstufen besser aus. Ich denke, mit dieser Lösung können wir weiterarbeiten.

Kommen wir nun zum Kontaktfeld. Zunächst fügen wir der `index.html` einige Klassen hinzu, um die beiden Absätze, mit denen wir die Telefonnummer und die Adresse ausgezeichnet haben, getrennt ansprechen zu können.

```
<div id="contact">
  <h2>Kontaktdaten</h2>

  <p class="tel">0123 654321-0</p>
  <p class="address">Musterstr. 123 - 43210 Musterstadt</p>
</div>
```

Listing 8.31: Die beiden Absätze lassen sich am einfachsten über verschiedenen Klassen unterscheiden

Das CSS ist sehr einfach. Wir bestimmen Schrift- und Hintergrundfarben und nehmen einige Schriftdeklarationen vor. Zudem korrigieren wir einige innere und äußere Abstände.

```
/* Kontaktfeld */
div#contact {
  color: white;
  background-color: #E8AD47;
  padding: 0 24px 16px 24px;
}
#contact h2 {
  color: inherit;
  background-color: transparent;
  margin: 0;
}
#contact p.tel {
  color: #9F0F00;
  background-color: transparent;
  font-size: 2.2em;
  font-weight: bold;
  margin: .3em 0;
}
#contact p.address {
  font-weight: bold;
  font-size: .9em;
}
```

Listing 8.32: Styling des Kontaktfelds

Das Styling des Teasers ist nicht ganz so einfach. Deswegen gehen wir Schritt für Schritt vor. Wir kümmern uns zunächst um die richtigen Abstände des Teasers zum Rand der Kochbar sowie um die notwendige Polsterung. Zudem fügen wir die blauen Linien als Hintergrundbild hinzu.

```
/* Teaser */
div#teaser {
  color: #333;
  background: #fbfbfd url(images/teaser_bg.gif) top left;
  margin: 12px 12px 0 12px;
  padding: 12px;
}
```

Abbildung 8.28: Styling des Teasers, Abstände und Hintergrundbild

Nun gestalten wir die Überschrift, ändern Farben, Schriftformatierung und Abstände und binden den orangefarbenen Strich für die Unterstreichung als Hintergrundbild ein.

```
#teaser h2 {
  color: #FF7700;
  background: transparent url(images/teaser-h2_bg.gif) bottom
    center no-repeat;
  font-size: 2.2em;
  text-align: center;
  font-weight: normal;
  padding: 0 0 12px 0;
  margin: 0;
  text-transform: none;
}
```

Abbildung 8.29: Styling des Teasers, Überschrift zweiter Ordnung

Nun sehen wir, dass der Abstand zwischen den Überschriften zu klein ist, außerdem liegt »Mulligatawny« nicht richtig auf der blauen Linie. Wir richten das `h3`-Element entsprechend aus.

```
#teaser h3 {
  margin: 12px 0 19px 0;
  padding: 0;
}
```

Abbildung 8.30: Styling des Teasers, Ausrichtung der Überschrift dritter Ordnung

Anschließend müssen wir die inneren und äußeren Abstände des Absatzes, seine Schriftgröße und die Zeilenhöhe so ausrichten, dass die einzelnen Zeilen auf den blauen Linien des Hintergrundbilds ausgerichtet zu sein scheinen. Mit folgenden Deklarationen bekommen wir dies hin.

```
#teaser p {
  margin: 14px 0 0 0;
  padding: 0;
  font-size: 11px;
  font-weight: bold;
  line-height: 18px;
}
```

Listing 8.33: Styling des Teasers, Absätze

Abbildung 8.31: Schritt 10

ÜBUNG

Sidebar

1. *Kopieren Sie die Dateien* `sidebar_bg.gif`, `sidebar-rahmen.gif`, `teaser-bg.gif` *und* `teaser-h2_bg.gif` *in Ihren* `images`*-Ordner.*
2. *Ergänzen Sie Ihre* `index.html` *um den Container* `#sidebar-inner`.
3. *Ergänzen Sie Ihre* `default.css` *um die Regeln für die Sidebar, das Kontaktfeld und den Teaser.*
4. *Speichern Sie und betrachten Sie die Kochbar im Browser. Vergleichen Sie mit Abbildung 8.31.*

Die Sidebar haben wir formatiert, nun folgt der Content.

8.13 Schritt 11: Content

Es ist verhältnismäßig einfach, den Content an die richtige Position zu bekommen und zu verhindern, dass die Inhalte unter der Sidebar umbrechen. Wir gehen in zwei Schritten vor:

- Zunächst setzen wir den rechten Margin-Bereich so, dass der Content genau an die Sidebar anschließt. Für alle anderen Seiten setzen wir `margin` auf `0`.
- Anschließend deklarieren wir die inneren Abstände über die Eigenschaft `padding`. Dabei brauchen wir uns nur um die horizontalen Abstände zu kümmern; die vertikalen Abstände sind `0`.

```
/**
 * @section Content
 */
div#content {
  margin: 0 0 0 288px;
  padding: 0 12px 0 36px;
}
```

Listing 8.34: Styling des Contents

Natürlich hätten wir einfach `margin: 0 12px 0 324px;` schreiben können, jedoch finde ich es sauberer, zunächst ein lückenloses Spaltenlayout zu formulieren und zusätzliche Abstände über weitere Deklarationen zu realisieren.

Nun fehlen noch zwei Formatierungen für die Inhalte: Den oberen Margin-Bereich der Hauptüberschrift setzen wir auf `0`, damit sie direkt an der oberen Kante des Contents beginnt. Und wir sorgen dafür, dass das Gemüse floatiert. Es ist das einzige `img`-Element im Content (und zwar derzeit auf allen Seiten), daher können wir ganz unkompliziert vorgehen und einfach alle Grafiken innerhalb von `#content` floatieren.

```
#content h1 {
  margin-top: 0;
}
#content img {
  float: left;
  margin: .5em .5em .5em 0;
}
```

Listing 8.35: Nur wenige Content-Elemente benötigen spezielle Formatierungen

ÜBUNG

Content

1. *Ergänzen Sie Ihre* `default.css`.
2. *Speichern Sie und betrachten Sie die Kochbar im Browser. Vergleichen Sie das Ergebnis mit Abbildung 8.32.*

Abbildung 8.32: Schritt 11

Den Abschluss jeder Seite bildet die Fußzeile, um die wir uns als Nächstes kümmern.

8.14 Schritt 12: Fußzeile

Kommen wir zur Fußzeile; zum Ende der Seite und gleichzeitig auch zum vorläufigen Abschluss des Stylesheets. Die Fußzeile besteht nur aus einem `div`-Element mit der ID `footer` und darin zwei Absätzen, daher ist ihr Styling nicht sonderlich schwierig.

Wichtig ist allerdings, dass wir Folgendes nicht vergessen: Die Fußzeile steht sowohl unterhalb des Contents als auch unterhalb der floatierten Sidebar! Wir müssen daran denken, die Eigenschaft `clear` auf den Wert `left` zu setzen – oder besser gleich auf `both`. Darüber hinaus müssen wir beachten, dass die Fußzeile eine andere Schriftart hat als der Rest der Seite, nämlich die serifenlose Schriftart Tahoma.

Für die Breitenbeschränkung und die horizontale Zentrierung setzen wir wieder die Eigenschaften `width` und `margin` ein. Wir benötigen dieses Mal keinen inneren Container, sondern deklarieren direkt für die Absätze innerhalb des Footers. Sollten wir die Fußzeile später einmal durch zusätzliche Elemente erweitern wollen, können wir an dieser Stelle immer noch Änderungen vornehmen.

```
/**
 * @section Footer
 */
div#footer {
  color: #7F6063;
  background-color: #F3F2FA;
  padding: 24px 0;
  font-family: Tahoma, Arial, sans-serif;
  font-size: .9em;
  clear: both;
  margin: 24px 0 0 0;
}
#footer p {
  width: 932px;
  margin: 0 auto;
}
```

Listing 8.36: Styling der Fußzeile

ÜBUNG

Content

1. *Ergänzen Sie Ihre* `default.css`.
2. *Speichern Sie, betrachten Sie die Kochbar im Browser und vergleichen Sie sie mit Abbildung 8.33.*
3. *Validieren Sie sowohl Ihr Markup als auch das Stylesheet. Wenn Sie beim Aufbau der Dokumente keinen Fehler gemacht haben, validieren sowohl die* `index.html` *als auch die* `default.css` *vollständig ohne Fehler oder Warnungen.*

Wir sind fertig – die Homepage ist vollständig. Und das war doch gar nicht so schwierig, oder?

Lassen Sie uns mit den Unterseiten weitermachen.

Abbildung 8.33: Schritt 12

8.15 Schritt 13: Über die Kochbar

Wir haben das Glück, dass sich die Unterseiten nur geringfügig von der Startseite unterscheiden. Die Unterseite sieht nahezu identisch aus: Es ändern sich nur die Inhalte des Hauptinhaltsbereichs `#content` und Details an der Navigation.

Wir erstellen eine Kopie der `index.html` und nennen sie `ueber-uns.html` – so wie wir es bei der Auszeichnung der Navigation vorgesehen haben; anschließend nehmen wir alle Änderungen vor und fügen den neuen Content ein. Folgendes ist zu tun:

- Wir müssen den Titel anpassen. Verschiedene Seiten brauchen verschiedene Titel. Heute ist es üblich, den Seitentitel so zu gestalten, dass er eine Art umgekehrten Ariadne-Faden ergibt, beginnend bei der aktuellen Seite. Als Trennzeichen haben sich der Trennstrich (`-`) und der senkrechte Strich (`|`) bewährt, wobei ich den Trennstrich vorziehe. Unser Titel für die »Über uns«-Seite lautet somit: `Über uns - Kochbar`.
- Ebenso wie den Titel müssen wir auch die Beschreibung ändern und an die aktuelle Seite anpassen. In Ermangelung eines geeigneten Texts lassen wir das entsprechende `meta`-Element einfach weg – das ist nicht weiter schlimm, Suchmaschinen werden die Seite auch so finden.
- Es ist üblich, das Logo auf Unterseiten mit der Startseite zu verlinken. Wohlgemerkt, auf Unterseiten! Die Regel, dass eine Seite niemals auf sich selbst verlinken sollte, gilt auch für dieses Detail. Deswegen haben wir das Logo auf der Homepage unverlinkt ausgezeichnet, müssen die Verlinkung jedoch jetzt nachholen.
- Der Content besteht nur aus der Hauptüberschrift und drei Absätzen. Das können wir mittlerweile mit einer gewissen Routine auszeichnen.

Der Quelltext der Website sieht so aus:

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

<html xmlns="http://www.w3.org/1999/xhtml"
  xml:lang="de" lang="de">
  <head>
    <meta http-equiv="Content-Type"
      content="text/html; charset=utf-8" />

    <title>Über uns - Kochbar</title>

    <meta name="author-personal"
      content="Michael Jendryschik" />
    <meta name="author-mail"
      content="mailto:michael@jendryschik.de" />

    <meta name="language" content="de" />
    <meta name="robots" content="index, follow" />

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

```
<body>
  <div id="head">
    <h1><a href="index.html"><img src="images/logo.png"
      alt="Kochbar" width="288" height="100" /></a></h1>

    <ul>
       <li><a href="index.html">Startseite</a></li>
       <li><strong title="Hier befinden Sie sich
         gerade.">Kochbar</strong></li>
      <li><a href="speisekarte.html">Speisekarte</a></li>
      <li><a href="oeffnungszeiten.html">
        Öffnungszeiten</a></li>
      <li><a href="kontakt.html">Kontakt</a></li>
    </ul>
  </div>

  <div id="main">
    <div id="main-inner">
      <div id="sidebar">
        <!-- Sidebar -->
      </div>

      <div id="content">
         <h1>In die Töpfe geschaut…</h1>

         <p>Die Philosophie der Kochbar ist denkbar einfach: An
           unserem Essen sollt Ihr uns messen! Keine großen
           Inszenierungen, kein unnötiger Schnickschnack: Was
           bei uns auf den Teller kommt, lebt von seiner
           außergewöhnlichen Qualität sowie einer perfekten
           Zubereitung.</p>

         <p>Damit das so ist, verwenden wir einen großen Teil
           unserer Zeit auf die sorgfältige Auswahl erlesener
           Produkte. Wir orientieren uns dabei am Rhythmus der
           Jahreszeiten, denn schließlich soll alles stets ganz
           frisch auf den Teller kommen. Was dem Essen dann
           noch fehlt, schaffen ein wenig Phantasie bei der
           Zubereitung sowie eine kräftige Portion Mut bei der
           Zusammenstellung der Zutaten.</p>

         <p>Unser Chefkoch Jim Stelzer ist Absolvent der
           Pariser Elitekochschule »Le Cordon Bleu« und hat
           bereits in vielen renommierten Restaurants als »Sous
           Chef« und »Chef de Cuisine« wirken können. Mit der
           Kochbar, die er gemeinsam mit seinem Jugendfreund
           Jonathan Quayle Higgins führt, hat er sich einen
           persönlichen Kindheitstraum erfüllt.</p>
      </div>
```

```
        </div>

        <div id="footer">
          <!-- Fußzeile (unverändert) -->
        </div>
      </div>
    </body>
  </html>
```

Listing 8.37: ueber-uns.html

ÜBUNG

Über die Kochbar, Markup

1. *Erstellen Sie ein neues leeres Dokument und speichern Sie es in Ihrem Ordner als »ueber-uns.html«. Oder speichern Sie eine Kopie der* `index.html` *unter diesem Namen. Denken Sie beim Speichern daran, UTF-8 als Kodierung einzustellen.*
2. *Fügen Sie den Quelltext aus Listing 8.37 ein.*
3. *Speichern Sie und betrachten Sie das Ergebnis im Browser. Es sollte so aussehen wie das in Abbildung 8.34.*
4. *Funktionieren die Links in der Navigation und das verlinkte Logo? Können Sie von der »Über uns«-Seite zur Startseite navigieren und zurück?*
5. *Validieren Sie das Dokument. Ist es fehlerfrei?*

Sicherlich sehen Sie den Rahmen um das Logo und die Lücke zwischen Kopfbereich und Illustration. Die Lücke entsteht dadurch, dass die Dicke des Rahmens zur Höhe des Bilds addiert wird und das Logo dadurch insgesamt höher ist als die 100 px, auf die wir das Hintergrundbild zurechtgeschnitten haben.

Verlinkte Bilder erhalten standardmäßig einen Rahmen, der einem Nutzer signalisieren soll, dass er auf das Bild klicken kann. Dieser Rahmen ist allerdings ziemlich hässlich und häufig überflüssig, da aus anderen Gründen klar ist, dass das Bild verlinkt ist. Nutzer erwarten, auf ein Logo klicken zu können, wir brauchen es also nicht zusätzlich deutlich zu machen. Mittels `border: 0;` können wir den Rahmen entfernen.

```
/**
 * @section Bilder
 */

/* Rahmen verlinkter Bilder entfernen */
a img { border: 0; }
```

Listing 8.38: Unterdrücken des standardmäßig dargestellten Rahmens bei verlinkten Grafiken

Abbildung 8.34: Über uns

ÜBUNG

Über die Kochbar, Styling

1. *Ergänzen Sie Ihre* `default.css`.
2. *Speichern Sie und betrachten Sie das Ergebnis im Browser. Der Rahmen um das Logo und somit auch die Lücke sollten jetzt weg sein.*

8.16 Schritt 14: Impressum

Als Nächstes erstellen wir das Impressum. Der einfachste Weg ist, die Datei `ueber-uns.html` zu kopieren und die Kopie als `impressum.html` im selben Ordner abzuspeichern. Öffnen Sie sie dann in Ihrem Editor, um sie zu bearbeiten. Wir passen den Dokumenttitel an und ändern die Navigation.

Der Content besteht aus der Hauptüberschrift, zwei Absätzen und einem `address`-Element:

```
<h1>Impressum</h1>

<p>Inhaltlich Verantwortlicher gemäß § 10 Absatz 3 <abbr
title="Medien-Dienste-Staatsvertrag">MDStV</abbr>:</p>

<address>
  Michael Jendryschik<br />
  Kolmarer Straße 1<br />
  44137 Dortmund<br />
  E-Mail: <a href="mailto:michael@jendryschik.de">
    michael@jendryschik.de</a>
</address>

<p>Dies ist eine Beispiel-Website zur <a href="http://
jendryschik.de/wsdev/einfuehrung/">Einführung in XHTML, CSS und
Webdesign</a>. Weitere Informationen finden Sie in der <a
href="http://jendryschik.de/wsdev/einfuehrung/
einleitung/kochbar">Einleitung zur Kochbar</a>.</p>
```

Listing 8.39: Der Inhalt des Impressums

Das Stylesheet brauchen wir nicht zu ergänzen; alle Elemente sehen bereits so aus, wie sie aussehen sollen.

ÜBUNG

Impressum

1. *Erstellen Sie ein neues leeres Dokument und speichern Sie es in Ihrem Ordner als »impressum.html«. Oder speichern Sie eine Kopie der* `ueber-uns.html` *unter diesem Namen. Denken Sie beim Speichern daran, UTF-8 als Kodierung einzustellen.*
2. *Fügen Sie den Quelltext ein.*
3. *Speichern Sie und betrachten Sie das Impressum im Browser.*

8.17 Schritt 15: Speisekarte

Die Speisekarte ist die nächste Unterseite, die wir anlegen werden.

ÜBUNG

Speisekarte vorbereiten

Bitte bereiten Sie das Dokument `speisekarte.html` *so vor, dass wir uns um die Auszeichnung des Contents kümmern können. Erstellen Sie das Dokument auf Basis der* `ueber-uns.html`*, passen Sie Titel sowie Navigation an und leeren Sie das* `#content`*-Element.*

Beim Blick auf den Entwurf (siehe Abbildung 8.2) drängt sich keine bestimmte Art der Auszeichnung auf. Was ist auf dem Screenshot zu sehen? Überschriften, Absätze und Listen? Aber welche Art von Listen? Vielleicht Definitionslisten mit dem Gericht und dessen Beschreibung als Definitionsterm und dem Preis als Definitionsbeschreibung? Eher nicht. Oder vielleicht nur mit dem Gericht als Definitionsterm? Dann müssten wir die Beschreibung und den Preis im `dd`-Element unterbringen, aber wie zeichnen wir diese dann aus? Als Tabelle?

Bleiben wir einmal bei der Tabelle. Können wir die gesamte Speisekarte nicht vielleicht mittels einiger Tabellen auszeichnen? Die Gerichte in der linken Spalte und deren Preise in der rechten? Probieren wir es aus:

```
<table>
  <tr>
    <td>Knoblauchbrot<br />
      frisch aus dem Ofen</td>
    <td>1,80 €</td>
  </tr>
  <tr>
    <td>Hausgemachte Tomaten-Cremesuppe<br />
      mit Knoblauchcroutons</td>
    <td>2,80 €</td>
  </tr>
  <!-- weitere Gerichte -->
</table>
```

Das könnte funktionieren. Auch die Überschriften bekommen wir bequem unter:

```
<table>
  <caption>Vorspeisen und Suppen</caption>
  <tr>
    <td>Knoblauchbrot<br />
      frisch aus dem Ofen</td>
    <td>1,80 €</td>
  </tr>
  <tr>
    <td>Hausgemachte Tomaten-Cremesuppe<br />
      mit Knoblauchcroutons</td>
    <td>2,80 €</td>
  </tr>
  <!-- weitere Gerichte -->
</table>
```

Fehlen noch zwei Kleinigkeiten:

- Im Entwurf ist der Name der Gerichte größer und fetter als gewöhnlicher Fließtext und zudem in einer anderen Farbe dargestellt. Ich denke, hinter dieser Betonung steckt auch eine semantische Bedeutung. Wir sollten die Namen der Gerichte mit `em` oder `strong` auszeichnen. Ich verwende in solchen Fällen meistens `strong` (stark betont), obwohl `em` (betont) vielleicht die bessere Alternative wäre.
- Wir sollten für alle Tabellen der Speisekarte eine gemeinsame Klasse vergeben, um sie von anderen Tabellen unterscheiden und gezielt formatieren zu können. Aus demselben Grund sollten wir die Preiszellen mit einer Klasse versehen. Die Preise sehen übrigens genau so aus wie die Namen der Gerichte, allerdings gibt es keinen semantischen Grund, sie ebenfalls mit `em` oder `strong` auszuzeichnen.

```
<table class="speisekarte">
  <caption>Vorspeisen und Suppen</caption>
  <tr>
    <td><strong>Knoblauchbrot</strong><br />
      frisch aus dem Ofen</td>
    <td class="price">1,80 €</td>
  </tr>
  <tr>
    <td><strong>Hausgemachte Tomaten-Cremesuppe</strong><br />
      mit Knoblauchcroutons</td>
    <td class="price">2,80 €</td>
  </tr>
  <!-- weitere Gerichte -->
</table>
```

Die Auszeichnung mittels Tabellen funktioniert. Sie ist praktikabel, einfach und semantisch sauber. Wir können den Rest der Speisekarte auf dieselbe Art und Weise auszeichnen. Ich führe den Quelltext an dieser Stelle nicht komplett auf, weil sich der Schritt für Schritt ausgezeichnete Quelltext nur noch wiederholt, aber Sie können sich die `speisekarte.html` im Downloadpaket[7] anschauen.

ÜBUNG

Speisekarte, Markup

1. *Fügen Sie die Inhalte ein.*
2. *Speichern Sie und betrachten Sie die Speisekarte im Browser. Es sollte so aussehen, wie in Abbildung 8.35 gezeigt.*
3. *Validieren Sie die Speisekarte. Haben Sie alles korrekt ausgezeichnet?*

7 *http://jendryschik.de/wsdev/einfuehrung/buch/*

Abbildung 8.35: Die Speisekarte sieht unformatiert sehr unübersichtlich aus

Unformatierte Tabellen sind nicht sonderlich übersichtlich: Die Inhalte der Zellen kleben ohne Abstand oder Rahmen aneinander. Den Kopf schütteln müssen wir über die standardmäßige Darstellung der `caption`-Elemente: Tabellenüberschriften sehen im Browser so aus wie horizontal zentrierter Quelltext ohne Abstände oder typografische Besonderheiten.

Fangen wir mit den Tabellenüberschriften an. Wir erweitern den Block »Regeln für allgemeine Elemente« um eine Regel für `caption`. Hinter den Regeln für Anker ist ein guter Platz dafür. Die Tabellenüberschriften sollen so aussehen wie Überschriften zweiter Ordnung (`h2`), also übernehmen wir alle Regeln, ergänzen sie und passen sie an.

```
/**
 * @section Tabellen
 */
caption {
  color: #3C6331;
  background-color: white;
  font-size: 1.5em;
  text-align: left;
  font-weight: bold;
  text-transform: uppercase;
  padding: 0;
  margin: 1.5em 0 .5em 0;
}
```

Listing 8.40: Styling von Tabellenüberschriften

Anschließend nehmen wir Formatierungen speziell für Tabellen der Klasse `speisekarte` vor. Die Regeln schreiben wir an das Ende des Stylesheets innerhalb eines neuen Blocks »Regeln für spezielle Elemente«. Tragen wir zunächst einmal zusammen, was überhaupt zu tun ist:

- Wir müssen alle Tabellen auf die volle Breite (`100%`) bringen.
- Jede Zeile wird unterstrichen durch eine helle, dicke, gepunktete Linie. Mit der Eigenschaft `border` kommen wir an dieser Stelle nicht weiter, aber wir können ein passendes Hintergrundbild erstellen und einbinden.
- Alle Tabellenzellen benötigen ein wenig Platz, den wir über innere Abstände realisieren.
- Die Namen der Gerichte und der Preis benötigen einige Deklarationen zur Schriftformatierung: Farbe, Größe, Gewicht und Großschreibung.
- Als Letztes passen wir die vertikale Ausrichtung des Preises an und sorgen dafür, dass er nicht umbricht.

```
/**
 * Regeln für spezielle Elemente
 *
 * @section Speisekarte
 */
table.speisekarte { width: 100%; }

.speisekarte tr {
  color: inherit;
  background: url(images/speisekarte_bg.gif)
    bottom left repeat-x;
}
.speisekarte td,
.speisekarte th {
  padding: 10px 5px 8px 0;
}
.speisekarte td strong,
.speisekarte td.price {
  color: #cbb8ba;
  background-color: transparent;
  font-size: 1.2em;
  text-transform: uppercase;
  font-weight: bold;
}
.speisekarte td.price {
  vertical-align: bottom;
  white-space: nowrap;
}
```

Listing 8.41: Styling der Speisekarte

ÜBUNG

Speisekarte, Styling

1. *Ergänzen Sie Ihre* `default.css`.
2. *Speichern Sie und betrachten Sie die Speisekarte im Browser. Die Seite sollte nun aussehen wie der Entwurf.*
3. *Validieren Sie das Stylesheet. Hat sich irgendwo ein Fehler eingeschlichen, oder haben Sie alles richtig gemacht?*

Weiter geht es mit den Öffnungszeiten.

8.18 Schritt 16: Öffnungszeiten

ÜBUNG

Öffnungszeiten vorbereiten

1. *Bitte bereiten Sie das Dokument* `oeffnungszeiten.html` *so vor, dass wir uns um die Auszeichnung des Contents kümmern können. Erstellen Sie das Dokument auf Basis einer fertigen Unterseite, passen Sie Titel sowie Navigation an und leeren Sie das* `#content`*-Element.*
2. *Werfen Sie einen Blick auf die Inhalte (siehe Abbildung 8.3) und überlegen Sie sich, wie Sie die Tabelle auszeichnen würden. Falls nötig, rufen Sie sich Kapitel 5.7 noch einmal in Erinnerung.*

Die Überschrift zeichnen wir mit `h1` aus, den darauf folgenden Absatz mit `p`. Interessant ist die Auszeichnung der Tabelle, die wir gemeinsam Schritt für Schritt aufbauen.

Beginnen wir mit dem `table`-Element. Eine kurze Beschreibung des Tabellenaufbaus mittels `summary` ergibt Sinn. Darüber hinaus verwenden wir vorübergehend das Attribut `border`, um die Aufteilung in Zeilen und Spalten während der Auszeichnung besser sehen zu können. Später werden wir das Attribut wieder entfernen.

```
<table summary="Öffnungszeiten der Kochbar und des Biergartens,
  jeweils mittags und abends" id="oeffnungszeiten" border="1">
  <!-- Tabelleninhalt -->
</table>
```

Listing 8.42: Auszeichnung der Öffnungszeiten, Tabellencontainer

Die Tabellenüberschrift zeichnen wir mittels `caption` aus.

```
<table summary="Öffnungszeiten der Kochbar und des Biergartens,
  jeweils mittags und abends" id="oeffnungszeiten" border="1">
  <caption>Öffnungszeiten der Kochbar</caption>
  <!-- Tabelleninhalt -->
</table>
```

Listing 8.43: Auszeichnung der Öffnungszeiten, Tabellenüberschrift

Nun folgt ein kleiner Trick, der uns die weitere Arbeit enorm vereinfachen wird: Wir bauen das Gitterraster auf. Wir zählen die Zeilen und Spalten und fügen entsprechend viele `tr`- und `td`-Elemente ein. Es gibt sechs Zeilen; in der letzten Spalte sehen wir das am besten. Und wir zählen vier Spalten, am einfachsten zu sehen anhand der beiden mittleren Zeilen. Wir bauen also ein 4-x-6-Raster auf:

```
<table summary="Öffnungszeiten der Kochbar und des Biergartens,
  jeweils mittags und abends" id="oeffnungszeiten" border="1">
  <caption>Öffnungszeiten der Kochbar</caption>
  <tr>
    <td></td>
    <td></td>
    <td></td>
    <td></td>
  </tr>
  <tr>
    <td></td>
    <td></td>
    <td></td>
    <td></td>
  </tr>
    <tr>
    <td></td>
    <td></td>
    <td></td>
    <td></td>
  </tr>
    <tr>
    <td></td>
    <td></td>
    <td></td>
    <td></td>
  </tr>
    <tr>
    <td></td>
    <td></td>
    <td></td>
    <td></td>
```

```
    </tr>
      <tr>
      <td></td>
      <td></td>
      <td></td>
      <td></td>
    </tr>
  </table>
```

Listing 8.44: Auszeichnung der Öffnungszeiten, Raster

Befassen wir uns nun mit Zeilengruppen, den Elementen `thead`, `tfoot` und `tbody`. Können Sie sehen, ob und wie Sie die Zeilen aufteilen können? Ich denke, wir sollten so vorgehen, wie in Abbildung 8.36 zu sehen:

- Die umrahmten Zeilen sind der Tabellenrumpf, denn sie enthalten die eigentlichen Daten.
- Die Zeilen darüber enthalten nur Überschriften und bilden den Tabellenkopf.
- In den Zeilen darunter, die wir im Tabellenfuß zusammenfassen können, stehen Metainformationen.

		Tageszeiten	
		mittags	abends
Betrieb	Kochbar	12:00–15:00	18:00–22:00
	Biergarten	–	18:00–01:00
		Uhrzeiten in MESZ	
Öffnungszeiten gelten nur im Sommer			

Abbildung 8.36: Aufteilung der Zeilengruppen

Wir teilen die Spalten folglich in drei Gruppen zu je zwei Zeilen auf. Bitte beachten Sie die nicht intuitive Reihenfolge der Zeilengruppen: zunächst `thead`, dann schon `tfoot`, am Ende `tbody`.

```
<table summary="Öffnungszeiten der Kochbar und des Biergartens,
  jeweils mittags und abends" id="oeffnungszeiten" border="1">
  <caption>Öffnungszeiten der Kochbar</caption>
  <thead>
    <tr>
      <td></td>
```

```
        <td></td>
        <td></td>
        <td></td>
      </tr>
      <tr>
        <td></td>
        <td></td>
        <td></td>
        <td></td>
      </tr>
    </thead>
    <tfoot>
      <tr>
        <td></td>
        <td></td>
        <td></td>
        <td></td>
      </tr>
        <tr>
        <td></td>
        <td></td>
        <td></td>
        <td></td>
      </tr>
    </tfoot>
    <tbody>
      <tr>
        <td></td>
        <td></td>
        <td></td>
        <td></td>
      </tr>
        <tr>
        <td></td>
        <td></td>
        <td></td>
        <td></td>
      </tr>
    </tbody>
  </table>
```

Listing 8.45: Auszeichnung der Öffnungszeiten, Zeilengruppen

Bevor wir uns die Inhalte näher anschauen, kümmern wir uns zunächst um die Tabellenstruktur und verbinden mittels der Attribute `colspan` und `rowspan` alle Zellen so, wie es der Entwurf verlangt. Achten Sie darauf, dass anschließend Ihr Gitterraster noch vollständig und korrekt ist – schnell vergisst man, obsolet gewordene Zellen zu entfernen.

```
<table summary="Öffnungszeiten der Kochbar und des Biergartens,
  jeweils mittags und abends" id="oeffnungszeiten" border="1">
  <caption>Öffnungszeiten der Kochbar</caption>
  <thead>
    <tr>
      <td colspan="2" rowspan="2"></td>
      <td colspan="2"></td>
    </tr>
    <tr>
      <td></td>
      <td></td>
    </tr>
  </thead>
  <tfoot>
    <tr>
      <td colspan="2"></td>
      <td colspan="2"></td>
    </tr>
    <tr>
      <td colspan="4"></td>
    </tr>
  </tfoot>
  <tbody>
    <tr>
      <td rowspan="2"></td>
      <td></td>
      <td></td>
      <td></td>
    </tr>
      <tr>
      <td></td>
      <td></td>
      <td></td>
    </tr>
  </tbody>
</table>
```

Listing 8.46: Auszeichnung der Öffnungszeiten, Zeilen und Spalten verbinden

Nun können wir die Zellen befüllen. Gleichzeitig sollten wir uns Gedanken dazu machen, welche Zellen Tabellenkopf- und welche Datenzellen sind. Stimmen Sie mit mir überein, dass alle hellgrauen und die beiden leeren Zellen Datenzellen sind, während »Tageszeiten«, »mittags«, »abends«, »Betrieb«, »Kochbar« und »Biergarten« Kopfzellen sind?

```
<table summary="Öffnungszeiten der Kochbar und des Biergartens,
  jeweils mittags und abends" id="oeffnungszeiten" border="1">
  <caption>Öffnungszeiten der Kochbar</caption>
  <thead>
    <tr>
      <td colspan="2" rowspan="2"></td>
      <th colspan="2">Tageszeiten</th>
    </tr>
    <tr>
      <th>mittags</th>
      <th>abends</th>
    </tr>
  </thead>
  <tfoot>
    <tr>
      <td colspan="2"></td>
      <td colspan="2">Uhrzeiten in MESZ</td>
    </tr>
    <tr>
      <td colspan="4">Öffnungszeiten gelten nur im Sommer</td>
    </tr>
  </tfoot>
  <tbody>
    <tr>
      <th rowspan="2">Betrieb</th>
      <th>Kochbar</th>
      <td>12:00-15:00</td>
      <td>18:00-22:00</td>
    </tr>
    <tr>
      <th>Biergarten</th>
      <td>-</td>
      <td>18:00-01:00</td>
    </tr>
  </tbody>
</table>
```

Listing 8.47: Auszeichnung der Öffnungszeiten und Einfügen der Inhalte

Wir sind mit der Auszeichnung der Tabelle vorerst fertig. Wie Abbildung 8.37 zeigt, ist unser Zwischenergebnis allerdings weder wirklich übersichtlich noch hübsch anzuschauen. Das ändern wir nun durch einige CSS-Regeln.

Für die Tabelle der Öffnungszeiten deklarieren wir zunächst das Modell der zusammenfallenden Rahmen, geben an, dass Browser leere Tabellenzellen anzeigen sollen, und entfernen den Rahmen.

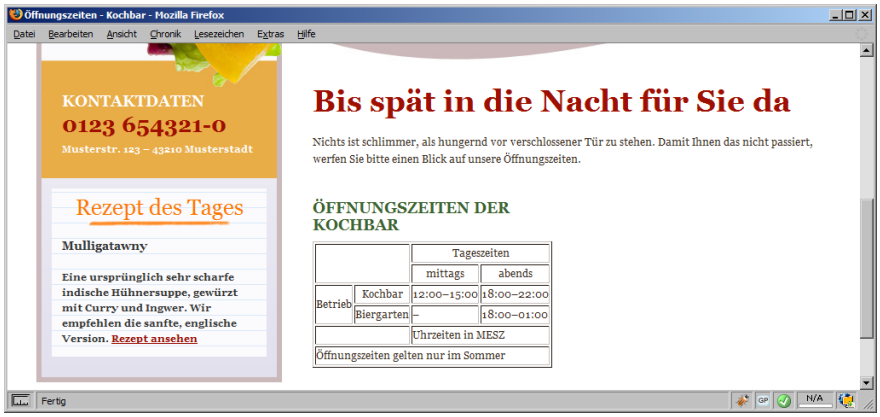


Abbildung 8.37: Die unformatierte Tabelle sieht nur wegen des Rahmens einigermaßen übersichtlich aus

```
/**
 * @section Öffnungszeiten
 */
table#oeffnungszeiten {
  border-collapse: collapse;
  empty-cells: show;
  border: 0;
}
```

Listing 8.48: Styling der Öffnungszeiten, Tabellencontainer

Anschließend nehmen wir einige Zellendeklarationen vor. Wir deklarieren einen 4 px breiten Rahmen, die Schrift- und Hintergrundfarbe sowie innere Abstände.

```
#oeffnungszeiten td,
#oeffnungszeiten th {
  border: 4px solid white;
  color: #5D5D5D;
  background-color: #F5F4F8;
  padding: .5em 1em;
}
```

Listing 8.49: Styling der Öffnungszeiten, Zellen

Beginnen wir mit den Feinheiten. Elemente vom Typ `th` sehen im Tabellenkopf anders aus als im Tabellenrumpf; die Schrift im Tabellenrumpf ist kleiner als im Rest der Tabelle. Über Kontextselektoren können wir die Zellen in den unterschiedlichen Bereichen ansprechen und die entsprechenden Deklarationen vornehmen. Zudem machen wir Gebrauch von Attributselektoren, um die

äußeren Überschriftenzellen, die sich über zwei Spalten oder Zeilen erstrecken, von den inneren Überschriftenzellen zu unterscheiden.

```
#oeffnungszeiten thead td,
#oeffnungszeiten thead th {
  color: #5D5D5D;
  background-color: #E8E7EE;
  text-align: center;
}
#oeffnungszeiten thead th[colspan="2"] {
  color: white;
  background-color: #CBB8BA;
  font-weight: bold;
}

#oeffnungszeiten tbody th {
  color: #5D5D5D;
  background-color: #e8dbc5;
  text-align: center;
}
#oeffnungszeiten tbody th[rowspan="2"] {
  color: white;
  background-color: #E8AD47;
  font-weight: bold;
}

#oeffnungszeiten tfoot td {
  text-align: center;
  font-size: .9em;
}
```

Listing 8.50: Styling der Öffnungszeiten, spezielle Formatierung der Zellen

Damit haben wir die Formatierung der Seite » Öffnungszeiten « abgeschlossen.

ÜBUNG

Öffnungszeiten

1. *Fügen Sie die Inhalte ein und speichern Sie.*
2. *Ergänzen Sie Ihre* `default.css`.
3. *Speichern Sie und betrachten Sie die Öffnungszeiten im Browser. Die Tabelle sollte nun so aussehen wie der Entwurf.*
4. *Validieren Sie das Markup und Ihr Stylesheet, um sicherzustellen, dass Sie keine Fehler gemacht haben.*

8.19 Schritt 17: Kontaktformular

Die Auszeichnung von Formularen gehört zu den schwierigsten Aufgaben bei der Erstellung von Webseiten und – meiner Ansicht nach – auch zu denen, die am wenigsten Spaß machen. Es handelt sich dabei um eine technisch orientierte Arbeit, bei der Webautoren häufig durch die Arbeit der Programmierer eingeschränkt werden, die die im Hintergrund stattfindenden Prozesse definieren. Formulare sind häufig die Schnittstelle zwischen Nutzern und interaktiven Prozessen. Dadurch erfordern sie eine erhöhte Beachtung der Forderungen der Web-Ergonomie und Zugänglichkeit.

ÜBUNG

Kontaktseite vorbereiten

1. *Bitte bereiten Sie das Dokument* `kontakt.html` *so vor, dass wir uns um die Auszeichnung des Kontaktformulars kümmern können. Erstellen Sie das Dokument auf Basis einer bestehenden Unterseite, passen Sie Titel sowie Navigation an und leeren Sie den Content-Container.*

Analog zur Auszeichnung der Tabelle mit den Öffnungszeiten gehen wir auch hier Schritt für Schritt vor, damit Sie lernen, wie Webautoren bei der Auszeichnung komplexer Inhalte vorgehen. Beginnen wir mit dem Formularcontainer. Als Wert für das `action`-Attribut geben wir `#top` an; sendet ein Nutzer das Formular ab, springt er wieder nach oben an den Beginn der Seite. Wir folgen den Empfehlungen (siehe Kapitel 5.10.1) und wählen `post` als HTTP-Methode.

```
<form action="#top" method="post">
</form>
```

Listing 8.51: Auszeichnung des Kontaktformulars, Formularcontainer

Schauen wir uns das Formular einmal genauer an (siehe Abbildung 8.4). Wir sehen drei Bereiche: »Persönliche Angaben«, »Angaben zur Reservierung« und »Ihre Nachricht«. Wir können Elemente vom Typ `fieldset` und `legend` verwenden, um das Formular aufzuteilen. Darunter folgt die Absenden-Schaltfläche, die wir in einem Absatz mit der ID `submit` einschließen.

```
<form action="#top" method="post" id="contactForm">
  <fieldset>
    <legend>Persönliche Angaben</legend>
  </fieldset>

  <fieldset>
    <legend>Angaben zur Reservierung</legend>
  </fieldset>
```

```
    <fieldset>
      <legend>Ihre Nachricht</legend>
    </fieldset>

    <p id="submit"><input value="Absenden" type="submit" /></p>
  </form>
```

Listing 8.52: Auszeichnung des Kontaktformulars, Aufteilung in drei Bereiche

Wenn wir uns die Darstellung der Elemente innerhalb eines Fieldsets anschauen, kommen wir schnell zu dem Ergebnis, dass wir alle Steuerelemente und deren Beschriftung in Tabellen anordnen sollten. Diese Vorgehensweise lässt sich semantisch rechtfertigen, schließlich stehen Beschriftung und Steuerelement in einer Beziehung zueinander. Vor allem aber ist eine Tabelle sehr gut dafür geeignet, alle Elemente bei optimaler Skalierbarkeit in einer Flucht anzuordnen, wie es der Entwurf fordert. Andere Auszeichnungsformen sind natürlich denkbar, für den Einstieg allerdings zu schwierig zu implementieren.

Bei der Auszeichnung der Steuerelemente kommt nichts auf Sie zu, was Sie nicht bereits in Kapitel 5.10, »Formulare«, gelernt haben. Erlauben Sie mir dennoch einige Anmerkungen.

- Eine Person wird entweder mit »Herr« oder »Frau« angeredet, sinnvoll sind daher Optionsfelder, da Nutzer von Optionsfeldern gleichen Namens in einem Formular nur eines auswählen können.
- Bitte achten Sie auf den Bezug zwischen Beschriftung und Steuerelement. Wir verwenden beide Varianten, zum einen den direkten Bezug, beispielsweise bei der Anrede, zum anderen den indirekten Bezug über das `for`-Attribut.
- Jedes Auswahlmenü können wir auch als eine Liste von Optionsfeldern darstellen, übersichtlich wäre eine solche allerdings nicht. Die Auswahl des richtigen Steuerelements für einen bestimmten Zweck gehört zu den Aufgaben eines Webautors.
- Die Vorbelegung des Datums soll die Eingabe im gewünschten Format erleichtern. Wohlgemerkt: erleichtern, nicht sicherstellen. Dies erledigen Webautoren und Programmierer professioneller Websites mittels client- und serverseitiger Technologien. Häufig überprüfen Formulare bei der Eingabe oder beim Absenden mittels JavaScript, ob alle Daten den Vorgaben entsprechen und vollständig sind, und nach dem Absenden überprüft ein Script auf dem Server die Konsistenz der Daten.

```
<form action="#top" method="post" id="contactForm">
  <fieldset>
    <legend>Persönliche Angaben</legend>

    <table>
      <tbody>
        <tr class="odd">
          <th>Anrede</th>
          <td><label><input name="anrede" value="herr"
            checked="checked" type="radio" /> Herr</label>
            <label><input name="anrede" value="frau"
            type="radio" /> Frau</label></td>
        </tr>
        <tr>
          <th><label for="name">Name</label></th>
          <td><input name="name" id="name" size="30"
            type="text" /></td>
        </tr>
        <tr class="odd">
          <th><label for="telefon">Telefon</label></th>
          <td><input name="telefon" id="telefon" size="30"
            type="text" /></td>
        </tr>
        <tr>
          <th><label for="e-mail">E-Mail</label></th>
          <td><input name="e-mail" id="e-mail" size="30"
            type="text" /></td>
        </tr>
      </tbody>
    </table>
  </fieldset>

  <fieldset>
    <legend>Angaben zur Reservierung</legend>

    <p>Falls Sie einen Tisch reservieren möchten, benötigen wir
      noch folgende Informationen. Bitte beachten Sie auch
      unsere <a href="oeffnungszeiten.html">
      Öffnungszeiten</a>.</p>

    <table>
      <tbody>
        <tr class="odd">
          <th><label for="date">Datum</label></th>
          <td><input name="date" id="date" size="30" value="zum
            Beispiel 24.12.2008" onblur="if(this.value == '')
            this.value='zum Beispiel 24.12.2008'"
            onfocus="if(this.value == 'zum Beispiel 24.12.2008')
            this.value=''" type="text" /></td>
```

```
    </tr>
    <tr>
      <th><label for="time">Uhrzeit</label></th>
      <td>
        <select name="time" id="time">
           <option>bitte wählen...</option>
           <option>12:00</option>
           <option>12:30</option>
           <option>13:00</option>
           <option>13:30</option>
           <option>14:00</option>
           <option>18:00</option>
           <option>18:30</option>
           <option>19:00</option>
           <option>19:30</option>
           <option>20:00</option>
           <option>20:30</option>
           <option>21:00</option>
           <option>21:30</option>
           <option>22:00</option>
           <option>22:30</option>
           <option>23:00</option>
           <option>23:30</option>
        </select>
      </td>
    </tr>
    <tr class="odd">
      <th><label for="persons">Personen</label></th>
      <td>
        <select name="persons" id="persons">
           <option selected="selected">1-2</option>
           <option>3-5</option>
           <option>6-10</option>
           <option>10-15</option>
           <option>mehr als 15</option>
        </select>
      </td>
    </tr>
    <tr>
      <th><label for="smoker">Raucherbereich</label></th>
      <td><input name="smoker" id="smoker"
        type="checkbox" /></td>
    </tr>
    <tr class="odd">
      <th><label for="beerGarden">Biergarten</label></th>
      <td><input name="beerGarden" id="beerGarden"
        type="checkbox" /></td>
    </tr>
  </tbody>
```

```
      </table>
    </fieldset>

    <fieldset>
      <legend>Ihre Nachricht</legend>

      <textarea name="nachricht" id="nachricht"
        cols="33" rows="8"></textarea>
    </fieldset>

    <p id="submit"><input value="Absenden" type="submit" /></p>
  </form>
```

Listing 8.53: Auszeichnung des Kontaktformulars, Steuerelemente

Damit haben wir die Auszeichnung des Formulars abgeschlossen. Es sieht jetzt so aus, wie in Abbildung 8.38 gezeigt. Kommen wir direkt zum Styling. Die gesamte Kochbar-Website enthält nur dieses eine Formular, alle Formatierungen können wir daher im Bereich der Regeln für allgemeine Elemente vornehmen.

Abbildung 8.38: Das unformatierte Kontaktformular

Zunächst verschaffen wir den Fieldsets ein wenig Raum und gestalten ihre Rahmen und Namen.

```
/**
 * @section Formulare
 */
fieldset {
  border: 1px solid #cbb8ba;
  margin: 1em 0;
  padding: 0 1em;
}
legend {
  color: white;
  background-color: #cbb8ba;
  padding: .3em 1em;
}
```

Listing 8.54: Styling des Kontaktformulars, Fieldsets

Abbildung 8.39: Fieldset-Elemente teilen das Formular in mehrere Gruppen auf

Schauen wir uns an, was wir tun müssen, um die Steuerelemente zu gestalten.

- Wir korrigieren die horizontale Ausrichtung von Kontrollkästchen und Optionsfeldern, die derzeit ein wenig zu hoch positioniert sind.
- Für Texteingabefelder deklarieren wir einen anderen Rahmen und vergrößern die inneren Abstände.
- Dasselbe tun wir für `textarea`-Elemente. Zudem geben wir einen vertikalen äußeren Abstand an und vergrößern das Texteingabefeld auf die volle Breite. Am besten messen wir die genaue Breite im Entwurf pixelgenau ab. Die Alternative ist die Deklaration von `width: 100%;` zusammen mit `border-left` und `border-right`, allerdings führt das zu einer Überbreite (100 % + 2 px) und dies wiederum häufig zu Problemen, die wir vermeiden wollen.
- Den Text der Absenden-Schaltfläche formatieren wir fett und passen Farben und Abstände an.
- Da wir schon dabei sind: Lassen Sie uns die Beschriftungen gleich mit formatieren. Wir haben dafür gesorgt, dass Beschriftungen anklickbar sind, um das zugehörige Steuerelement zu aktivieren, folglich setzen wir einen entsprechenden Mauszeiger, der dieses Verhalten auch visualisiert. Ein wenig Padding sorgt für mehr Abstand zwischen den Beschriftungen und den dazugehörigen Steuerelementen. Bei den Optionsfeldern der Anrede ist gut zu sehen, dass das notwendig ist.

```
input[type="checkbox"],
input[type="radio"] {
  vertical-align: middle;
}
input[type="text"] {
  border: 1px solid #cbb8ba;
  padding: 3px 10px;
}
input[type="submit"] {
  color: #2e050a;
  background-color: #f5f4f8;
  font-weight: bold;
  padding: 0 6px;
}
textarea {
  border: 1px solid #cbb8ba;
  margin: 1em 0;
  padding: 5px 15px;
  width: 535px;
}
label {
  cursor: pointer;
  padding-right: .5em;
}
```

Listing 8.55: Styling des Kontaktformulars, Steuerelemente

Abbildung 8.40: Formatierung der Steuerelemente

Nun formatieren wir die Tabellen, die die Beschriftungen und Steuerelemente ausrichten. Dabei nehmen wir alle Deklarationen generell für Tabellen innerhalb von Formularcontainern vor (Selektor `form table`).

Zunächst setzen wir die Breite auf `100%` und verschaffen den Tabellen ein wenig mehr vertikalen äußeren Abstand.

```
/* Tabellen in Formularen */
form table {
  width: 100%;
  border-collapse: collapse;
  margin: .8em 0;
}
```

Listing 8.56: Styling des Kontaktformulars, Tabellen

Anschließend kümmern wir uns um die Einfärbung der Tabellenzeilen. Der Entwurf sieht vor, dass die Zeilen abwechselnd hell und dunkel eingefärbt sind. Ungerade Zeilen sind hell, gerade Zeilen dunkel. Tja, wie wollen wir hier

vorgehen? Wir könnten jeder Zeile eine Klasse vergeben, z. B. `odd` und `even`, oder nur jeder zweiten und diese dann mittels eines Klassenselektors im Stylesheet ansprechen. Allerdings bin ich der Ansicht, dass wir hier einen sehr guten Anwendungsfall für Progressive Enhancement haben: Warum setzen wir nicht die Struktur-Pseudoklasse `:nth-child(n)` ein? In Kapitel 6.5.5 habe ich beschrieben, wie wir sie verwenden können, um Tabellenzeilen unterschiedlich einzufärben. Hier gehen wir ein wenig anders vor: Wir deklarieren zunächst, dass die Browser alle Zeilen (`form tr`) dunkel einfärben sollen, und anschließend, dass gerade Zeilen (`form tr:nth-child(odd)`) hell sein sollen. Browser, die die Struktur-Pseudoklasse verstehen, färben die Zeilen nun abwechselnd hell und dunkel ein, alle anderen nicht. Nutzer älterer Browser werden nicht einmal merken, dass ihnen etwas entgeht, während bei Nutzern moderner Browser die Tabelle ein wenig hübscher aussieht.

```
form tr {
  color: #5e2021;
  background-color: #e8e7ee;
  border-bottom: 4px solid white;
}
form tr:nth-child(odd) {
  color: #5e2021;
  background-color: #f5f4f8;
}
```

Listing 8.57: Styling des Kontaktformulars, Einfärbung der Zeilen

Nun fehlen noch einige Deklarationen für die Zellen. Die Werte der Eigenschaft `padding` sind für Kopf- und Datenzellen die gleichen, also fassen wir sie in einer Regel zusammen. Anschließend deklarieren wir die Breite von `th`-Elementen und die horizontale Ausrichtung ihrer Inhalte.

```
form td,
form th {
  padding: 5px 5px 5px 15px;
}
form th {
  width: 40%;
  text-align: left;
}
```

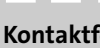
Listing 8.58: Styling des Kontaktformulars, Tabellenzellen

Abschließend formatieren wir den Absatz, der die Absenden-Schaltfläche enthält.

```
form p#submit {
  color: white;
  background-color: #cbb8ba;
```

```
    padding: 10px 15px;
    margin: 2.5em 0;
  }
```

Listing 8.59: Styling des Kontaktformulars, Absenden-Schaltfläche

 ÜBUNG

Kontaktformular

1. *Ergänzen Sie die* `kontakt.html` *und das Stylesheet.*
2. *Speichern Sie und betrachten Sie das Kontaktformular im Browser. Die Darstellung sollte dem Entwurf entsprechen.*
3. *Validieren Sie beide Dokumente. Ist die Kochbar-Website weiterhin fehlerfrei?*

8.20 Schritt 18: Rezept des Tages

Jetzt fehlt nur noch die Auszeichnung des Rezepts des Tages, dann sind wir erst einmal fertig.

ÜBUNG

Rezeptseite vorbereiten

1. *Erstellen Sie einen Unterordner »rezepte«.*
2. *Erstellen Sie darin eine Datei »mulligatawny.html« und nehmen Sie die üblichen Anpassungen vor.*
3. *Passen Sie alle relativen Pfade an. Wir befinden uns in einem Unterverzeichnis, das bedeutet, alle relativen URLs, die im Dokument stehen, führen ins Leere. Wenn Sie die Seite aufbauen und parallel im Browser prüfen, wie sie aussieht, werden Sie feststellen, dass das Stylesheet nicht geladen wird, ebenso wenig das Logo. Das liegt daran, dass die Pfade nicht stimmen. Der relative Pfad zum Logo lautet nicht mehr* `images/logo.png`*, sondern* `../images/logo.png`*. Korrigieren Sie alle Pfade und testen Sie, ob Sie nichts vergessen haben.*
4. *Laden Sie die Seite im Browser und lassen Sie uns loslegen!*

Wir haben auf allen Seiten der Kochbar links in der Sidebar einen Teaser auf das Rezept des Tages. Dieser Teaser ergibt auf der Rezeptseite jedoch keinen Sinn: Wir sind ja schon dort, wohin der Teaser uns verweisen möchte. Also ändern wir ihn und lassen ihn die Speisekarte anteasern.

```
<div id="sidebar">
  <div id="sidebar-inner">
    <div id="contact">
      <h2>Kontaktdaten</h2>
```

```
        <p class="tel">0123 654321-0</p>
        <p class="address">Musterstr. 123 - 43210 Musterstadt</p>
      </div>

      <div id="teaser">
        <h2>Speisekarte</h2>

        <p>Die Mulligatawny ist eines unserer vielfältigen,
          internationalen Gerichte. Wir haben noch mehr zu bieten!
          <a href="../speisekarte.html">Speisekarte
          ansehen</a></p>
      </div>
    </div>
  </div>
```

Listing 8.60: Auf der Seite des Rezepts des Tages gibt es einen Teaser auf die Speisekarte

Ein Kochrezept besteht aus mindestens zwei Teilen: den Zutaten und der Zubereitung. Hinzu kommen häufig Hinweise, etwa Vorschläge zu Beilagen, Haltbarkeit oder Aufbewahrung. Anstatt diesen Aufbau mit »langweiligen« Zwischenüberschriften wiederzugeben, können wir die einzelnen Bestandteile mit einer Definitionsliste zusammenfassen. Ich möchte wie folgt vorgehen:

```
<dl>
  <dt>Zutaten für 6 Portionen</dt>
  <dd>
    <!-- Zutaten -->
  </dd>

  <dt>Zubereitung</dt>
  <dd>
    <!-- Zubereitung -->
  </dd>

  <dt>Hinweis</dt>
  <dd>
    <!-- Hinweis -->
  </dd>
</dl>
```

Sind Sie damit einverstanden? Ich gebe zu: Das ist eine ungewöhnliche Art, Definitionslisten einzusetzen, die Sie ansonsten im Web zumeist nur bei Standardfällen antreffen wie bei der Auszeichnung eines Glossars oder einer Literaturliste – wenn überhaupt. Aber warum sollten wir das Einsatzgebiet von Definitionslisten nicht sinnvoll erweitern?

Schauen wir uns an, wie wir die dd-Elemente füllen.

- Die Liste der Zutaten ist unstrittig eine solche. Allerdings sollten wir uns die Frage stellen, ob die Einträge der Liste geordnet oder ungeordnet sind. Denkbar ist beides, wir belassen es jedoch bei der Auszeichnung mit ul.
- Die Informationen zur Zubereitung sind eine Abfolge von Arbeitsanweisungen, daher können wir sie als geordnete Liste auszeichnen. Auch dies ist eine gewagte Form der Auszeichnung, aber sie ist semantisch korrekt, und warum sollten Webautoren nicht auch hin und wieder ein wenig kreativ sein?
- Der Zubereitungshinweis ist reine Prosa. Wir könnten ihn als Absatz auszeichnen, aber es reicht, den Text einfach als Inhalt des dd-Elements zu notieren.

Das Rezept enthält zahlreiche Abkürzungen, die wir allesamt mit abbr oder acronym auszeichnen.

```
<div id="content">
  <h1>Rezept des Tages: Mulligatawny</h1>

  <p>Die Mulligatawny ist eine brennend scharfe indische Suppe.
    Der Name stammt von den Wörtern »Mullaga« und »Thanni« ab,
    was so viel bedeutet wie »Pfefferwasser«. Es ist eine
    Hühner-Tomaten-Suppe, die sehr stark mit Curry und
    Muskatnuss gewürzt ist. Es werden oft Gemüsestreifen, Nüsse
    und Reis, bisweilen auch Portwein hinzugenommen. Das
    vorliegende Rezept ist die sanfte, englische Version.</p>

  <dl>
    <dt>Zutaten für 6 Portionen</dt>
    <dd>
      <ul>
        <li>2 Zwiebeln</li>
        <li>2 Knoblauchzehen</li>
        <li>1 Bund Suppengrün</li>
        <li>1 Petersilienwurzel</li>
        <li>1 etwa 5 <abbr title="Zentimeter">cm</abbr> langes
          Stück frische Ingwerwurzel</li>
        <li>1 Suppenhuhn (etwa 2 <abbr
          title="Kilogramm">kg</abbr>)</li>
        <li>2 <abbr title="Liter">l</abbr> Salzwasser</li>
        <li>Salz</li>
        <li>250 <abbr title="Gramm">g</abbr> junge Möhren</li>
        <li>1 Bund Lauchzwiebeln</li>
        <li>1 kleiner säuerlicher Apfel</li>
        <li>250 <abbr>g</abbr> gekochter Schinken am Stück</li>
        <li>1 <abbr title="Esslöffel">EL</abbr> Butter</li>
        <li>2 <abbr>EL</abbr> Currypulver</li>
```

```
    <li>1 bis 2 kleine, rote Chilischoten</li>
    <li>300 <abbr>g</abbr> gekochter Langkornreis</li>
    <li>2 <abbr>EL</abbr> Zitronensaft</li>
    <li>1/2 Bund Petersilie</li>
    <li>2 Eier</li>
    <li>1/8 <abbr>l</abbr> süße Sahne</li>
  </ul>
</dd>

<dt>Zubereitung</dt>
<dd>
  <ol>
    <li>Zwiebeln und Knoblauch abziehen und grob zerteilen.
      Suppengrün putzen und grob hacken. Petersilien- und
      Ingwerwurzel schälen und in Stücke schneiden.</li>
    <li>Alles Gemüse mit dem Huhn und dem Salzwasser in
      einen ausreichend großen Topf geben und zum Kochen
      bringen. Zugedeckt auf kleiner Flamme 2 bis 2 1/2
      Stunden garen, bis das Huhn ganz weich ist.</li>
    <li>Das Huhn herausnehmen, die Haut abziehen, das
      Fleisch von den Knochen lösen und klein
      schneiden.</li>
    <li>Die Brühe sieben, erkalten lassen und
      entfetten.</li>
    <li>Die Möhren putzen und in feine Scheiben
      schneiden.</li>
    <li>Die Lauchzwiebeln putzen, waschen und mit allen
      saftigen grünen Blättern in etwa fingerbreite Stücke
      schneiden.</li>
    <li>Den Apfel vierteln, schälen, vom Kerngehäuse
      befreien und würfeln.</li>
    <li>Den Schinken in kleine Würfel schneiden.</li>
    <li>Die Chilischoten klein hacken.</li>
    <li>Die Butter in einem großen Topf zerlassen. Das
      Currypulver und die Chilischoten darin bei schwacher
      Hitze unter Rühren anrösten.
    </li>
    <li>Möhren, Lauchzwiebeln, Apfel und Schinken zugeben
      und alles bei mittlerer Hitze unter ständigem Wenden
      einige Minuten schmoren.
    </li>
    <li>Die Brühe zugießen und kräftig aufkochen. Das
      Fleisch, den Reis und den Zitronensaft zugeben und
      zugedeckt auf der abgeschalteten Kochplatte etwa 5
      Minuten ziehen lassen. Die Suppe darf nicht mehr
      kochen!</li>
    <li>Die Petersilie waschen und hacken.</li>
    <li>Die Eier trennen. Eigelb mit Sahne und etwas heißer
      Suppe verquirlen und in die Mulligatawny rühren.</li>
```

```
        <li>Die Suppe abschmecken und mit der gehackten
          Petersilie bestreut servieren.</li>
      </ol>
    </dd>

    <dt>Hinweis</dt>
    <dd>Wer möchte, kann die Mulligatawny auch ohne Fleisch
      zubereiten. In jedem Fall sollte die fertige Suppe sofort
      serviert werden, damit der Reis nicht zu weich und die
      Mulligatawny nicht zu dick wird.</dd>
  </dl>
</div>
```

Listing 8.61: Auszeichnung des Rezepts des Tages

Abbildung 8.41: Die Standardformatierung einer Definitionsliste ist sehr reduziert

Die fertig ausgezeichnete Definitionsliste sieht sehr langweilig und unübersichtlich aus (siehe Abbildung 8.41). Wir benötigen allerdings nicht viel CSS,

um sie wie im Entwurf zu formatieren. Wir setzen die richtigen Außenabstände für `ul`, `ol` und `li` und formatieren `dt`-Elemente so, dass sie wie Überschriften zweiter Ordnung aussehen.

```
/**
 * @section Listen
 */
ul, ol { margin: 1em 0 1em 2em; }

li { margin: 0 0 .5em 0; }

dt {
  color: #3C6331;
  background-color: white;
  font-size: 1.5em;
  text-align: left;
  font-weight: bold;
  text-transform: uppercase;
  padding: 0;
  margin: 1.5em 0 .5em 0;
}
```

Listing 8.62: Allgemeine Listenformatierungen

Wenn Sie dieses CSS in Ihr Stylesheet einfügen, werden Sie feststellen, dass die Hauptnavigation ein wenig von der unteren Kante des Kopfbereichs abrückt. Das müssen noch korrigieren:

```
/**
 * @section Hauptmenü
 */
#head ul {
  position: absolute;
  bottom: 0;
  right: 30px;
  margin: 0;
}
```

Listing 8.63: Korrektur des äußeren Abstands des Hauptmenüs

ÜBUNG

Kontaktformular

1. *Ergänzen Sie die* `mulligatawny.html` *und das Stylesheet.*
2. *Speichern Sie und betrachten Sie das Rezept im Browser. Die Darstellung sollte dem Entwurf entsprechen.*
3. *Lassen Sie den Validator über die Seite laufen. Wenn er weiterhin nichts bemängelt, ist die gesamte Kochbar fehlerfrei. Sehr gute Arbeit!*

8.21 Schritt 19: Browsertests und Bugfixes

Wir haben uns die Kochbar-Website bisher nur im Firefox angeschaut. Diese Vorgehensweise ist empfehlenswert, da Sie stets auf modernen Browsern entwickeln sollten, die sich eng an die W3C-Empfehlungen halten. Aber vergessen Sie niemals, auch in anderen Browsern zu testen! Werfen Sie in diesem Zusammenhang noch einmal einen Blick in Kapitel 2.7.9 »Browserkrieg«.

Als Erstes überprüfen wir die Darstellung in dem Browser, der uns vermutlich die meiste Arbeit machen wird: Internet Explorer 6. Allerdings – und ich bin überrascht – sieht die Startseite genau so aus, wie sie aussehen soll! Auch die Seite über die Kochbar und das Impressum sind fehlerfrei. Erst bei der Speisekarte gibt es gleich mehrere Darstellungsfehler (siehe Abbildung 8.42). Zwischen dem Absatz und der Tabelle ist ein großer Abstand; offenbar beginnt die Tabelle erst unterhalb der Sidebar. Zudem stimmen die Abstände der `caption`-Elemente nicht.

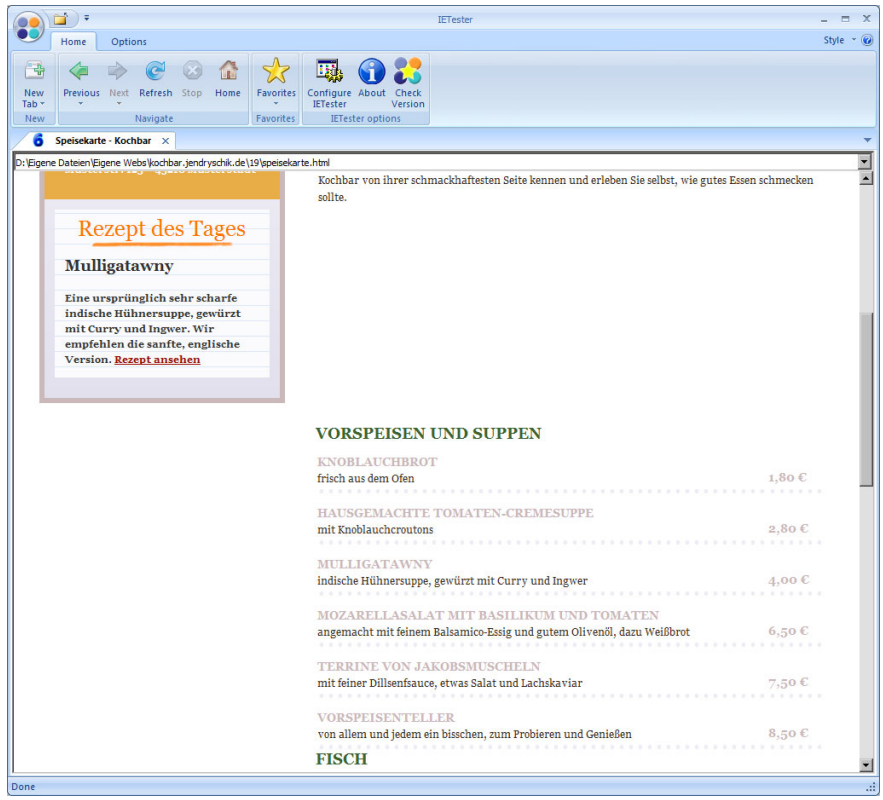


Abbildung 8.42: IE 6: Die Speisekarte beginnt erst unterhalb der Sidebar

Der IE 6 hat ein Problem damit, dass sich die Tabelle über die volle Content-Breite erstreckt. Bevor wir uns zu lange damit aufhalten, herauszufinden, woran das liegt, geben wir einfach einen festen Wert an, mit dem sich auch der IE zufriedengibt.

```
 /**
 * Regeln für spezielle Elemente
 *
 * @section Speisekarte
 */
table.speisekarte { width: 590px; }
```

Listing 8.64: Korrektur der Tabellenbreite der Speisekarte für den IE 6

Der Internet Explorer 6 kennt die Eigenschaft `margin` für das `caption`-Element nicht. Wir müssen anders vorgehen, um die gewünschten Abstände zu erhalten. Wir setzen `margin` auf `0` und deklarieren die Abstände für `padding`; das interpretiert dann auch der IE 6.

```
/**
 * @section Tabellen
 */
caption {
  color: #3C6331;
  background-color: white;
  font-size: 1.5em;
  text-align: left;
  font-weight: bold;
  text-transform: uppercase;
  margin: 0;
  padding: 1.5em 0 .5em 0;
}
```

Listing 8.65: Korrektur der Abstände von Tabellenüberschriften für den IE 6

Während wir die IE-Bugs korrigieren, müssen wir natürlich parallel die Auswirkungen im Firefox im Auge behalten. Es sieht aber gut aus: Wir haben die Darstellungsfehler im IE ohne Nebenwirkungen korrigiert.

Die Tabelle mit den Öffnungszeiten sieht im IE 6 im Wesentlichen richtig aus. Die Deklarationen für die äußeren Überschriftenzellen stellt er nicht dar, allerdings haben wir das bereits einkalkuliert, als wir einen Attributselektor eingesetzt haben, den der IE 6 nicht versteht.

Die Kontaktseite hingegen sieht ziemlich kaputt aus (siehe Abbildung 8.43). Was ist denn da passiert?

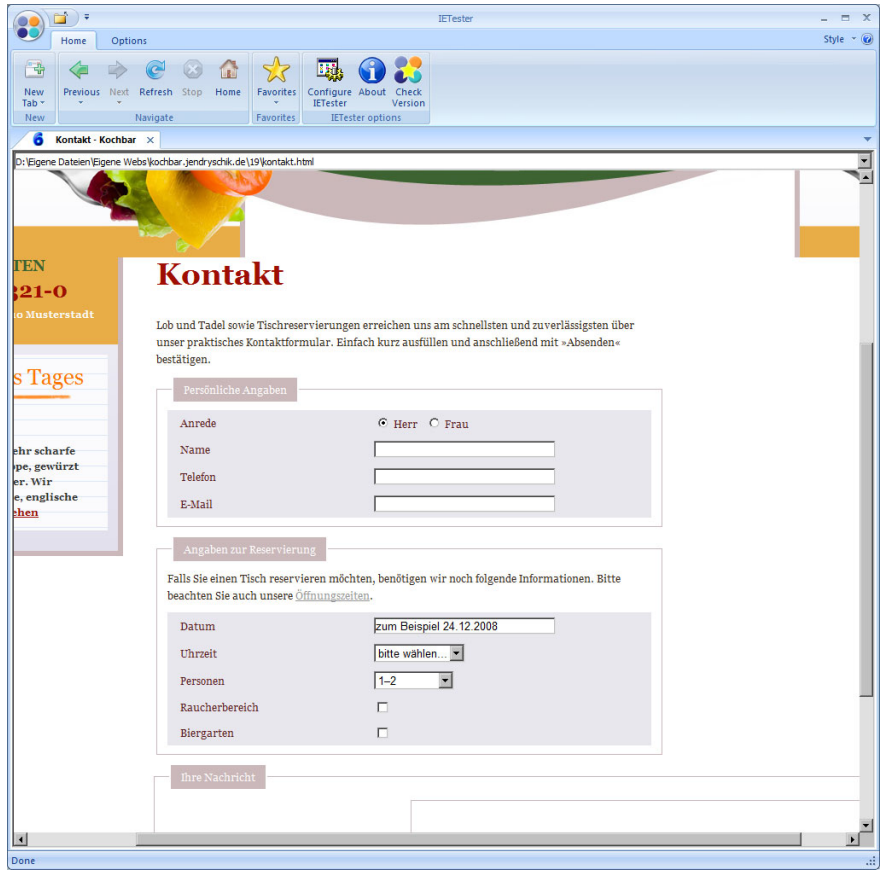


Abbildung 8.43: Die Kontaktseite sieht im IE 6 schlimmer aus, als sie eigentlich ist

Ich gebe zu, diese Frage ist nicht leicht zu beantworten. Eines ist klar: Wir haben es hier mit einem `hasLayout`-Bug zu tun, aber mit welchem? Und was genau ist das Problem? Es sieht so aus, als würde das `textarea`-Element sich genau den Abstand nach links gönnen, den das `#content`-Element für `margin-left` hat. Aber weshalb? Es könnte sich um den »*IE inherited margin bug*«[8] handeln, aber so ganz trifft er nicht zu. Woher weiß die `textarea`-Box überhaupt von dem Margin-Bereich der äußeren Box? Müsste dieser Margin dafür nicht durch die dazwischenliegende `#content`-Box hindurchtunneln?

Wie auch immer, wir lösen das Problem, indem wir zur Allzweckwaffe gegen IE-Probleme greifen: Wir sorgen dafür, dass das Element »Layout« hat, das wir im Verdacht haben, das Problem zu verursachen. Nach einigem Ausprobieren landen wir bei `#content`. Wir schreiben also:

8 *http://www.positioniseverything.net/explorer/inherited_margin.html*

```
/**
 * @section Content
 */
div#content {
  margin: 0 0 0 288px;
  padding: 0 12px 0 36px;
  height: 1%;
}
```

Und das Problem ist umgangen.

Nun müssen wir noch ein weiteres Problem lösen: Die Tabellenzeilen sind im IE 6 nicht eingefärbt. Das liegt daran, dass dieser Browser mit der Deklaration von `background-color` für `tr` nichts anzufangen weiß. Allerdings können Tabellenzellen einen Hintergrund haben. Wir ändern die Regeln wie folgt:

```
/* Tabellen in Formularen */
form table {
  width: 100%;
  border-collapse: collapse;
  margin: .8em 0;
}
form tr {
  color: inherit;
  background-color: #e8e7ee;
}
form tr:nth-child(odd) {
  color: inherit;
  background-color: #f5f4f8;
}
form td,
form th {
  color: #5e2021;
  background-color: transparent;
  border-bottom: 4px solid white;
  padding: 5px 5px 5px 15px;
}
form th {
  width: 40%;
  text-align: left;
}
```

Jetzt macht auch der Internet Explorer alles richtig.

Nach diesem Schock nehmen wir nun allen Mut zusammen und schauen uns das Rezept des Tages an – die Seite ist jedoch einwandfrei.

Damit haben wir die Kochbar-Website im IE 6 getestet und alle Probleme beseitigt. Machen wir nun dasselbe im IE 7 und im IE 8[9]. Ich kann keine Probleme oder wesentlichen Unterschiede feststellen. Sehen Sie etwas?

9 Die Kochbar wurde mit der zweiten Betaversion des IE 8 getestet.

Auch in folgenden Browsern präsentiert sich die Kochbar-Website vollkommen fehlerfrei:

- Firefox 2.0.0.17
- Opera 8.54
- Opera 9.60 (der sogar die Tabellenzeilen im Kontaktformular unterschiedlich einfärbt)
- Konqueror 4.1.2

Im Safari 3.1.2 für Macintosh und Windows sieht die Schrift geringfügig kleiner aus als in den anderen Browsern, auch stimmen einige Abstände nicht exakt überein. Es handelt sich jeweils um Kleinigkeiten, die nur auffallen, wenn man die Darstellung mit der in anderen Browsern vergleicht. Wir brauchen nicht aktiv zu werden, sondern lassen alles, wie es ist.

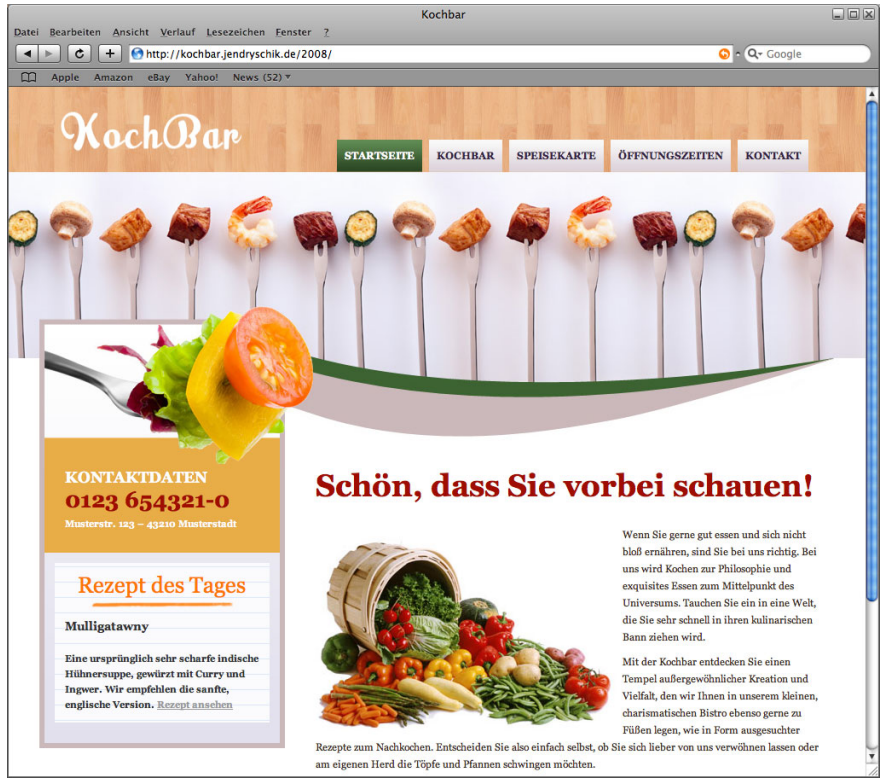


Abbildung 8.44: Im Safari können wir geringfügige Unterschiede feststellen, wenn wir ganz genau hinsehen

Wie sieht die Seite eigentlich mit Lynx aus? Abbildung 8.45 zeigt uns, dass wir zufrieden sein können. Alles ist übersichtlich formatiert, wir erkennen die Navigation, Überschriften, Links; auch die Unterseiten sehen sehr übersichtlich aus – saubere Arbeit!

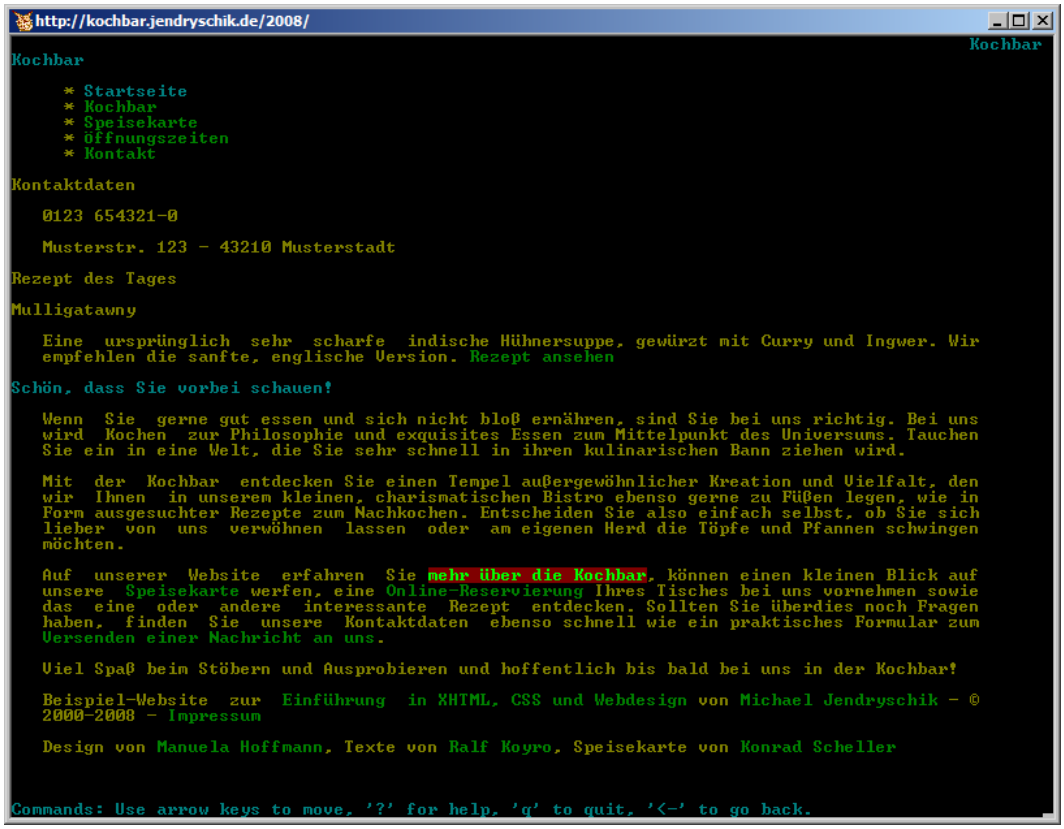


Abbildung 8.45: Die Kochbar im Textbrowser Lynx

8.22 Das Ergebnis

Die Kochbar ist fertig, und ich hoffe, dass Sie vieles von dem anwenden konnten, was Sie bei der Lektüre dieses Buchs gelernt haben. Ich habe versucht, Ihnen in diesem Buch und ganz besonders im letzten Kapitel zu zeigen, was ich unter der Trennung von Markup und Layout verstehe. Sauberes XHTML in Verbindung mit modernem CSS ist die Basis jeder gelungenen Website. Wenn Sie das Buch gewissenhaft durchgearbeitet haben, dann haben Sie alles gelernt, was Sie wissen müssen, um ein guter Webautor zu werden.

Ich ermutige Sie hiermit ausdrücklich, das erworbene Wissen in der Praxis anzuwenden. Verwenden Sie die hier vorgestellten Lösungen bei Ihren eigenen Projekten und wagen Sie den Blick über den Tellerrand. Lesen Sie viel und bleiben Sie am Ball, wenn es um neue Entwicklungen geht. Es gibt viele gute Bücher auf dem Markt und viele Weblogs, deren tägliche Lektüre Sie Schritt für Schritt voranbringen wird. Helfen Sie mit, dass das Web ein Ort wird, an dem wir uns wohlfühlen können! Ich wünsche Ihnen viel Erfolg dabei.

9 Anhang

9.1 Glossar

Siehe ***Zugänglichkeit***.

Accessibility

Anführungszeichen

Dienen der visuellen Auszeichnung von direkter Rede und Zitaten sowie der Hervorhebung von Wortteilen, Wörtern oder Satzteilen. Im deutschen Schriftsatz werden zwei Varianten verwendet: doppelte normale Anführungszeichen, bestehend aus einem doppelten unteren linken Anführungszeichen (U+201E) und einem doppelten oberen rechten Anführungszeichen (U+201C) („foo“), sowie doppelte spitze Anführungszeichen, bestehend aus einem doppelten spitzen Anführungszeichen nach rechts (U+00BB) und einem doppelten spitzen Anführungszeichen nach links (U+00AB) (»foo«). Dazu kommen jeweils die sogenannten halben Anführungszeichen (‚foo‘ und ›foo‹) für Eigennamen und Begriffsdefinitionen sowie verschachtelte Zitate. Sie stehen ohne Zwischenraum vor und nach dem eingeschlossenen Textabschnitt. Die Zeichen, die Sie über ⇧+2 sowie ⇧+# erhalten, nämlich die Zeichen " und ', haben keine typografische Bedeutung.

Anker

Quelle und/oder Ziel eines Hyperlinks. In HTML und XHTML unter anderem durch das Element `a` repräsentiert.

Siehe ***Hyperlink***, ***HTML***, ***XHTML***, ***Element***.

ANSI (American National Standards Institute) US-amerikanisches Institut zur Normung industrieller Verfahrensweisen. Mitglied in der ISO. Deutsches Gegenstück ist DIN. (*http://www.ansi.org*)

Siehe ***ISO***, ***DIN***.

Apache HTTP Server Ein Open-Source-Produkt der Apache Software Foundation und der heute am weitesten verbreitete Webserver.

Siehe ***Open Source***, ***Webserver***.

Arbeitsentwurf Siehe ***W3C-Arbeitsentwurf***.

Ariadne-Faden (auch: Brotkrumen-Navigation) Der griechischen Mythologie zufolge ein Geschenk der Prinzessin Ariadne an Theseus. Mithilfe des Fadens fand Theseus den Weg durch das Labyrinth, in dem sich der Minotaurus befand. Bezeichnung für eine Art der Navigation auf Websites, die die Hierarchie widerspiegelt und dem Benutzer so eine zusätzliche Orientierungshilfe bietet.

Siehe ***Website***, ***Benutzer***.

ASCII (American Standard Code for Information Interchange) Zeichensatz, der auf dem lateinischen Alphabet basiert, wie er im modernen Englisch und von Computern zur Darstellung von Text verwendet wird. Wird seit Ende der 1960er verwendet und ist eine 7-Bit-Codierung, benutzt also 128 Zeichen.

Siehe ***Zeichensatz***.

At-Regel Grundkonstruktion eines Stylesheets. Dient unter anderem der Angabe, für welches Ausgabemedium Regeln gelten sollen, dem Import zusätzlicher Stylesheets oder der Angabe der aktuellen Zeichenkodierung. Beginnt mit einem `@`-Zeichen, unmittelbar gefolgt von einem Bezeichner (z. B. `@import`, `@page`), und umfasst alles inklusive des nächsten Semikolons oder Deklarationsblocks.

Siehe ***Stylesheet***, ***Regel***, ***Zeichenkodierung***, ***Deklarationsblock***.

Attribut Paar aus einem Namen und einem (Text-)Wert, der einem Element zugeordnet wird.

Siehe ***Element***.

Bar Siehe ***Foo***.

Barrierefreiheit Der Begriff Barrierefreiheit kommt ursprünglich aus der Behindertenbewegung und bezeichnet bestimmte Qualitäten gestalteter Lebensbereiche, wodurch diese für Menschen mit Behinderung nutzbar sind. Die gesetzliche Definition lautet: »Barrierefrei sind (...) technische Gebrauchsgegenstände, Systeme der Informationsverarbeitung, akustische und visuelle Informationsquellen und Kommunikationseinrichtungen (...), wenn sie für behinderte Men-

schen in der allgemein üblichen Weise, ohne besondere Erschwernis und grundsätzlich ohne fremde Hilfe zugänglich und nutzbar sind.« ***[BGG 2002]***

Siehe ***Zugänglichkeit***.

Benutzbarkeit

Siehe ***Web-Ergonomie***.

Benutzer

Ein Mensch, der mit einem Benutzerprogramm arbeitet, um eine Ressource anzuzeigen, anzuhören oder anderweitig zu nutzen.

Siehe ***Benutzerprogramm***, ***Ressource***.

Benutzerprogramm (auch: User Agent)

Ein beliebiges Programm, das eine Ressource ausliest und weiterverarbeitet. Ein Benutzerprogramm kann eine Ressource anzeigen, sie vorlesen, ihren Ausdruck veranlassen oder sie in ein anderes Format umwandeln. In den meisten Fällen ist ein einfacher Webbrowser gemeint.

Siehe ***Ressource***, ***Webbrowser***.

BGG (Behindertengleichstellungsgesetz)

Ziel des im Februar 2002 verabschiedeten Behindertengleichstellungsgesetzes ist es, »die Benachteiligung von behinderten Menschen zu beseitigen und zu verhindern sowie die gleichberechtigte Teilhabe von behinderten Menschen am Leben in der Gesellschaft zu gewährleisten und ihnen eine selbstbestimmte Lebensführung zu ermöglichen«. ***[BGG 2002]***

Bit

Kunstwort aus binary digit, englisch für Binärziffer. Die Informationsmenge 1 Bit gibt an, welche von zwei möglichen Zuständen zutrifft, beispielsweise die Stellung eines Schalters.

BITV (Barrierefreie Informationstechnik-Verordnung)

Stellt Anforderungen und Bedingungen auf, die behinderten Menschen, denen die Nutzung der Informationstechnik üblicherweise nur eingeschränkt möglich ist, den Zugang dazu zu eröffnen. Grundlage des barrierefreien Webdesigns in Deutschland und die verbindliche Rechtsverordnung zu § 11 Behindertengleichstellungsgesetz. Der für Webentwickler relevante Teil der BITV versteckt sich in Anlage 1, die die Anforderungen und Bedingungen aufführt. ***[BITV 2002]***

Blocksatz

Im Druck übliche Methode, einen Text so zu setzen, dass die Zeilen durch Erweiterung der Wortzwischenräume auf die gleiche Breite gebracht werden. Für die Verwendung im World Wide Web nicht geeignet, da Worttrennung in Webbrowsern nicht ohne Weiteres möglich ist und dadurch sehr große Wortzwischenräume entstehen können.

Siehe ***World Wide Web***, ***Webbrowser***.

Braillezeile

Spezielles Ausgabegerät, das über ein taktiles Display mit in den meisten Fällen acht beweglichen Stiften Punktemuster entsprechend dem Braille-Alphabet ausgibt. Auf diese Weise werden sehbehinderte Benutzer in die Lage

versetzt, selbstständig am Computer zu arbeiten. Eine Braillezeile wird oftmals zusammen mit einem Screenreader verwendet.

Siehe ***Benutzer, Screenreader***.

Brotkrumen-Navigation Siehe ***Ariadne-Faden***.

Browser Siehe ***Webbrowser***.

Browserstylesheet Wird von Webbrowsern vor allen anderen Stylesheets für ein Dokument angewendet. Es sollte die Elemente so präsentieren, dass sie den allgemeinen Darstellungserwartungen für die Dokumentsprache entsprechen (z. B. wird ein Element `em` in HTML bei visuellen Browsern unter Verwendung einer kursiven Schrift dargestellt). Kann durch Stylesheets des Benutzers oder des Webautors überschrieben werden.

Siehe ***Webbrowser, Stylesheet, Dokument, Element, HTML, Benutzer, Webautor***.

Bug Begriff für Softwarefehler.

Bugfix Bezeichnung für das Beheben oder einen Patch zum Beheben eines Softwarefehlers.

Siehe ***Bug***.

Camino Ein auf Mozilla basierender Webbrowser für Mac OS X. Vormals unter dem Namen Chimera bekannt.

Siehe ***Mozilla, Webbrowser***.

CDATA-Abschnitt (Character Data) Wird in XML-Dokumenten dazu verwendet, dem Parser mitzuteilen, dass kein Markup folgt, sondern normaler Text. Enthaltene Markup-Zeichen werden vom Parser ignoriert.

Siehe ***XML, Dokument, Parser, Textauszeichnung, Zeichen***.

CERN (Conseil Européen pour la Recherche Nucléaire) Forschungszentrum für Teilchenphysik in der Nähe von Genf. Tim Berners-Lee erfand während seiner Arbeit am CERN das World Wide Web. (*http://www.cern.ch*)

Siehe ***World Wide Web***.

Client Programm, das den Dienst eines anderen Programms nutzt, z. B. ein Webbrowser, ein Editor oder ein Suchroboter.

Siehe ***Server, Webbrowser, Suchroboter***.

Content-Management-System

Softwaresystem für das Administrieren von Inhalten mit Unterstützung des Erstellungsprozesses basierend auf der Trennung von Inhalten und Struktur. Content-Management-Systeme verwenden die Mechanismen des Dokumentenmanagements.

Siehe ***World Wide Web***, ***Dokument***.

Crawler

Siehe ***Suchroboter***.

CSS (Cascading Style Sheets)

Formatierungssprache, die es Webautoren und Benutzern erlaubt, Formatierungen (zum Beispiel Schriften, Abstände und akustische Aspekte) von strukturierten Dokumenten durchzuführen. CSS vereinfacht mit der Trennung des Dokumentinhalts von dessen Präsentation die Erfassung von Webdokumenten und die Verwaltung von Websites. ***[REC-CSS10 1999] [REC-CSS20 1998] [WD-CSS21 2006]***

Siehe ***Webautor***, ***Benutzer***, ***Dokument***, ***HTML***, ***XML***, ***World Wide Web***, ***Website***.

Deklaration

Bestandteil eines CSS-Deklarationsblocks. Besteht aus einer Eigenschaft, gefolgt von einem Doppelpunkt, gefolgt von einem Wert. Vor und hinter diesen Bestandteilen kann sich Whitespace befinden. Mehrere Deklarationen werden durch Semikola voneinander getrennt.

Siehe ***CSS***, ***Deklarationsblock***, ***Eigenschaft***, ***Whitespace***.

Deklarationsblock

Bestandteil einer CSS-Regel. Beginnt mit einer öffnenden geschweiften Klammer und endet mit der zugehörigen schließenden geschweiften Klammer. Innerhalb der geschweiften Klammern dürfen beliebige Zeichen stehen. Klammern, eckige Klammern und geschweifte Klammern müssen immer paarweise auftreten und dürfen verschachtelt werden. Einfache und doppelte Anführungszeichen müssen ebenfalls paarweise auftreten. Üblicherweise enthält der Deklarationsblock eine oder mehrere Deklarationen.

Siehe ***CSS***, ***Regel***, ***Zeichen***, ***Deklaration***.

Deprecated

Ein Element oder Attribut einer Auszeichnungssprache gilt als deprecated (missbilligt, veraltet), wenn es durch neuere Konstrukte oder Technologien überholt wurde. Missbilligte (X)HTML-Elemente und -Attribute sind nur noch in der Transitional-DTD enthalten, nicht jedoch in der Strict-DTD.

Siehe ***Element***, ***Attribut***, ***HTML***, ***XHTML***, ***DTD***.

DIN (Deutsches Institut für Normung e.V.)

Die für die Normungsarbeit in Deutschland zuständige Institution. Vertritt die deutschen Interessen in den weltweiten und europäischen Normungsorganisationen. Mitglied der ISO. Amerikanisches Gegenstück ist das ANSI. (*http://www.din.de*)

Siehe ***ISO***, ***ANSI***.

Doctype Siehe ***Dokumenttyp-Deklaration***.

Doctype Switching Verfahren zur Bestimmung des Darstellungsmodus eines (X)HTML-Dokuments in aktuellen Webbrowsern in Abhängigkeit von der Art und der Notation der verwendeten Dokumenttyp-Definition. Im Quirks Mode wird eine Webseite wie durch alte, inkompatible Browser dargestellt; der Standards Compliance Mode bemüht sich um eine Darstellung gemäß den W3C-Spezifikationen. Aktuelle Gecko- und Opera-Versionen unterteilen den Standards Compliance Mode in einen Almost Standards Mode und einen Full Standards Mode.

Siehe ***HTML***, ***XHTML***, ***Dokument***, ***Webbrowser***, ***Dokumenttyp-Definition***, ***W3C-Spezifikation***, ***Gecko***, ***Opera***.

Dokument Bezeichnet in der elektronischen Datenverarbeitung eine nicht ausführbare Datei mit in der Regel durch Menschen lesbarem Inhalt, z. B. HTML-Dokument, Word-Dokument. Besteht aus Inhalt, Struktur, Informationen zur visuellen Darstellung und semantischen Informationen.

Siehe ***HTML***, ***Semantik***.

Dokumenttyp Auch: Ausprägung, Derivat, Sprache; konkrete Instanz einer Auszeichnungssprache. SGML-Dokumenttypen, z. B. HTML oder DocBook, werden durch eine DTD definiert, XML-Dokumenttypen, z. B. XHTML oder SVG, durch eine DTD oder ein XML Schema.

Siehe ***SGML***, ***HTML***, ***DTD***, ***XML***, ***XHTML***, ***SVG***, ***XML Schema***.

Dokumenttyp-Definition Siehe ***DTD***.

Dokumenttyp-Deklaration Benennt die Dokumenttyp-Definition, die für das Dokument verwendet wird. Teilt der verarbeitenden Software mit, um welchen Dokumenttyp es sich handelt.

Siehe ***DTD***, ***Dokument***, ***Dokumenttyp***.

DOM (Document Object Model) W3C-Empfehlung, die Programmen eine allgemeine Vorgehensweise für den Zugriff auf Bestandteile eines Dokuments liefert.

Siehe ***W3C-Empfehlung***, ***Dokument***.

DTD (Document Type Definition) In SGML ein Metadokument, das Informationen darüber enthält, wie ein Satz von SGML-Elementtypen verwendet werden kann. In XML wird diese Rolle häufig von XML Schema übernommen.

Siehe ***SGML***, ***Meta***, ***Dokument***, ***Elementtyp***, ***XML***, ***XML Schema***.

Eigenschaft Der Bestandteil einer CSS-Deklaration, dem ein Wert zugeordnet wird.

Siehe ***CSS***, ***Deklaration***.

Element

Konkrete Ausprägung eines Elementtyps, bestehend aus Leeres-Element-Tag oder Start-Tag und gegebenenfalls End-Tag sowie Inhalt.

Siehe ***Elementtyp***, ***Leeres-Element-Tag***, ***Start-Tag***, ***End-Tag***.

Elementtyp

Innerhalb der Dokumenttyp-Definition oder des XML Schemas, das den SGML- oder XML-Dokumenttyp definiert, festgelegtes syntaktisches Konstrukt, das Struktur oder gewünschtes Verhalten repräsentiert. Sowohl HTML als auch XHTML bestehen abhängig von der gewählten Variante aus bis zu 91 Elementtypen, beispielsweise Absätze (`p`), Tabellen (`table`) oder Anker (`a`). Eine konkrete Ausprägung eines Elementtyps heißt Element.

Siehe ***Dokumenttyp-Definition***, ***XML Schema***, ***SGML***, ***XML***, ***Dokumenttyp***, ***HTML***, ***XHTML***, ***Element***.

Empfehlung

Siehe ***W3C-Empfehlung***.

End-Tag (auch: Schluss-Tag, Ende-Tag)

Schließt die Notation eines Elements ab. Besteht aus öffnender spitzer Klammer, dem Ende-Kennzeichen `/`, dem Elementnamen und schließender spitzer Klammer. Beispiel: `</foo>`.

Siehe ***Element***.

Ergonomie

Wissenschaft von der Verbesserung der Schnittstelle zwischen Benutzer (Mensch) und Objekt (Maschine).

Siehe ***Benutzer***, ***Web-Ergonomie***.

Firefox

Freier Webbrowser, der wie Mozilla die Gecko Rendering Engine zur Darstellung von HTML- und XHTML-Dokumenten und den XML-Dokumenttyp XUL (XML User-interface Language) zur Gestaltung der **GUI** verwendet. Wurde im Jahr 2004 durch über Spenden finanzierte, großformatige Anzeigen in der New York Times und der Frankfurter Allgemeinen Zeitung auch außerhalb der Webgemeinde bekannt und verbreitet sich seitdem enorm.

Siehe ***Webbrowser***, ***Mozilla***, ***Gecko***, ***Rendering Engine***, ***HTML***, ***XHTML***, ***Dokument***, ***XML***, ***Dokumenttyp***, ***GUI***, ***World Wide Web***.

Flash

Auf Vektorgrafiken basierendes Grafik- und Animationsformat der amerikanischen Firma Adobe (früher: Macromedia), das heutzutage auf vielen Websites eingesetzt wird.

Siehe ***Website***.

Fokus

Man spricht davon, dass ein Element den Fokus hat, wenn es Tastaturereignisse oder andere Arten der Texteingabe (z. B. Sprache) akzeptiert, beispielsweise ein Anker, den man über die [⇆]-Taste angesteuert, oder ein Texteingabefeld, in das man hineingeklickt hat.

Siehe ***Element***.

Foo Definitionsfreier Stellvertreterausdruck, das heißt Ausdruck für »irgendetwas«, »irgendeins«. Dient oft als Platzhalter in technischen Anleitungen oder Beispielen. Werden mehrere Ausdrücke gebraucht, ist die Reihenfolge foo, bar, baz, qux, quux üblich. Der Begriff geht auf US-amerikanische Soldaten des Zweiten Weltkriegs zurück. In Anlehnung an das deutsche Wort »furchtbar« benannten sie etwas, das völlig zerstört oder kaputt war, mit der Abkürzung FUBAR (Fucked Up Beyond All Repair), was im Laufe der Zeit als »foobar« in den amerikanischen Sprachgebrauch überging und auf diesem Weg in den deutschen Sprachraum zurückkehrte (vgl. Wikipedia).

Frames Ermöglichen es, eine Webseite in verschiedene Unterbereiche zu zerteilen und in jeden Bereich eine eigene Ressource zu laden. Das einzelne Segment wird als Frame bezeichnet, die Definition aller Frames als Frameset. Frames wurden von Netscape mit dem Netscape Navigator 2.0 eingeführt und durch das W3C im Rahmen von HTML 4 standardisiert, verlieren heute allerdings zunehmend an Bedeutung.

Siehe ***Webseite***, ***Ressource***, ***Netscape Navigator***, ***W3C***, ***HTML***.

FTP (File Transfer Protocol) Netzwerkprotokoll zur Übertragungen von Dateien vom Server zum Client, vom Client zum Server oder clientgesteuert zwischen zwei Servern.

Siehe ***Server***, ***Client***.

Gebrauchstauglichkeit Das Ausmaß, in dem ein Produkt (z. B. eine Website) durch bestimmte Benutzer in einem bestimmten Nutzungskontext verwendet werden kann, um bestimmte Ziele effektiv, effizient und zufriedenstellend zu erreichen.

Siehe ***Website***, ***Benutzer***.

Gecko Name einer Rendering Engine, die zur Darstellung von (X)HTML-Dokumenten genutzt wird. Wird mit besonderem Augenmerk auf die vollständige Unterstützung von offenen Standards wie HTML, CSS, DOM, XML und JavaScript entwickelt. Wird unter anderem von den Webbrowsern Netscape Navigator, Mozilla, Firefox, Camino und Konqueror verwendet.

Siehe ***Rendering Engine***, ***HTML***, ***XHTML***, ***Dokument***, ***CSS***, ***DOM***, ***XML***, ***JavaScript***, ***Webbrowser***, ***Netscape Navigator***, ***Mozilla***, ***Firefox***, ***Camino***, ***Konqueror***.

generic coding Bezeichnet das bereits in den 1960er-Jahren entwickelte Konzept der Trennung des Informationsgehalts eines Dokuments von dessen äußerer Form.

Siehe ***Dokument***.

generic markup

Begriff für das Konzept der Auszeichnung von Inhalten nach semantischen anstatt nach visuellen Gesichtspunkten. Geht auf den New Yorker Buchdesigner Stanley Rice zurück und fand mit der Entwicklung von GML die erste für die Entwicklung des World Wide Webs relevante Anwendung.

Siehe ***Semantik***, ***GML***, ***World Wide Web***.

GIF (Graphics Interchange Format)

Digitales Bildformat mit guter, verlustfreier Komprimierung für Bilder mit geringer Farbtiefe (bis zu 256 Farben). 1987 vom US-Onlinedienst CompuServe eingeführt. Vor allem wegen seiner effizienten Kompression LZW (Lempel-Ziv-Welch-Algorithmus) populär. Ermöglicht das Speichern mehrerer Bilder in einer einzigen Datei, was vor allem für einfache Animationen verwendet wird.

GML (Generic Markup Language)

Durch Charles Goldfarb, Edward Mosher und Raymond Lorie im Jahr 1969 entwickelte Auszeichnungssprache. Basiert auf den Konzepten generic coding und generic markup. Führte das Konzept eines formal definierten Dokumenttyps mit einer verschachtelten Elementstruktur ein. Grundstein für die Entwicklung von SGML.

Siehe ***generic coding***, ***generic markup***, ***Dokumenttyp***, ***Element***, ***SGML***.

GUI (Graphical User Interface)

Grafische Schnittstelle auf Computern, die eine Interaktion mit dem Benutzer verlangen, also derjenige Teil einer Software, den der Benutzer unmittelbar (vor sich auf dem Monitor beziehungsweise der Anzeige) sieht (vgl. Wikipedia).

Siehe ***Benutzer***.

Gültigkeit

Ein Dokument ist gültig (valide), wenn es erfolgreich gegen die zugrunde liegende DTD geprüft werden kann. Die Elemente, Attribute und Attributwerte dürfen nur so verwendet (und verschachtelt) werden, wie es für die gewählte Sprachversion definiert ist.

Siehe ***Dokument***, ***DTD***, ***Element***, ***Attribut***.

Hexadezimalsystem

Zahlensystem zur Basis 16. Benutzt die Ziffern 0 bis 9 sowie für die dezimalen Zahlen 10 bis 15 die Buchstaben A bis F. Oftmals wird die Hexadezimalschreibweise durch das Voranstellen von »x« (in HTML) oder »0x« (in Programmiersprachen wie C++ oder Java) kenntlich gemacht, damit eine Verwechslung mit der Dezimal- oder anderen Schreibweisen ausgeschlossen wird.

Siehe ***Programmiersprache***.

Homepage

Die erste Seite einer Website, also die Seite, die von einem Benutzer in der Regel als Erstes aufgerufen wird, zu der er immer wieder zurückkommt und die deutlich öfter betrachtet wird als irgendeine andere Seite. Sollte vermitteln,

welche Inhalte, Produkte oder Dienstleistungen auf der Website angeboten werden, was das Unternehmen macht, das die Website repräsentiert, und wodurch es sich von der Konkurrenz abhebt. Ziel ist die Beantwortung folgender Fragen: Wo bin ich? Was kann ich hier tun?

Siehe ***Website***, ***Benutzer***.

HTML (Hypertext Markup Language)

Bildet zusammen mit HTTP und URI die fundamentalen Mechanismen des World Wide Webs. Die Auszeichnungssprache HTML vereint die semantische Auszeichnung von Inhalten und Hypertext und ermöglicht Webautoren unter anderem, Dokumente mit Überschriften, Text, Tabellen, Auflistungen, Hyperlinks zu anderen Dokumenten und Ressourcen sowie anderen Arten von Auszeichnungen online zu veröffentlichen, Formulare zur Interaktion mit Benutzern zu entwerfen sowie Multimedia-Elemente wie Videos, Musik und Animationen, aber auch Applikationen direkt in ihre Dokumente einzufügen. Wurde von Tim Berners-Lee als SGML-Dokumenttyp entworfen, 1995 unter der Leitung der IETF erstmalig als Norm standardisiert und als RFC 1866 »Hypertext Markup Language – 2.0« veröffentlicht ***[RFC-1866 1995]***. Es folgten HTML 3.2 ***[REC-HTML32 1997]***, HTML 4.0 ***[REC-HTML40 1998]*** und HTML 4.01 ***[REC-HTML401 1999]***.

Siehe ***HTTP***, ***URI***, ***World Wide Web***, ***Semantik***, ***Hypertext***, ***Webautor***, ***Dokument***, ***Hyperlink***, ***Ressource***, ***SGML***, ***Dokumenttyp***, ***IETF***, ***Norm***, ***RFC***.

HTML-Kompatibilitätsrichtlinien

Eine in Anhang C der XHTML-Empfehlung aufgestellte Sammlung von Entwurfsrichtlinien für Webautoren, die möchten, dass ihre XHTML-Dokumente von vorhandenen HTML-Benutzerprogrammen wiedergegeben werden können.

Siehe ***XHTML***, ***W3C-Empfehlung***, ***Webautor***, ***Dokument***, ***HTML***, ***Benutzerprogramm***.

HTTP (Hypertext Transfer Protocol)

Im Jahr 1990 von Tim Berners-Lee entwickeltes Kommunikationsschema zum Austausch von Daten und noch heute das Standardprotokoll des World Wide Webs. HTTP 1.0 wurde im Mai 1996 veröffentlicht. Bereits im August desselben Jahres folgte HTTP 1.1. HTTP wird primär zur Anfrage und Übertragung von Ressourcen verwendet. Wird durch Erweiterung seiner Anfragemethoden, Header-Informationenen und Fehlercodes im Rahmen von HTTP 1.1 zunehmend zum Austausch beliebiger Daten zwischen Clients und Servern verwendet.

Siehe ***World Wide Web***, ***Ressource***, ***Client***, ***Server***.

Hyperlink

Ein Verweis von einem Quell- zu einem Zielanker, in den meisten Fällen von einem Dokument zu einem anderen (externer Link) oder von einer Stelle in einem Dokument zu einer anderen im selben Dokument (interner Link), der

über einen Computer effizient verfolgt werden kann. Ein Hyperlink ist die Verbindungseinheit in Hypertext.

Siehe ***Anker, Dokument, Effizienz, Hypertext***.

Hypertext

Als Hypertext bezeichnet man die nicht lineare Organisation heterogener Objekte, deren netzartige Struktur durch logische Verbindungen zwischen atomisierten Wissenseinheiten hergestellt wird. Oder einfacher: Hypertext ist die nicht lineare Präsentation von Inhalten. Die einzelnen Inhalte werden Knoten genannt. Über Verweise navigiert man von einem Knoten zum anderen. Den Begriff prägte der US-amerikanische Gesellschafts- und Computerwissenschaftler Ted Nelson im Jahr 1965 im Rahmen seines Hypertextsystems Xanadu.

Siehe ***Hyperlink, Xanadu***.

IANA (Internet Assigned Numbers Authority)

US-amerikanische Organisation, die die Vergabe von IP-Adressen, Top-Level-Domains und IP-Protokollnummern regelt. (*http://www.iana.com*)

IETF (Internet Engineering Task Force)

Offene, internationale Vereinigung von Netzwerktechnikern, Herstellern und Anwendern, organisiert in über 80 Arbeitsgruppen mit insgesamt mehr als 700 Mitgliedern. Erarbeitet Vorschläge zur Standardisierung des Internets. (*http://www.ietf.org*)

Internationalisierung

Gestattet und vereinfacht die Benutzung einer Webseite für verschiedenen Sprachen, Schriften und Kulturen.

Internet Explorer

Der heute mit Abstand meistgenutzte Webbrowser. Seit Windows 95 SR2 Bestandteil des Betriebssystems Microsoft Windows. Verliert aufgrund zahlreicher Sicherheitslöcher und zum Teil mangelhafter Unterstützung aktueller Standards immer mehr Marktanteile an Konkurrenzprodukte wie Mozilla, Opera oder vor allem Firefox.

Siehe ***Webbrowser, Mozilla, Opera, Firefox***.

ISO (International Organization for Standardization)

Internationale Vereinigung der Standardisierungsgremien von 148 Ländern. Verabschiedet internationale Standards in allen technischen Bereichen sowie Verfahrensstandards. Deutschland ist seit 1951 durch das DIN in der ISO vertreten. (*http://www.iso.org*)

Siehe ***DIN***.

ISO 8859

Definiert in zurzeit 15 verabschiedeten und einer verworfenen Teilnorm verschiedene 8-Bit-Zeichensätze für die Informationstechnik. Die ersten 128 Zeichen (0 bis 127) entsprechen dabei ASCII, die Positionen 128 bis 159 sind für Steuerzeichen reserviert, die Positionen 160 bis 255 werden für Sonderzeichen

wie etwa Umlaute oder andere diakritische Zeichen verwendet, die in einer Vielzahl von Sprachen mit lateinischem Grundalphabet notwendig sind.

Siehe ***Zeichensatz***, ***Zeichen***, ***ASCII***, ***Steuerzeichen***.

Java — Objektorientierte Programmiersprache, die unter anderem von Patrick Naughton, Mike Sheridan und James Gosling im Auftrag des amerikanischen Computerherstellers Sun Microsystems entworfen wurde. Wird noch heute weiterentwickelt. Man unterscheidet Java-Applets, die in einem Webbrowser ausgeführt werden, Java-Servlets, die auf einem Webserver ausgeführt werden, und Java-Anwendungen, die als eigenständige Programme ausgeführt werden (vgl. ***[Balzert 2004]***).

Siehe ***Programmiersprache***, ***Webserver***.

JavaScript — Clientseitige Skriptsprache, die dazu entwickelt wurde, Webseiten dynamisch zu gestalten. Mittels einer Schnittstelle zum DOM können Elemente manipuliert werden, nachdem das Dokument zum Client übertragen wurde.

Siehe ***Client***, ***Skriptsprache***, ***Webseite***, ***DOM***, ***Element***, ***Dokument***.

JAWS (Job Access With Speech) — Screenreader der Freedom Scientific GmbH (*http://www.freedomsci.de*), der eine Sprachtechnologie einsetzt, die auf der Grundlage des Windows-Betriebssystems den Einsatz aller gängigen Softwareanwendungen und den Zugang zum Internet möglich macht. Verwendet eine integrierte Sprachausgabe, um den aktuellen Bildschirminhalt wiederzugeben. Gleichzeitig kann die Ausgabe auf einer Braillezeile erfolgen.

Siehe ***Screenreader***, ***Braillezeile***.

JPEG (Joint Photographic Experts Group) — Eines der am weitesten verbreiteten Formate zur digitalen Speicherung von Bildern. Bietet ein standardisiertes verlustbehaftetes Kompressionsverfahren, den sogenannten Baseline Codec.

Kaskade — Designprinzip von CSS. Stylesheets können drei verschiedene Ursprünge haben: Webautor, Benutzer und Benutzerprogramm. Stylesheets aus diesen drei Ursprüngen überlappen sich im Gültigkeitsbereich und arbeiten gemäß der Kaskade zusammen. Dabei wird jeder Regel eine Spezifität zugeordnet. Gelten mehrere Regeln, setzt sich diejenige mit der höchsten Spezifität durch. Importierte Stylesheets sind ebenfalls kaskadenartig angeordnet, ihre Spezifität ist von ihrer Importreihenfolge abhängig.

Siehe ***CSS***, ***Stylesheet***, ***Webautor***, ***Benutzer***, ***Benutzerprogramm***, ***Regel***.

KHTML — Rendering Engine in einigen Browsern, unter anderem in Konqueror und Safari, die ein HTML-Dokument analysiert und dessen korrekte Darstellung

übernimmt. Sehr schnell in der Darstellung, aber bei fehlerhaftem HTML-Code weniger tolerant als Gecko.

Siehe ***Rendering Engine***, ***Browser***, ***Konqueror***, ***Safari***, ***Gecko***.

Kombinator

Verkettet mehrere einfache Selektoren miteinander, um zusätzliche Übereinstimmungsbeschränkungen zu definieren. Kombinatorzeichen sind Whitespace, das Größer-als-Zeichen, das Pluszeichen und die Tilde.

Siehe ***Selektor***, ***Whitespace***.

Konqueror

Dateimanager und Webbrowser der grafischen Betriebssystemoberfläche KDE. Agiert durch die Komponente KHTML als Webbrowser mit eingebauter Unterstützung unter anderem für JavaScript und CSS.

Siehe ***Webbrowser***, ***KHTML***, ***JavaScript***, ***CSS***.

Leeres Element

Element, dessen Inhaltsmodell leer ist. Wird in SGML-Dokumenten ausschließlich mit Start-Tag und in XML-Dokumenten hauptsächlich als Leeres-Element-Tag notiert.

Siehe ***Element***, ***SGML***, ***Dokument***, ***Start-Tag***, ***XML***, ***Leeres-Element-Tag***.

Leeres-Element-Tag

Notation eines leeren Elements in XML-Dokumenten. Das XHTML-Element `br` beispielsweise wird wie folgt notiert: `<br/>` oder `<br />`. Das Leerzeichen vor dem Ende-Kennzeichen dient der Kompatibilität zu HTML-Benutzerprogrammen.

Siehe ***Leeres Element***, ***XML***, ***Dokument***, ***XHTML***, ***Leerzeichen***, ***HTML***, ***Benutzerprogramm***.

Leerzeichen

Dient der Abgrenzung von Wörtern innerhalb eines fortlaufenden Texts. Man erhält ein Leerzeichen durch Betätigen der [ ].

Leerzeichen, geschützt

Entspricht in der Breite dem einfachen Leerzeichen. Bei einem geschützten Leerzeichen wird nicht umbrochen. In HTML und XHTML über die Zeichenreferenz ` ` zu notieren.

Siehe ***Leerzeichen***, ***HTML***, ***XHTML***, ***Zeichenreferenz***.

Lynx

Textbrowser für Unix- und Windows-Systeme, besonders optimiert auf den Einsatz an Terminals. Wurde um 1992 an der Universität von Kansas entwickelt, ist heute noch verbreitet und zählt damit zu den Webbrowsern mit der längsten Entwicklungsgeschichte.

Siehe ***Textbrowser***, ***Webbrowser***.

Markup

Siehe ***Textauszeichnung***.

MathML (Mathematical Markup Language) XML-Dokumenttyp zur Darstellung von Formeln und komplexen Ausdrücken. Inhalte werden in einer logischen Struktur unabhängig von ihrer grafischen Gestaltung abgelegt. Im April 1998 als W3C-Empfehlung veröffentlicht, seit Oktober 2003 in der aktuellen Version 2.0 verfügbar.

Siehe ***XML***, ***Dokumenttyp***, ***W3C-Empfehlung***.

Memex (Memory Extender) Im Jahr 1945 von dem US-amerikanischen Wissenschaftler Vannevar Bush entwickelte Vision eines Systems, das das Wissen eines bestimmten Gebiets elektronisch aufbereitet leicht zugänglich darstellen kann. Kann als maschinelle Unterstützung des assoziativen Denkens angesehen werden. Inspirierte Tim Berners-Lee bei der Entwicklung des World Wide Webs.

Siehe ***World Wide Web***.

Meta Ein Präfix, das anzeigt, dass etwas auf sich selbst angewendet wird. Ein Metadokument beispielsweise ist ein Dokument über ein Dokument.

Siehe ***Dokument***.

Metasprache Sprache, mit der eine andere Sprache erklärt oder definiert wird, mit der also die Regeln (die Grammatik) für diese Sprache festgelegt werden, z. B. SGML und XML.

Siehe ***SGML***, ***XML***.

MIME-Typ (Multimedia Internet Message Extensions) Wurde ursprünglich zur Angabe des Datentyps von E-Mail-Anhängen entwickelt, findet mittlerweile allerdings auch in weiteren Diensten Verwendung. Besteht aus der Angabe eines Medientyps und der Angabe eines Untertyps. Beide Angaben werden durch einen Schrägstrich (`/`) voneinander getrennt. HTML-Dokumente beispielsweise werden mit dem MIME-Typ `text/html` ausgeliefert.

Siehe ***HTML***, ***Dokument***.

Modularisierung von XHTML Im April 2001 veröffentlichte W3C-Empfehlung. Darin werden die in XHTML 1.0 verfügbaren Elementtypen in eine Sammlung von abstrakten Modulen aufgeteilt, die bestimmte Funktionstypen anbieten. Module können miteinander und mit anderen Modulen kombiniert werden, sodass darauf basierend sowohl erweiterte als auch abgespeckte Dokumenttypen entworfen werden können. ***[REC-ModXHTML 2001]***

Siehe ***W3C-Empfehlung***, ***XHTML***, ***Elementtyp***, ***Dokumenttyp***.

Mosaic Erster Webbrowser, der Grafiken direkt, ohne sie nachzuladen, auf einer Website anzeigen konnte. Wurde als Projekt der Software Group am NSCA von Eric Bina und Marc Andreessen entwickelt. Erste Versionen erschienen ab September 1993 und basierten auf einer auf Unix-Rechnern weit verbreiteten gra-

fischen Benutzeroberfläche, dem X Window System. Im Laufe des Jahres folgten Versionen für Windows und Mac. Ein Großteil des Mosaic-Entwicklerteams arbeitete später an der Entwicklung des Netscape Navigators.

Siehe ***Webbrowser, Website, NCSA, Netscape Navigator.***

Mozilla

Aus Webbrowser, E-Mail-Client und weiteren Werkzeugen bestehende freie Programmsammlung der Mozilla Foundation. Der Webbrowser basiert auf der Rendering Engine Gecko, die Programmoberfläche auf dem eigens entwickelten XML-Dokumenttyp XUL (XML User-interface Language).

Siehe ***Webbrowser, Client, Rendering Engine, Gecko, XML, Dokumenttyp.***

Multimedia

Der Begriff Multimedia bezeichnet die Kombination von Inhalten aus mehreren Medien, darunter Text, Grafik, Animationen, Audio und Video.

Namensraum

Dient der eindeutigen Zuordnung eines Objekts in einem bestimmten Kontext mit dem Ziel, Namenskonflikte zu vermeiden. XML unterstützt ebenfalls Namensräume, wobei das Präfix in den meisten Fällen durch einen Doppelpunkt vom lokalen Namen getrennt wird.

Siehe ***XML.***

NCSA (National Center for Supercomputing Applications)

Wurde 1986 an der amerikanischen Universität von Illinois als eines von ursprünglich fünf Zentren des National Science Foundation's Supercomputer Centers Program gegründet. Seit 1993 weltweit bekannt durch den ersten grafischen Webbrowser Mosaic und den NSCA httpd, einen Webserver und Vorläufer des Apache HTTP Server. (*http://www.ncsa.uiuc.edu*)

Siehe ***Webbrowser, Mosaic, Webserver, Apache HTTP Server.***

Netscape Navigator

Ursprünglich auf Mosaic basierender Webbrowser. Die ersten Versionen erschienen bereits 1994 und erweiterten das damals aktuelle HTML 2.0 unter anderem um Tabellen, Frames, Scripting-Funktionen und Multimedia-Elemente. Wurde an Privatanwender kostenlos weitergegeben und hatte dadurch maßgeblichen Anteil an der größer werdenden Popularität des World Wide Webs. Sein Marktanteil betrug bereits 1995 weit über 80 Prozent. Wurde durch Microsofts verstärkte Bemühungen, den Internet Explorer am Markt zu etablieren, Ende der 1990er zunehmend verdrängt und hat heute nahezu keine Bedeutung mehr. Markeninhaber von Netscape ist AOL. Aktuelle Versionen ab 6.0 beruhen auf Mozilla.

Siehe ***Mosaic, Webbrowser, HTML, Frames, World Wide Web, Internet Explorer, Mozilla.***

Newsgroup Teilbereich des Usenets, vergleichbar mit einem Schwarzen Brett, mit dessen Hilfe öffentlich über ein bestimmtes Thema diskutiert wird, ähnlich wie in einem Webforum.

Siehe ***Usenet***.

Norm Eine als verbindlich anerkannte Regel.

Open Source Bezeichnet frei verfügbare Software, deren Quelltext in einer für den Menschen lesbaren und verständlichen Form frei erhältlich ist. Die Software darf abhängig von der gewählten Lizenz beliebig verändert, kopiert, ohne Lizenzgebühren verbreitet und genutzt werden.

Opera Webbrowser des norwegischen Unternehmens Opera Software ASA. Begann 1994 als Forschungsprojekt des norwegischen Telekommunikationsunternehmens Telenor und gehört heute aufgrund der guten Implementierung allgemein anerkannter Standards und seiner weitreichenden Konfigurierbarkeit zu den beliebtesten Browseralternativen.

Siehe ***Webbrowser***.

Parser Programm, das entscheidet, ob eine Eingabe zur Sprache einer bestimmten Grammatik gehört. Während des Parsens erfolgt eine syntaktische Überprüfung der Eingangsdaten. Bei der Überprüfung wird in der Regel aus den Daten ein Ableitungsbaum aufgebaut, um die Daten anschließend weiterverarbeiten zu können (vgl. Wikipedia).

PDF (Portable Document Format) Dateiformat, das von der Firma Adobe Systems entwickelt und 1993 veröffentlicht wurde. PDF-Dokumente geben die mit dem Erstellungsprogramm erzeugte Ausgabe einschließlich aller Schriften, Farben und Grafiken exakt wieder und dienen daher dem plattformübergreifenden Austausch von Dokumenten. Zur Anzeige von PDF-Dokumenten wird ein entsprechendes Programm benötigt, zum Beispiel Adobe Reader oder GSview.

Siehe ***Dokument***.

PHP (PHP Hypertext Preprocessor) Sehr beliebte serverseitige Skriptsprache mit einer an die Programmiersprache C angelehnten Syntax, die hauptsächlich zur Erstellung dynamischer Websites verwendet wird. Zeichnet sich durch leichte Erlernbarkeit, breite Datenbankunterstützung sowie die Verfügbarkeit zahlreicher zusätzlicher Funktionsbibliotheken aus.

Siehe ***Skriptsprache***, ***Programmiersprache***, ***Website***.

Plug-in Bezeichnung für ein Softwareprogramm, das in ein anderes Softwareprodukt integriert wird. Im World Wide Web wird häufig das Adobe Reader-Plug-in der Firma Adobe oder der Macromedia Flash Player für diverse Webbrowser

verwendet. Auch die Anzeige von MathML oder SVG erfordert in den meisten Webbrowsern die Installation eines Plug-ins.

Siehe ***World Wide Web***, ***PDF***, ***Flash***, ***Webbrowser***, ***MathML***, ***SVG***.

PNG (Portable Network Graphics)

Dateiformat zur Speicherung von Bilddaten. Wurde als freier Ersatz für das ältere, proprietäre Format GIF entworfen und ist weniger komplex als TIFF. Die Daten werden verlustfrei komprimiert abgespeichert (im Gegensatz zum verlustbehafteten JPEG-Dateiformat).

Siehe ***Proprietär***, ***GIF***, ***TIFF***, ***JPEG***.

Programmieren

Bezeichnet die Tätigkeit, Software mithilfe von Programmiersprachen zu erstellen.

Siehe ***Programmiersprache***.

Programmiersprache

Eine Programmiersprache ist eine formale Sprache zur Notation von Computerprogrammen und vermittelt genaue Angaben zu einer Kette von internen Verarbeitungsschritten, beteiligten Daten und deren Struktur in Abhängigkeit von internen oder externen Ereignissen (vgl. Wikipedia). HTML, XHTML und CSS sind keine Programmiersprachen.

Siehe ***HTML***, ***XHTML***, ***CSS***.

Proprietär

Nicht standardkonforme Erweiterung bestehender Empfehlungen oder Normen.

Siehe ***W3C-Empfehlung***, ***Norm***.

Quelltext

Begriff für den für Menschen lesbaren, in einer Programmiersprache geschriebenen Text eines Computerprogramms. Im allgemeinen Sprachgebrauch auch Bezeichnung für beliebigen menschenlesbaren Code, der für die Verarbeitung durch Software erstellt wurde.

Siehe ***Programmiersprache***.

Regel (auch: Regelmenge, Regelsatz)

Grundkonstruktion eines CSS-Stylesheets. Besteht aus mindestens einem Selektor und einem Paar geschweifter Klammern, Deklarationsblock genannt, die eine Liste mit null oder mehr durch Semikola voneinander getrennten Deklarationen enthalten.

Siehe ***CSS***, ***Stylesheet***, ***Selektor***, ***Deklarationsblock***, ***Deklaration***.

Regulärer Ausdruck

Reguläre Ausdrücke sind eine Familie von kompakten, leistungsfähigen formalen Sprachen, mit denen sich (Unter-)Mengen von Zeichenketten beschreiben lassen. Werden von vielen Texteditoren und Programmen verwendet, um bestimmte Muster zu suchen und diese dann durch etwas anderes zu ersetzen oder eine Aktion auszuführen (vgl. Wikipedia).

Rendering Engine Komponente eines Programms, die zur Darstellung von Dokumenttypen oder Benutzeroberflächen verwendet wird. So benötigt auch jeder Webbrowser eine solche Komponente, um die reinen (X)HTML- und CSS-Dokumente einzulesen und sie auf dem Bildschirm als Grafik, Animation, formatierten Text etc. auszugeben, beispielsweise Gecko oder KHTML.

Siehe ***Dokumenttyp***, ***Webbrowser***, ***HTML***, ***XHTML***, ***CSS***, ***Dokument***, ***Gecko***, ***KHTML***.

Ressource Dokumente und Daten aller Art, die sich im World Wide Web über einen URI adressieren lassen.

Siehe ***Dokument***, ***World Wide Web***, ***URI***.

RFC (Requests for Comments) Reihe von technischen und organisatorischen Dokumenten zum Internet, die am 7. April 1969 begonnen wurde. Bei der ersten Veröffentlichung noch im ursprünglichen Wortsinn zur Diskussion gestellt, behalten RFCs auch dann ihren Namen, wenn sie sich durch allgemeine Akzeptanz und Gebrauch zum Standard entwickelt haben.

Safari Webbrowser der Firma Apple. Gehört zum Lieferumfang von Mac OS X ab Version 10.3 (Panther) und ersetzte den bis dahin mitgelieferten Microsoft Internet Explorer als Standardbrowser. Verwendet die Rendering Engine WebKit, die auf KHTML basiert.

Siehe ***Webbrowser***, ***Internet Explorer***, ***Rendering Engine***, ***KHTML***.

Screenreader Zumeist von sehbehinderten Benutzern verwendetes Programm, das Informationen für den Bildschirm abfängt und diese entweder in synthetischer Sprache, zum Beispiel über eine Sprachausgabe, ausgibt oder in einer Braillezeile darstellt. Die in Deutschland meistverwendeten Screenreader sind JAWS, Virgo und Windows Eyes (vgl. ***[Hellbusch 2005]***).

Siehe ***Benutzer***, ***Braillezeile***, ***JAWS***.

Selektor Bestandteil einer CSS-Regel. Existiert immer in Kombination mit einem Deklarationsblock und besteht aus allem bis zu der ersten öffnenden geschweiften Klammer des Deklarationsblocks. Kann ein Benutzerprogramm den Selektor nicht parsen (weil es sich dabei beispielsweise nicht um gültiges CSS handelt), muss er den Deklarationsblock ebenfalls ignorieren.

Siehe ***CSS***, ***Regel***, ***Deklarationsblock***, ***Benutzerprogramm***.

Semantik Teilgebiet der Linguistik, das sich mit der Analyse und Beschreibung der Bedeutung von Sprache und sprachlichen Ausdrücken befasst.

Server Ein Dienst, der einem Client Ressourcen zur Verfügung stellt.

Siehe ***Client***, ***Ressource***.

Metasprache, mit deren Hilfe man Auszeichnungssprachen definieren kann. Um ein gültiges Dokument in SGML zu beschreiben, braucht man eine DTD. Ein SGML-Dokument ist ein konkreter Dokumenttyp von SGML. Der mit Abstand bekannteste SGML-Dokumenttyp ist HTML. SGML wurde auf Basis von GML durch Arbeitsgruppen des ANSI und der ISO unter der Koordination von Charles Goldfarb entwickelt und im Jahr 1986 als ISO-Standard 8879 veröffentlicht.

SGML (Standard Generalized Markup Language)

Siehe ***Metasprache***, ***Dokument***, ***DTD***, ***Dokumenttyp***, ***HTML***, ***GML***, ***ANSI***, ***ISO***.

Programmiersprache, die die direkte Ausführung des Programmcodes ermöglicht. Programme, die in Skriptsprachen geschrieben sind, werden Scripts genannt. Es handelt sich immer um Textdateien oder Textfragmente, die mit einem normalen Editor bearbeitet werden können. JavaScript und PHP sind Skriptsprachen.

Skriptsprache

Siehe ***Programmiersprache***, ***JavaScript***, ***PHP***.

Vom W3C entwickelter XML-Dokumenttyp für zeitsynchronisierte, multimediale Inhalte. Ermöglicht die Einbindung und Steuerung von Multimedia-Elementen wie Audio, Video, Text und Grafik in Webseiten. SMIL-Dateien können mit Java-Applets und -Servlets oder CGI-Scripts verknüpft werden. Wird in MMS (Multimedia Messaging Service) verwendet.

SMIL (Synchronized Multimedia Markup Language)

Siehe ***W3C***, ***XML***, ***Dokumenttyp***, ***Java***.

Menschengerechte Gestaltung eines Software-Arbeitsplatzes, das heißt der Anwendungssoftware und der Arbeitsoberfläche (vgl. ***[Balzert 2004]***).

Software-Ergonomie

Siehe ***W3C-Spezifikation***.

Spezifikation

Leitet die Notation eines Elements ein. Besteht aus einer öffnenden spitzen Klammer (U+003C), dem Elementnamen und einer schließenden spitzen Klammer (U+003E). Beispiel: `<foo>`.

Start-Tag

Siehe ***Element***.

Zeichen eines Zeichensatzes, die keine darstellbaren Zeichen repräsentieren (z. B. Wagenrücklauf, Zeilenvorschub). Wurden ursprünglich zur Ansteuerung von Textausgabegeräten wie Textdruckern, Telegrammgeräten oder Fernschreibern verwendet. Ermöglichen es, Steuerungsbefehle für die Ausgabegeräte innerhalb des Zeichensatzes zu übertragen, anstatt die Steuerungsinformationen über ein anderes Protokoll zu übertragen (vgl. Wikipedia).

Steuerzeichen

Siehe ***Zeichen***, ***Zeichensatz***, ***Wagenrücklauf***, ***Zeilenvorschub***.

Dokument, das einem Benutzerprogramm beschreibt, wie sich Elemente eines strukturierten Dokuments (z. B. HTML oder XHTML) in eine bestimmte bildli-

Stylesheet

che, hörbare oder fühlbare Präsentation umwandeln lassen. Erlauben es, Inhalte unabhängig vom Ausgabegerät zu interpretieren, ohne den Inhalt zu verändern. CSS und XSL sind Stylesheetsprachen (vgl. Wikipedia).

Siehe ***Dokument***, ***Benutzerprogramm***, ***Element***, ***HTML***, ***XHTML***, ***CSS***, ***XSL***.

Suchmaschine

Programm zur Recherche von Dokumenten, die in einem Computer oder einem Computernetzwerk wie dem World Wide Web gespeichert sind. Liefert nach Eingabe eines Suchbegriffs eine sogenannte Trefferliste von Verweisen auf relevante Dokumente sowie zumeist einige Metainformationen. Die Datenbeschaffung einer Suchmaschine erfolgt automatisch über Suchroboter.

Siehe ***Dokument***, ***World Wide Web***, ***Meta***, ***Suchroboter***.

Suchroboter (auch: Spider, Webcrawler)

Programm, das vor allem von Suchmaschinen eingesetzt wird, um Inhalte von Websites automatisch einzulesen und zur Indexierung auszuwerten. Gelangt über Hyperlinks von einer Webseite zur anderen. Dabei werden alle bekannten Ressourcen gespeichert und der Reihe nach besucht. Neu gefundene Ressourcen werden der Liste aller Ressourcen hinzugefügt.

Siehe ***Suchmaschine***, ***Website***, ***Hyperlink***, ***Webseite***, ***Ressource***.

SVG (Scalable Vector Graphics)

XML-Dokumenttyp zur Beschreibung zweidimensionaler Vektorgrafiken. Seit 2001 W3C-Empfehlung. Unterstützt mittels SMIL Animationen. Manipulationen des SVG-DOM sind über eingebettete Funktionen wie beispielsweise JavaScript möglich. ***[REC-SVG 2003]***

Siehe ***XML***, ***Dokumenttyp***, ***W3C-Empfehlung***, ***SMIL***, ***DOM***, ***JavaScript***.

Syntax

Behandelt die Muster und Regeln, nach denen Wörter zu größeren funktionellen Einheiten wie Sätzen zusammengestellt und Beziehungen wie Teil – Ganzes, Abhängigkeit etc. zwischen diesen formuliert werden (Satzbau). In der Informatik wird eine formale Grammatik festgelegt, um bei einer formalen Sprache erlaubte Konstruktionen festzulegen (vgl. Wikipedia).

Siehe ***Programmiersprache***.

Systembezeichner

Bestandteil der Dokumenttyp-Deklaration. Gibt den URI der DTD an.

Siehe ***Dokumenttyp-Deklaration***, ***URI***, ***DTD***.

Tag

Bestandteil der Notation eines Elements.

Siehe ***Element***, ***Start-Tag***, ***End-Tag***, ***Leeres-Element-Tag***.

Textauszeichnung

Bezeichnete ursprünglich die handschriftlichen Anmerkungen eines Layouters am Rand eines Dokuments, die später im Satz berücksichtigt wurden. Auf diese Weise wurden das Seitenformat bestimmt, Seiteninhalte angeordnet sowie typografische Festlegungen getroffen. Das Markup beschränkte sich

damals fast ausschließlich auf visuelle Aspekte eines Texts. Mit dem Konzept des generic markup richtete sich der Fokus auf die semantische Auszeichnung von Inhalten, wie sie gerade bei SGML- und XML-Dokumenttypen üblich ist. Markierungen machen dabei eine Aussage über die Art der markierten Stelle.

Siehe ***Dokument***, ***generic markup***, ***SGML***, ***XML***, ***Dokumenttyp***.

Textbrowser

Bezeichnung für Webbrowser, die lediglich Text, also keine Bilder oder andere Multimedia-Elemente, darstellen können. Ermöglicht Webautoren einen Blick auf eine Website, wie sie von Screenreadern oder Suchrobotern gelesen wird. Einer der meistverwendeten Textbrowser ist Lynx.

Siehe ***Webbrowser***, ***Webautor***, ***Website***, ***Screenreader***, ***Suchroboter***, ***Lynx***.

Tidy

Als Open Source frei erhältliches Programm, das dazu dient, (X)HTML-Dokumente zu prüfen und auf Wunsch automatisch zu korrigieren und zu bereinigen. Konvertiert HTML in XHTML und verschönert Quelltext. Die TidyLib-Bibliothek kann auf verschiedene Weise in diverse andere Programme integriert werden.

Siehe ***Open Source***, ***HTML***, ***XHTML***, ***Dokument***.

Typografie

Das Gestalten mit Schrift, von Schrift sowie mittels Schrift, Bildern, Linien, Flächen und dem sogenannten typografischen Raum (einem optisch wirksamen Abstand) in einem visuellen Medium. Ziel ist es, die Intention eines Texts durch optimale Lesbarkeit der dargestellten Information bestmöglich zu übermitteln. Möglichkeiten der typografischen Gestaltung sind der Einsatz unterschiedlicher Schriftarten, Schriftgrößen und Schriftschnitte sowie die Wahl der Zeilenlänge, des Zeilenabstands, des Satzspiegels innerhalb des Papierformats und vieles, vieles mehr (vgl. Wikipedia).

Unicode

Internationaler Standard mit dem Ziel, für jedes grafische Zeichen aller bekannten Schriftkulturen und Zeichensysteme einen digitalen Code festzulegen. Enthält zu sämtlichen Zeichen bestehender Zeichensätze nach Industriestandards und nationalen sowie ISO-Normen eine eindeutige Entsprechung.

Siehe ***Zeichen***, ***Zeichensatz***, ***ISO***, ***Norm***.

URI (Uniform Resource Identifier)

Zeichenfolge, die zur Identifizierung einer abstrakten oder physikalischen Ressource dient. Wurde mit RFC 1630 im Juni 1992 von Tim Berners-Lee im Rahmen seines Projekts »World Wide Web« am CERN eingeführt. Aktuell ist RFC 3986 vom Januar 2005. Ein URI ist entweder ein URL, ein URN oder beides.

Siehe ***Ressource***, ***World Wide Web***, ***CERN***, ***URL***, ***URN***.

URL (Uniform Resource Locator) Unterart von URI, die eine Ressource über ihren primären Zugriffsmechanismus identifiziert. *http://example.com/* ist ein klassischer URL, auch *ftp://user:password@example.com/* für FTP und *mailto:webmaster@example.com* für die Angabe von E-Mail-Adressen.

Siehe ***URI***, ***Ressource***, ***FTP***.

URN (Uniform Resource Name) Unterart von URI, die eine Ressource mittels eines vorhandenen oder frei zu vergebenden Namens identifiziert und auf Lebensdauer fest mit der Ressource verbunden ist, ganz gleich, wo und wie diese erreichbar ist. URNs können heute nicht direkt aufgerufen werden. Vielmehr müssen sie erst in URLs übersetzt werden, die nicht unbedingt direkt die gesuchte Ressource bezeichnen müssen, sondern auch zu Metadokumenten oder Bezugsquellen führen können.

Siehe ***URI***, ***Ressource***, ***URL***, ***Meta***, ***Dokument***.

US-ASCII Siehe ***ASCII***.

Usability Siehe ***Web-Ergonomie***.

Usenet 1979 in den USA ins Leben gerufenes weltweites elektronisches Netzwerk aus Newsgroups, an dem jeder, der Zugang zu einem Newsserver hat, teilnehmen kann.

Siehe ***Newsgroup***.

User Agent Siehe ***Benutzerprogramm***.

UTF (Unicode Transformation Format) Beschreibt Methoden, ein Unicode-Zeichen auf eine Folge von Bytes abzubilden. UTF-8 beispielsweise ist eine 8-Bit-Zeichenkodierung, die alle Zeichen aus Unicode abbildet. Die Zeichen aus ASCII werden in UTF-8 in genau einem Byte kodiert, die nachfolgenden Zeichen in zwei, drei oder vier Bytes.

Siehe ***Unicode***.

Validator Freier Service, über den man Dokumente wie HTML- und XHTML-Dokumente bezüglich ihrer Konformität zu W3C-Empfehlungen und anderen Standards überprüfen kann. Der bekannteste Validator ist der W3C Markup Validation Service.

Siehe ***Dokument***, ***HTML***, ***XHTML***, ***W3C-Empfehlung***.

Validität Siehe ***Gültigkeit***.

W3C (World Wide Web Consortium) Im Jahr 1994 durch Tim Berners-Lee und Al Vezza gegründeter Verband verschiedener Interessengruppen, der Empfehlungen und Vorschläge für neue Webstandards erarbeitet. Diese Interessengruppen setzen sich aus IT-Unternehmen und Forschungsinstituten zusammen. Meist werden diese Empfeh-

lungen in kürzester Zeit zum Industriestandard. Obwohl es das W3C erst relativ kurze Zeit gibt, geht von ihm der größte Einfluss auf die Entwicklung des World Wide Webs aus. Das W3C definiert Standards und entwickelt Empfehlungen für neue Sprachen und Dokumenttypen. (*http://www.w3.org*)

Siehe ***World Wide Web***, ***W3C-Empfehlung***, ***Dokumenttyp***.

W3C-Arbeitsentwurf (auch: Working Draft)

Teil des Prozesses des W3C hin zu einer offiziellen W3C-Empfehlung. Status eines Dokuments, das das W3C zur Überprüfung durch Mitglieder, technische Organisationen und die interessierte Öffentlichkeit herausgegeben hat.

Siehe ***W3C***, ***W3C-Empfehlung***, ***Dokument***.

W3C-Empfehlung (auch: Recommendation)

Formaler Text des W3C, der die Syntax und Semantik oder die Implementierung einer Webtechnologie beschreibt, beispielsweise eines bestimmten Dokumenttyps. Entspricht einer offiziellen Norm eines Standardisierungsgremiums. Einige Empfehlungen sind allgemein anerkannt und haben quasi den Status einer Norm oder stellen die Grundlage einer solchen dar.

Siehe ***W3C***, ***World Wide Web***, ***Dokumenttyp***, ***Norm***, ***ISO***, ***DIN***, ***HTML***, ***XML***, ***CSS***.

W3C-Spezifikation

Umgangssprachliche Bezeichnung für eine formelle Veröffentlichung des W3C, beispielsweise für eine W3C-Empfehlung.

Siehe ***W3C***, ***W3C-Empfehlung***.

Wagenrücklauf

Name eines Steuerzeichens, das die Einfügemarke auf dem Computer an den Anfang der Zeile bewegt. Bezeichnet im ursprünglichen Sinn den Zeilenrücklauf der Walze einer Schreibmaschine. Die übliche Abkürzung ist CR (Carriage Return).

Siehe ***Steuerzeichen***.

WAI (Web Accessibility Initiative)

Eine Domäne des W3C, die versucht, die Benutzung des World Wide Webs auch für Menschen mit Behinderungen sicherzustellen. (*http://www.w3.org/WAI/*)

Siehe ***W3C***, ***World Wide Web***.

WCAG (Web Content Accessibility Guidelines)

Von der WAI im Mai 1999 herausgegebene Sammlung von Empfehlungen, die die Zugänglichkeit von Websites erhöhen sollen. Enthält im Wesentlichen 14 Richtlinien mit dazugehörigen Checkpunkten, die näher ausführen, wie diese Richtlinien einzuhalten sind. In der zweiten, überarbeiteten Fassung, die derzeit noch als Arbeitsentwurf vorliegt, wird ein grundlegend anderer Ansatz verfolgt. Nicht die korrekte Verwendung von HTML und XHTML sowie CSS steht im Vordergrund, stattdessen werden Prinzipien und Richtlinien aufgestellt, die unabhängig von heutigen und zukünftigen Technologien gelten. Die

Checkpunkte sind allgemeiner formuliert, sodass sie sich auf beliebige Technologien anwenden lassen. ***[REC-WCAG10 1999] [WD-WCAG20 2006]***

Siehe ***WAI***, ***Zugänglichkeit***, ***Website***, ***HTML***, ***XHTML***, ***CSS***.

Web-Ergonomie

Software-Ergonomie für Websites beziehungsweise Webanwendungen. Ziel ist die Entwicklung und Evaluierung gebrauchstauglicher Websites, die Benutzer zur Erreichung ihrer Arbeitsergebnisse befähigen und dabei ihre Belange im jeweiligen Nutzungskontext beachten (vgl. ***[Balzert 2004]***).

Siehe ***Software-Ergonomie***, ***Website***, ***Benutzer***.

Webautor

Eine Person, die Dokumente und gegebenenfalls zugehörige Stylesheets schreibt.

Siehe ***Dokument***, ***Stylesheet***.

Webbrowser

Ein spezielles Benutzerprogramm, das hauptsächlich zum Anzeigen von Webseiten über HTTP verwendet wird. Moderne Browser verfügen über eine ganze Palette weiterer Funktionen, z. B. FTP, E-Mail oder Usenet. Man unterscheidet zwischen grafischen und Textbrowsern. Den ersten Browser schuf Tim Berners-Lee 1990 im Rahmen der Entwicklung des World Wide Webs. Zu den meistverwendeten Browsern gehören heute Internet Explorer, Mozilla, Firefox, Opera, Safari und Konqueror.

Siehe ***Benutzerprogramm***, ***Webseite***, ***HTTP***, ***FTP***, ***Usenet***, ***Textbrowser***, ***World Wide Web***, ***Internet Explorer***, ***Netscape Navigator***, ***Mozilla***, ***Firefox***, ***Opera***, ***Safari***, ***Konqueror***.

Webseite

Dokument, das mit einem Benutzerprogramm von einem Webserver abgerufen werden kann. Bestandteil einer Website. Besteht zumeist aus strukturiertem Text und Multimedia-Elementen.

Siehe ***Dokument***, ***Benutzerprogramm***, ***Webserver***, ***Website***.

Webserver

Ein Server, der Ressourcen über HTTP bereitstellt, die sich über einen URI adressieren lassen.

Siehe ***Server***, ***Ressource***, ***HTTP***, ***URI***.

Website

Zusammenhängendes Onlineangebot eines Anbieters, das zumeist aus mehreren Webseiten besteht, die sich nicht zwangsläufig auf einem einzigen Server befinden beziehungsweise unter einer einzigen Domain erreichbar sein müssen. Die Startseite einer Website, also die Webseite, die von einem Benutzer in der Regel als Erstes aufgerufen wird, wird Homepage genannt. Eine Website zeichnet sich im Allgemeinen durch einheitliche grafische Gestaltung sowie Navigation aus.

Siehe ***Webseite***, ***Server***, ***Benutzer***, ***Homepage***.

Whitespace

In der Datenverarbeitung und Programmierung gebräuchliche allgemeine Bezeichnung für Zeichen in einem Text, die im Editor nicht sichtbar (weiß, leer) sind und dennoch Platz in Anspruch nehmen. Je nach Kontext werden verschiedene Zeichen als Whitespace angesehen, in HTML und XHTML sind es die Zeichen Leerzeichen, Tabulator, Wagenrücklauf und Zeilenvorschub (vgl. Wikipedia).

Siehe ***Programmieren***, ***Zeichen***, ***HTML***, ***XHTML***, ***Leerzeichen***, ***Wagenrücklauf***, ***Zeilenvorschub***.

Wohlgeformtheit

Ein Dokument ist wohlgeformt, wenn es nach den in Kapitel 2.1 der XML 1.0-Empfehlung definierten Regeln strukturiert ist. Kurz zusammengefasst, besagen die Regeln: Es existiert genau ein Wurzelelement; alle Elemente werden durch ein Start-Tag und ein End-Tag begrenzt oder bestehen aus einem Leeres-Element-Tag; Elemente sind paarweise ineinander verschachtelt (keine Überkreuzung); Attributwerte müssen immer in Anführungszeichen gesetzt werden; es gibt keine Attributminimierung.

Siehe ***Dokument***, ***XML***, ***Element***, ***Start-Tag***, ***End-Tag***, ***Leeres-Element-Tag***, ***Attribut***.

Workaround

Provisorische Lösung eines bekannten Problems, um, ohne den eigentlichen Fehler zu beheben, zum gewünschten Ergebnis zu gelangen. Häufig die einzige Möglichkeit, Softwarefehler zu umgehen, beispielsweise bekannte Schwächen der CSS-Implementierung in Webbrowsern (vgl. Wikipedia).

Siehe ***CSS***, ***Webbrowser***.

Working Draft

Siehe ***W3C-Arbeitsentwurf***.

World Wide Web

Das heute weltweit größte und populärste Hypertextsystem wurde Anfang der 1990er von Tim Berners-Lee am CERN als Dokumentationssystem entwickelt. Berners-Lee verband die Ideen generic coding, Hypertext und Internet miteinander und ließ sich von früheren Entwicklungen wie Ted Nelsons Xanadu oder Vannevar Bushs Memex inspirieren. Das World Wide Web basierte ursprünglich und im Grunde auch noch heute auf den drei Kernstandards HTTP, HTML und URI.

Siehe ***Hypertext***, ***CERN***, ***generic coding***, ***Xanadu***, ***Memex***, ***HTTP***, ***HTML***, ***URI***.

World Wide Web Consortium

Siehe ***W3C***.

Xanadu

In den 1960er-Jahren begründetes Hypertextprojekt des US-amerikanischen Gesellschafts- und Computerwissenschaftlers Ted Nelson. Das Ziel war die Bildung einer riesigen, universalen Bibliothek mit zahllosen miteinander ver-

netzten Dokumenten. Die einzelnen Hypertexteinheiten sollten nicht nur verwaltet und miteinander in Beziehung gesetzt werden, sondern auch Integrität bei extensivem Mehrbenutzerbetrieb, Rückverfolgung der Entstehungshistorie eines Dokuments sowie Schutz von Urheber-, Nutzungs- und Vervielfältigungsrechten sollten gewährleistet werden. Das Projekt wurde aufgrund seiner Komplexität kommerziell nicht erfolgreich, inspirierte Tim Berners-Lee allerdings zur Entwicklung des World Wide Webs.

Siehe ***Hypertext***, ***World Wide Web***.

XForms — W3C-Standard für Formulare zur Datenerfassung im World Wide Web. XForms wurde entworfen, um (X)HTML-Formulare zu ersetzen und mit der übrigen XML-Technologie zu verbinden. XForms ist Bestandteil des noch in der Entwicklung befindlichen XHTML 2.0.

Siehe ***W3C***, ***World Wide Web***, ***HTML***, ***XHTML***, ***XML***.

XHTML (Extensible Hypertext Markup Language) — So wie HTML ein SGML-Dokumenttyp ist, entstand XHTML 1.0 im Januar 2000 als Anwendung von XML. Im August 2002 erschien XHTML 1.0 in der zweiten Ausgabe, die einige Fehler korrigiert und die Errata zur ersten Ausgabe integriert hatte. Die Spezifikation beschreibt XHTML 1.0 als ersten Dokumenttyp einer Familie aktueller und zukünftiger XHTML-Dokumenttypen und -Module. Der Sprachumfang von HTML 4.01 hat sich in XHTML 1.0 nicht verändert, lediglich die Syntax wurde an die gegenüber SGML deutlich strengeren Regeln von XML angepasst. ***[REC-XHTML10 2002] [REC-XHTML11 2001]***

Siehe ***HTML***, ***SGML***, ***Dokumenttyp***, ***XML***, ***W3C-Spezifikation***.

XHTML Basic — Reduzierter XHTML-Dokumenttyp, der sich auf die Module für eine einfache Inhaltsdarstellung beschränkt, aber erweitert und ausgebaut werden kann, falls sein Sprachumfang für einen konkreten Fall nicht ausreicht.

Siehe ***XHTML***, ***Dokumenttyp***.

XLink (XML Linking Language) — XML-Dokumenttyp zur Realisierung von Strukturen, die Links beschreiben können, die den einfachen, unidirektionalen Hyperlinks des heutigen HTML und XHTML ähneln, sowie noch differenziertere Links. Seit Juni 2001 W3C-Empfehlung. ***[REC-XLink10 2001]***

Siehe ***XML***, ***Dokumenttyp***, ***Hyperlink***, ***HTML***, ***XHTML***, ***W3C-Empfehlung***.

XML (Extensible Markup Language) — Eine Teilmenge von SGML und wie diese eine Metasprache, mit deren Hilfe man Auszeichnungssprachen für Dokumente definieren kann. Wurde aus dem Bemühen heraus entwickelt, für das umfangreiche SGML einen Weg ins World Wide Web zu finden, und so entworfen, dass es eine einfache Implementierung und Zusammenarbeit sowohl mit SGML als auch mit HTML gewährleistet. Seit Februar 1998 eine W3C-Empfehlung. Die bekanntesten

XML-Dokumenttypen im Web sind XHTML, SVG und MathML. ***[REC-XML10 2004] [REC-XML11 2004]***

Siehe ***SGML***, ***Metasprache***, ***Dokument***, ***World Wide Web***, ***HTML***, ***W3C-Empfehlung***, ***Dokumenttyp***, ***XHTML***, ***SVG***, ***MathML***.

XML Schema

W3C-Empfehlung zum Definieren von XML-Dokumentstrukturen in Form eines eigenen XML-Dokumenttyps. Alternative zu DTD, allerdings wesentlich komplexer und umfangreicher.

Siehe ***W3C-Empfehlung***, ***XML***, ***Dokument***, ***Dokumenttyp***, ***DTD***.

XML-Deklaration

Erkennungszeichenfolge im Prolog eines XML-Dokuments. Bietet die Möglichkeit, XML-Version und Zeichenkodierung des Dokuments anzugeben.

Siehe ***XML***, ***Dokument***, ***Zeichenkodierung***.

XPath (XML Path Language)

Sprache zur Adressierung von Teilen eines XML-Dokuments.

Siehe ***XSL***, ***XML***, ***Dokument***.

XSL (Extensible Stylesheet Language)

Familie von Sprachen zur Erzeugung von Stylesheets für XML-Dokumente. Beinhaltet XSL-FO und XSLT.

Siehe ***Stylesheet***, ***XML***, ***Dokument***, ***XSL-FO***, ***XSLT***, ***XPath***.

XSL-FO (XSL Formatting Objects)

Bestandteil von XSL. XML-Dokumenttyp zur Beschreibung eines XML-Dokuments als Baum mit Formatierungsanweisungen und Stilangaben. Mithilfe von sogenannten FO-Prozessoren können Ausgabedokumente wie beispielsweise PDF oder ASCII-Text erzeugt werden.

Siehe ***XSL***, ***XML***, ***Dokumenttyp***, ***Dokument***, ***PDF***, ***ASCII***.

XSLT (XSL Transformations)

Bestandteil von XSL. Programmiersprache zur Transformation eines XML-Dokuments in ein anderes. Spezielle Prozessoren lesen das XSLT-Dokument ein und transformieren das Eingabedokument nach den angegebenen Regeln in ein Ausgabedokument.

Siehe ***XSL***, ***Programmiersprache***, ***XML***, ***Dokument***.

Zeichen

Kleine Komponente einer geschriebenen Sprache und Element eines Zeichenvorrats, unter anderem Buchstaben eines Alphabets, Ziffern oder andere grafische Symbole.

Siehe ***Zeichenvorrat***.

Zeichenkodierung

Zuordnung eines Bit-Musters zu einem Zeichensatz. In vielen Fällen nimmt man dafür die binäre Darstellung der natürlichen Zahl, die die Position eines Zeichens im Zeichensatz angibt.

Siehe ***Bit***, ***Zeichensatz***, ***Zeichen***.

Zeichenreferenz (auch: Entity) Dient der Eingabe von Zeichen, die nicht in dem gewählten Zeichenvorrat vorhanden sind, unter anderem in HTML- und XHTML-Dokumenten. Beginnt mit einem Ampersand und endet mit einem Semikolon (zum Beispiel ` `). Es werden numerische Zeichenreferenzen (numeric character reference) und benannte Zeichenreferenzen (character entitiy reference) unterschieden. Numerische Zeichenreferenzen können dezimal oder hexadezimal angegeben werden. Benannte Zeichenreferenzen verwenden eine mnemonische Zeichenkombination, die den Buchstaben beschreibt und sich mehr oder weniger leicht merken lässt.

Siehe ***Zeichen***, ***Zeichenvorrat***, ***HTML***, ***XHTML***.

Zeichensatz Geordneter, nummerierter Zeichenvorrat.

Siehe ***Zeichenvorrat***.

Zeichenvorrat Endliche, nicht leere, ungeordnete Menge aller Zeichen, die man zur Verfügung hat oder benutzen möchte.

Siehe ***Zeichen***.

Zeilenvorschub Name eines Steuerzeichens, das die Einfügemarke auf dem Computer in die nächste Zeile bewegt. Bezeichnet im ursprünglichen Sinn das Drehen der Walze bei einer Schreibmaschine. Die übliche Abkürzung ist LF (Line Feed).

Siehe ***Steuerzeichen***.

Zugänglichkeit Gestaltung einer Website, sodass sie Benutzern in so hohem Maße wie möglich zur Verfügung steht, auch wenn sie Einschränkungen irgendeiner Art (z. B. Sinnes- und/oder Körperbehinderungen) unterliegen.

Siehe ***Website***, ***Benutzer***.

9.2 Literaturverzeichnis

[Balzert 2004]
H. Balzert; »Webdesign & Web-Ergonomie«, Dortmund, W3L, 2004.

[Baron 2004]
D. Baron; »Mozilla's Quirks Mode«. URL: *https://developer.mozilla.org/en/Mozilla%27s_Quirks_Mode*

[BGG 2002]
Beauftragter der Bundesregierung für die Belange behinderter Menschen; »Gesetz zur Gleichstellung behinderter Menschen (Behindertengleichstellungsgesetz – BGG)«, 2002. URL: *http://www.behindertenbeauftragter.de/gesetzgebung/behindertengleichstellungsgesetz*

[Behme, Mintert 2000]
H. Behme, S. Mintert; »XML in der Praxis – Professionelles Web-Publishing mit der Extensible Markup Language«, 2. Auflage, München, Addison-Wesley Verlag, 2000.

[Berners-Lee 1999]
T. Berners-Lee; »Der Web-Report – Der Schöpfer des World Wide Webs über das grenzenlose Potential des Internets«, München, Verlagshaus Goethestraße GmbH & Co. KG, 1999.

[Birkenbihl 2005]
W3C (K. Birkenbihl); »Alle Möglichkeiten des WWW ausschöpfen«, 2005. URL: *http://www.w3.org/Consortium/Offices/Germany/info.html*

[BITV 2002]
»Verordnung zur Schaffung barrierefreier Informationstechnik nach dem Behindertengleichstellungsgesetz (Barrierefreie Informationstechnik-Verordnung – BITV)«, 2002. URL: *http://www.einfach-fuer-alle.de/artikel/bitv/*

[Bush 1945]
V. Bush; »As We May Think«, 1945. URL: *http://www.ps.uni-sb.de/~duchier/pub/vbush/vbush-all.shtml*
Deutsche, gekürzte Übersetzung: R. Winter; »As We May Think«, 1997. URL: *http://wwwcs.uni-paderborn.de/~winkler/bush_d.html*

[Caspers 2003]
T. Caspers; »Mikroskopischer Text im MS IE«, 2003. URL: *http://www.einfach-fuer-alle.de/artikel/ietext/*

[Caspers 2007]
T. Caspers; »Reine Formsache«, 2007. URL: *http://www.einfach-fuer-alle.de/artikel/barrierefreie-formulare-mit-html-css-und-javascript/*

[Chao, Rudel 2007]
I. Chao, C. Rudel; »Über hasLayout«, 2007. URL: *http://onhavinglayout.fwpf-webdesign.de/*

[Clark 2002]
J. Clark; »Building Accessible Websites«, New Riders Publishing, 2002.

[Connolly 2006]
Dan Connolly; »<title>: the most important element of a quality Web page«. URL: *http://www.w3.org/QA/Tips/good-titles*

[CR-WCAG20 2008]
W3C (B. Caldwell, M. Cooper, L Guarino Reid, G. Vanderheiden); »Web Content Accessibility Guidelines 2.0«, 2008. URL: *http://www.w3.org/TR/WCAG20/*

[Dever, Chalk 1984]
J. Dever, G. Chalk; »Einsamer Wolf, Band 1, Flucht aus dem Dunkel«, München, Wilhelm Goldmann Verlag, 1984.

[DN-WCAGIM 1999]
W3C (W. Chisholm, G. Vanderheiden, I. Jacobs); »DRAFT NOTE: Web Content Accessibility Guidelines Impact Matrix«, 1999. URL: *http://www.w3.org/WAI/GL/WCAG-impact-matrix*
Deutsche Übersetzung: F. Engstermann, R. Wallbruch; »ANMERKUNGS-ENTWURF: Zugänglichkeitsrichtlinien für Web-Inhalte, Beeinflussungsschema«, 2003. URL: *http://www.w3.org/2003/10/Notiz-WCAG-einflussschema.html*

[Dürst 2007]
W3C (M. Dürst); »Setting the HTTP charset parameter«, 2007. URL: *http://www.w3.org/International/O-HTTP-charset*

[ECMA-262 1999]
ECMA Standardizing Information and Communication Systems; »ECMA-Script Language Specification«, 1999. URL: *http://www.ecma-international.org/publications/standards/Ecma-262.htm*

[EfA 2003]
Einfach für Alle; »Gehörlose können doch lesen...?«, 2003. URL: *http://www.einfach-fuer-alle.de/artikel/gehoerlos/*

[EN ISO 9241-110 2006]
»Ergonomie der Mensch-System-Interaktion – Teil 110: Grundsätze der Dialoggestaltung«, Berlin, Beuth-Verlag, 2006.

[Fischer, Lischke 2003]
W3C (H. Fischer, P. Lischke); »Das World Wide Web Consortium (W3C) in sieben Punkten«, 2003. URL: *http://www.w3.org/Consortium/Offices/Germany/sieben.html*

[Hellbusch]
J. E. Hellbusch; »Klare Formulare«. URL: *http://www.barrierefreies-webdesign.de/knowhow/formulare/*

[Hellbusch 2004]
J. E. Hellbusch; »Barrierefreies Webdesign«, Heidelberg, dpunkt.verlag, 2005.

[Hellbusch 2005]
J. E. Hellbusch; »Zett Punkt Bee Punkt«, 2005. URL: *http://www.einfach-fuer-alle.de/artikel/abkuerzungen/*

[ISO-8879 1986]
ISO International Organization for Standardization; »Information processing – Text and office systems – Standard Generalized Markup Language (SGML)«, 1986. URL: *http://www.iso.org/cate/d16387.html*

[Jendryschik 2006]
M. Jendryschik; »Instandsetzung einer Ruine – Internet Explorer zwischen 6 und 7«, iX Magazin für professionelle Informationstechnik, Ausgabe 10/2006, S. 131–135.

[Jendryschik 2008]
M. Jendryschik; »Kernsanierung – Internet Explorer zwischen 7 und 8«, iX Magazin für professionelle Informationstechnik, Ausgabe 05/2008, S. 52–54.

[Kuhlen 1991]
R. Kuhlen; »Hypertext – Ein nicht-lineares Medium zwischen Buch und Wissensbank«, Berlin, Heidelberg, New York, 1991.

[Korpela 2008]
J. Korpela; »"HTML validation" is a good tool, but just a tool«, 2008. URL: *http://www.cs.tut.fi/~jkorpela/html/validation.html*

[Langenberg]
K. Langenberg; »Tipps zum praxisgerechten Aufbau von Stylesheets«. URL: *http://www.thestyleworks.de/tut-art/style-practice.shtml*

[Meiert 2006-1]
J. Meiert; »10 Schritte zur hochwertigen Website«, 2006. ISSN 1614-3124, Ausgabe 25. URL: *http://meiert.com/de/publications/articles/20060523/*

[Meiert 2006-2]
J. Meiert; »XHTML- und CSS-Code-Richtlinien bei GMX«, 2006. ISSN 1614-3124, Ausgabe 21. URL: *http://meiert.com/de/publications/articles/20060326/*

[Meiert 2008]
J. Meiert; »Empfehlungen für gute ID- und Klassennamen«, 2008. ISSN 1614-3124, Ausgabe 39. URL: *http://meiert.com/de/publications/articles/20081021/*

[Meyer 2004]
E. Meyer; »Images, Tables, and Mysterious Gaps«, 2002. URL: *http://developer.mozilla.org/en/Images,_Tables,_and_Mysterious_Gaps*
Deutsche Übersetzung: D. Boecker; »Bilder, Tabellen und seltsame Lücken«, 2002. URL: *http://www.dodabo.de/html+css/img-table/*

[Meyer 2007]
E. Meyer; »Formal Weirdness«, 2007. URL: *http://meyerweb.com/eric/thoughts/2007/05/15/*
Deutsche Übersetzung: Webkrauts (E. Eggert); »Verrückte Formulare«, 2007. URL: *http://www.webkrauts.de/2007/05/31/verrueckte-formulare/*

[Münz 1997]
S. Münz; »Hypertext«, 1997. URL: *http://neue-medien.fh-joanneum.at/praktikum/hypertext/htxt.htm*

[Münz et al.]
S. Münz et al.; »SELFHTML: XML / Dokumenttyp-Definitionen (DTDs)«. URL: *http://de.selfhtml.org/xml/dtd/*

[Note-WCAG10-TECHS 2000]
W3C (W. Chisholm, G. Vanderheiden, I. Jacobs); »Techniques for Web Content Accessibility Guidelines 1.0«, 2000. URL: *http://www.w3.org/TR/WCAG10-TECHS/*

[Opera]
Opera Software ASA; »The Opera 9 DOCTYPE Switches«. URL: *http://www.opera.com/docs/specs/doctype/*

[Ott 2008]
S. Ott; »Impressumspflicht«, 2008. URL: *http://www.linksandlaw.info/*

[Päper 2005]
C. Päper; »Typographie für Webautoren«, 2005. URL: *http://webdesign.crissov.de/Typographie*

[Radtke, Charlier 2006]
A. Radtke, M. Charlier; »Barrierefreies Webdesign – Attraktive Websites zugänglich gestalten«, München, Addison-Wesley Verlag, 2006.

[REC-CSS10 1999]
W3C (H. W. Lie, B. Bos); »Cascading Style Sheets, level 1«, 1999. URL: *http://www.w3.org/TR/REC-CSS1*

[REC-CSS20 1998]
W3C (H. W. Lie, B. Bos, C. Lilley, I. Iacobs); »Cascading Style Sheets, level 2«, 1998. URL: *http://www.w3.org/TR/REC-CSS2/*

[REC-HTML32 1997]
W3C (D. Raggett); »HTML 3.2 Reference Specification«, 1997. URL: *http://www.w3.org/TR/REC-html32*

[REC-HTML40 1998]
W3C (D. Raggett, A. Le Hors, I. Jacobs); »HTML 4.0 Specification«, 1998. URL: *http://www.w3.org/TR/1998/REC-html40-19980424/*

[REC-HTML401 1999]
W3C (D. Raggett, A. Le Hors, I. Jacobs); »HTML 4.01 Specification«, 1999. URL: *http://www.w3.org/TR/REC-HTML40/*
Deutsche Übersetzung: edition W3C.de (C. Kühnel, S. Mintert, S. Schumacher); »HTML 4.01-Spezifikation«. URL: *http://edition-w3.de/TR/1999/REC-html401-19991224/*

[REC-ModXHTML 2001]
W3C (M. Altheim, F. Boumphrey, S. Dooley, S. McCarron, S. Schnitzenbaumer, T. Wugofski); »Modularization of XHTML«, 2001. URL: *http://www.w3.org/TR/xhtml-modularization/*
Deutsche Übersetzung: edition W3C.de (S. Mintert); »Modularisierung von XHTML«, 2003. URL: *http://www.edition-w3c.de/TR/xhtml-modularization*

[REC-PNG 2003]
W3C (D. Duce); »Portable Network Graphics (PNG) Specification (Second Edition) Information technology — Computer graphics and image processing — Portable Network Graphics (PNG): Functional specification. ISO/IEC 15948:2003 (E)«, 2003. URL: *http://www.w3.org/TR/PNG/*

[REC-SVG 2003]
W3C (J. Ferraiolo, “¡‘ò _˜, D. Jackson); »Scalable Vector Graphics (SVG) 1.1 Specification«, 2003. URL: *http://www.w3.org/TR/SVG11/*

[REC-WCAG10 1999]
W3C (W. Chisholm, G. Vanderheiden, I. Jacobs); »Web Content Accessibility Guidelines 1.0«, 1999. URL: *http://www.w3.org/TR/WAI-WEBCONTENT/*
Deutsche Übersetzung: R. Hartmann; »Zugänglichkeitsrichtlinien für Web-Inhalte 1.0«, 2002. URL: *http://www.w3c.de/Trans/WAI/webinhalt.html*

[REC-XHTML10 2002]
W3C HTML Working Group; »XHTML 1.0 The Extensible HyperText Markup Language (Second Edition)«, 2002. URL: *http://www.w3.org/TR/xhtml1/*
Deutsche Übersetzung: edition W3C.de (J. Muhr, S. Mintert); »XHTML 1.0: Die Extensible HyperText Markup Language (Zweite Auflage)«. URL: *http://www.edition-w3c.de/TR/xhtml1*

[REC-XHTML11 2001]
W3C HTML Working Group; »XHTML 1.1 – Module-based XHTML«, 2001. URL: *http://www.w3.org/TR/xhtml11/*
Deutsche Übersetzung: M. Görtz; »XHTML 1.1 – Modulbasiertes XHTML«, 2001. URL: *http://www.8ung.at/w3c-trans-de/xhtml11/*

[REC-XML10 2004]
W3C (T. Bray, J. Paoli, C. M. Sperberg-McQueen, E. Maler, F. Yergeau); »Extensible Markup Language (XML) 1.0 (Third Edition)«, 2004. URL: *http://www.w3.org/TR/REC-xml/*
Deutsche Übersetzung: edition W3C.de (S. Mintert); »Extensible Markup Language (XML) Version 1.0 (Zweite Auflage)«, 2002. URL: *http://www.edition-w3c.de/TR/REC-xml*

[REC-XML11 2004]
W3C (T. Bray, J. Paoli, C. M. Sperberg-McQueen, E. Maler, F. Yergeau, J. Cowan); »Extensible Markup Language (XML) 1.1«, 2004. URL: *http://www.w3.org/TR/xml11/*
Deutsche Übersetzung: edition W3C.de (S. Mintert); »Extensible Markup Language (XML) 1.1«. URL: *http://www.edition-w3c.de/TR/xml11/*

[REC-XLink10 2001]
W3C (S. DeRose, E. Maler, D. Orchard); »XML Linking Language (XLink) Version 1.0«, 2001. URL: *http://www.w3.org/TR/xlink/*
Deutsche Übersetzung: edition W3C.de (D. Hinz, S. Mintert, A. Wienberg); »XML Linking Language (XLink) Version 1.0«, 2002. URL: *http://www.edition-w3c.de/TR/xlink/*

[Rehling 2002]
B. Rehling; »Sprachbarrieren Hörgeschädigter im Internet«, 2002. URL: *http://www.taubenschlag.de/bernd/referate/digitale_chancen/sprachbarrieren.htm*

[RFC-1630 1994]
T. Berners-Lee; »Universal Resource Identifiers in WWW: A Unifying Syntax for the Expression of Names and Addresses of Objects on the Network as used in the World-Wide Web«, 1994. URL: *http://www.faqs.org/rfcs/rfc1630.html*

[RFC-1766 1995]
H. Alvestrand; »Tags for the Identification of Languages«, 1995. URL: *http://www.faqs.org/rfcs/rfc1766.html*

[RFC-1866 1995]
T. Berners-Lee, D. Connolly; »Hypertext Markup Language – 2.0«, 1995. URL: *http://www.faqs.org/rfcs/rfc1866.html*

[RFC-1945 1996]
T. Berners-Lee, R. Fielding, H. Frystyk; »Hypertext Transfer Protocol – HTTP/1.0«, 1996. URL: *http://www.faqs.org/rfcs/rfc1945.html*

[RFC-2083 1997]
T. Boutell et al.; »PNG (Portable Network Graphics) Specification Version 1.0«, 1997. URL: *http://www.ietf.org/rfc/rfc2083.txt*

[RFC-2616 1999]
R. Fielding, J. Gettys, J. Mogul, H. Frystyk, L. Masinter, P. Leach, T. Berners-Lee; »Hypertext Transfer Protocol – HTTP/1.1«, 1999. URL: *http://www.faqs.org/rfcs/rfc2616.html*
Errata: *http://skrb.org/ietf/http_errata.html*

[RFC-3066 2001]
H. Alvestrand; »Tags for the Identification of Languages«, 2001. URL: *http://www.faqs.org/rfcs/rfc3066.html*

[RFC-3986 2005]
T. Berners-Lee, R. Fielding, L. Masinter; »Uniform Resource Identifier (URI): Generic Syntax«, 2005. URL: *http://www.faqs.org/rfcs/rfc3986.html*

[Schäfer]
M. Schäfer; »Bezüglich des Begriffs behindertengerecht«. URL: *http://molily.de/behindertengerecht*

[Schneegans]
C. Schneegans; »XHTML-Einmaleins«. URL: *http://schneegans.de/web/xhtml/*

[Schneider 2001]
W. Schneider; »Deutsch für Profis – Wege zu gutem Stil«, 14. Auflage, München, Wilhelm Goldmann Verlag, 2001.

[Schulmeister 2007]
R. Schulmeister; »Grundlagen hypermedialer Lernsysteme – Theorie – Didaktik – Design«, 4. Auflage, München, Oldenbourg Wissenschaftsverlag, 2007.

[Shea, Holschlag 2005]
D. Shea, M. E. Holzschlag; »Zen und die Kunst des CSS-Designs – Inspiration und Umsetzung«, München, Addison-Wesley Verlag, 2005.

[Silver 2006]
L. Silver; »CSS Enhancements in Internet Explorer 6«, 2006. URL: *http://msdn.microsoft.com/en-us/library/bb250395.aspx*

[Storrer 1999]
A. Storrer; »Was ist eigentlich eine Homepage? – Neue Formen der Wissensorganisation im World Wide Web«. URL: *http://www.ids-mannheim.de/grammis/orbis/hp/hpframe.html*

[Trebing 2002]
L. Trebing; »Geckos Beinahe-Standard-Modus«, 2002. URL: *http://www.ltrebing.de/interessen/www/devedge-almost-standards/*

[Vester 1980]
F. Vester; »Neuland des Denkens – Vom technokratischen zum kybernetischen Zeitalter«, München, DTV, 2002.

[WD-CSS21 2006]
W3C (B. Bos, T. Çelik, I. Hickson, H. W. Lie); »Cascading Style Sheets, level 2 revision 1«, 2006. URL: *http://www.w3.org/TR/CSS21/*

[WD-XHTML5 2008]
W3C/WHATWG (I. Hickson, D. Hyatt); »HTML 5 – A vocabulary and associated APIs for HTML and XHTML«, 2008. URL: *http://www.w3.org/TR/html5/*

[WD-XHTML20 2006]
W3C (J. Axelsson, M. Birbeck, M. Dubinko, B. Epperson, M. Ishikawa, S. McCarron, A. Navarro, S. Pemberton); »XHTML™ 2.0«, 2006. URL: *http://www.w3.org/TR/xhtml2/*

[Zeldman 2007]
J. Zeldman; »Webdesign mit Webstandards – Grenzenlos kompatibel (Deutsche Ausgabe)«, München, Addison-Wesley Verlag, 2007.

Index

!

!important 304, 333
.htaccess 113, 181
@charset 288
@import 287, 303
@media 287

A > > >

a (Element) 205
abbr (Attribut, th, td) 236
abbr (Element) 226
Abkürzung 226
Absätze 201
Accessibility 89, 173
acronym (Element) 226
action (Attribut, form) 253
address (Element) 264
Akronym 226
alt (Attribut, img) 213
Alternative Inhalte 92, 213, 215, 217, 218, 268, 440
Alternative Stylesheets 277
Anführungszeichen 109, 115, 125, 245, 246, 248, 424
Anker 205
Anwenderdefinierte Farben 317
Apache Webserver 112, 163, 181
Apostroph 115, 125
applet (Element) 218
Ariadne-Faden 231
ASCII 106, 113, 179
At-Regeln 286
Ausgabemedium 276, 287
Ausrichtung
  horizontal 341
  vertikal 365
Auswahlmenüs 258
Autorenstylesheet 303

B > > >

b (Element) 227
background (Eigenschaft) 330, 411
background-attachment (Eigenschaft) 327
background-color (Eigenschaft) 302, 324
background-image (Eigenschaft) 325
background-position (Eigenschaft) 328
background-repeat (Eigenschaft) 325, 326, 328
Barrierefreiheit 89, 90, 101, 173
  Verordnungen und Richtlinien 97
Benutzbarkeit 89
Benutzereingabe 223
Betonung 222
BGG 101
big (Element) 229
Bildbearbeitung 153
  Software 155, 158
Bildschirmauflösung 137, 321, 325, 337
Bildschirmfotodienste 83
Bindestrich 122
BITV 101, 202, 226, 236, 243, 263, 404
Blockelemente 185
  Breite 387
  horizontal zentrieren 388
blockquote (Element) 248
Blocksatz 342
body (Element) 185
border (Attribut, table) 239
border (Eigenschaft) 359, 395, 411
border-collapse (Eigenschaft) 415
border-color (Eigenschaft) 358
border-spacing (Eigenschaft) 419
border-style (Eigenschaft) 356
border-width (Eigenschaft) 358
bottom (Eigenschaft) 382
Boxen 186, 347
  Hintergrund 349
  Typen 349
  Zeilenbox 370
Boxmodell 347

Boxmodell-Bug 394
br (Element) 204
Braillezeile 91
Breite
  Content 386, 394
  Element 387, 394, 401
  Tabellen 413
Brotkrumen-Navigation 231
Browser 65
  Entwicklung 65
  Erwise 65
  Hack 155
  Internet Explorer 69
  Mosaic 67
  Netscape Navigator 69
  ViolaWWW 66
Browsererweiterungen 167
Browserkompatibilität 86, 306
Browserkrieg 69
Browser-Reset 453
Browserstylesheet 302, 363

C > > >

caption (Element) 236
caption-side (Eigenschaft) 422
CDATA 150, 191
CDATA-Abschnitt 191
checked (Attribut, input) 256
cite (Attribut, del, ins) 223
cite (Attribut, q, blockquote) 248, 249, 297
cite (Element) 248, 249
class (Attribut) 195, 250, 290
classid (Attribut, object) 220
clear (Eigenschaft) 372
code (Attribut, applet) 219
code (Element) 222, 265
codebase (Attribut, applet) 219
codebase (Attribut, object) 220
codetype (Attribut, object) 220
col (Element) 238
colgroup (Element) 238
color (Eigenschaft) 301, 316, 323, 343
cols (Attribut, textarea) 261
colspan (Attribut, th, td) 243
content (Attribut, meta) 193
content (Eigenschaft) 297, 423
counter-increment (Eigenschaft) 425
counter-reset (Eigenschaft) 425
CSS 61
  Ausblick 63
  Ausgabemedium 279
  Boxmodell 347
  Einbindung 274
  Entwicklung 61
  Entwurfskonzepte 272
  Hacks 311
  Modularisierung 63
  Syntax und Vokabular 280
  Tabellenmodell 411
  Validator 152
CSSDOC 282, 453
cursor (Eigenschaft) 406

D > > >

data (Attribut, object) 216
Dateiauswahl-Steuerelement 258
datetime (Attribut, del, ins) 225
dd (Element) 233
Deckkraft 403
Definition 222, 233
del (Element) 223
Design 269
dfn (Element) 222
display (Eigenschaft) 363, 370, 398
div (Element) 250
dl (Element) 233
Doctype Switching 307, 339, 388, 394
Document Root 47, 298
Dokumentänderung 223
Dokumentbeziehungen 207, 209
Dokumenttyp-Definition 31, 144
Dokumenttyp-Deklaration 181
DOM 159, 188
Download 53, 206
dt (Element) 233
DTD 31

E > > >

Editor 135
  textorientiert 138, 142
  WYSIWYG 135, 143

Element 176
Breite 349, 370
Fluss 372, 381, 383, 399, 403
Höhe 349
Elementfluss 370
em (Element) 222
empty-cells (Eigenschaft) 421
enctype (Attribut, form) 254

F > > >

Fangs 170
Farbfehlsichtigkeit 91
Farbwerte 315
Fehlerbehandlung 145, 160
Feinstrukturierung 250, 445
Festabstände 120
fieldset (Element) 263
FileZilla 164, 165
Firebug 168
Firefox 73, 81, 145, 167, 210, 224, 249, 279, 391, 395
Entwicklung 73
testen 81
Flash 217
float (Eigenschaft) 368
Floatierung 310, 368, 372, 403, 468, 478
Floats einschließen 403
Fokus 255, 293
font (Eigenschaft) 340
font-family (Eigenschaft) 331
font-size (Eigenschaft) 337
font-style (Eigenschaft) 334
font-variant (Eigenschaft) 335
font-weight (Eigenschaft) 336
for (Attribut, label) 262
form (Element) 253
Formulare 252, 499
Beschriftung 262
Strukturierung 263
Vorbelegung 255
Vorselektierung 256
FTP 46, 113, 164

G > > >

Gebärdensprache 93
Gebrauchstauglichkeit 272
Gecko 73, 81, 246, 275, 296, 303, 308, 426
Gedankenstrich 115, 123
Generierter Content 370, 423
GIF 154, 212
GIMP 156
Grafiken 212
Groß- und Kleinschreibung 335, 346
Grundsätze der Dialoggestaltung 97
Gruppierung 250
Gültigkeit 137, 144, 146, 191

H > > >

h1, h2, h3, h4, h5, h6 (Elemente) 201
Hack 311, 395
hasLayout 310, 515
head (Element) 184
height (Attribut, applet) 219
height (Attribut, img) 212
height (Attribut, object) 216
height (Eigenschaft) 390, 394, 401
Hintergrundbild 325, 370
Positionierung 328
Scrolleigenschaft 327
Wiederholung 326
Hintergrundfarbe 324
Höhe
Content 390, 394, 401
Element 391, 394, 401
Tabellen 414
Zeilenbox 396
Homepage
Definition 27
Hörbehinderungen 93
hover 293
href (Attribut, a) 205
href (Attribut, link) 209, 276
HTML 41, 54
Ausblick 59
Entwicklung 14, 55
Formatierung 187
Kommentare 177
Kompatibilitätsrichtlinien 191, 197

Tidy 160, 170
Validator 147
html (Element) 183
HTTP 41, 49, 53, 193, 216, 257
Protokollaufbau 50
http-equiv (Attribut, meta) 193
Hyperlink 205, 293, 296
Hypertext 34, 35, 40
Entwicklung 35
Hypertextsysteme 37, 40

I > > >

i (Element) 227, 247
iCab 81, 210
id (Attribut) 195, 207, 250, 291, 304
img (Element) 212, 214
inherit (Wert) 302
Initialwert 300
Inline-Elemente 185
Breite 387
input (Element) 254
ins (Element) 223
Internet Explorer 57, 69, 71, 80, 155, 192, 216, 246, 275, 279, 293, 294, 295, 303, 306, 307, 310, 311, 312, 339, 391, 394, 514
Entwicklung 69, 71
Filter 404
Hack 155, 311
hasLayout 310, 515
testen 80
ISO 8859 107, 113, 179

J > > >

Java 217
JavaScript 26, 191, 196, 255, 265
JAWS 87, 248
JPEG 154, 212

K > > >

Kapitälchen 335
Kaskade 302
kbd (Element) 223
KHTML 76, 77, 296
Knoppix 85
Kochbar 429
Grundgerüst 436
Kontaktformular 499
Navigation 440, 463
Öffnungszeiten 491
Rezept des Tages 508
Schriftformatierung 454
Speisekarte 486
Styling 450
Kombinatoren 289, 299
Kommentare
CSS 282
XHTML 177
Kompatibilitätsrichtlinien 191, 197
Konqueror 76, 81
Entwicklung 76
testen 81
Kontaktinformationen 264
Kontrollkästchen 256

L > > >

label (Attribut, optgroup) 260
label (Element) 262
lang (Attribut) 183, 192, 197, 296
Längen
absolut 319
relativ 321
Laufweite 344
Layoutgrafiken 213
Leeres-Element-Tag 191
Leerraum 180, 187
Behandlung 347
Leerzeichen 120
geschützt 115, 121
left (Eigenschaft) 382
legend (Element) 263
Leichte Sprache 94
Lesbarkeit 332, 337, 342, 346
letter-spacing (Eigenschaft) 344
li (Element) 230
line-height (Eigenschaft) 396
link (Element) 209, 276
LinkChecker 172

Listen 229, 363, 407, 440, 509
  Verschachtelung 232
Listenelementmarkierung 297, 410
  Grafiken 409
  Position 410
  Stil 407
list-style (Eigenschaft) 410
list-style-image (Eigenschaft) 409
list-style-position (Eigenschaft) 410
list-style-type (Eigenschaft) 407, 427
Lynx 78, 82, 172, 210
  testen 82

M > > >

margin (Eigenschaft) 303, 360, 388, 395, 401, 411
Maskierung 281, 396
Mauszeiger 406
max-height (Eigenschaft) 393
maxlength (Attribut, input) 255
max-width (Eigenschaft) 390
MeasureIt 172
media (Attribut, link) 276
Memex 37
meta (Element) 193
Metaangaben 436
method (Attribut, form) 254
MIME-Typ 51, 52, 57, 145, 181, 192, 193, 206, 216, 220, 254, 266, 275, 309
min-height (Eigenschaft) 392
min-width (Eigenschaft) 389
Mosaic 67
Mozilla 71, 73, 145, 279
multiple (Attribut, select) 259

N > > >

name (Attribut, a) 205, 206
name (Attribut, applet) 219
name (Attribut, meta) 193
name (Attribut, param) 216
Namensraum 183
Navigation 202, 209, 210
Netscape Navigator 69
noscript (Element) 265
Nummerierung 408, 425

O > > >

object (Element) 214
Objekte 214
ol (Element) 230
onmouseover (Attribut) 266
opacity (Eigenschaft) 403
Open Source 71, 73, 156, 161
Opera 74, 81, 210, 246, 279, 303, 308, 313, 426
  Entwicklung 74
  testen 81
optgroup (Element) 260
option (Element) 258
Optionsfeld 256
overflow (Eigenschaft) 401

P > > >

p (Element) 202
padding (Eigenschaft) 303, 354, 394, 411
param (Element) 216
Passwortfeld 255
PDF 59
Plug-in 159, 167
PNG 155, 212
Polsterung 354
position (Eigenschaft) 376
Positionierung 368, 376, 401
  absolut 378
  fest 381
  relativ 377
pre (Element) 264, 347
Progressive Enhancement 314, 497, 506
Pseudoelemente 296
Pseudoklassen 293

Q > > >

q (Element) 245
Qualitätssicherung 79, 143, 160, 439, 514
Quelltext 222
  Formatierung 187
  Richtlinien 451

R > > >

Rahmen 359
  Breite 358
  Farbe 358
  Stil 356
Randabstand 360
Rastergrafiken 153, 212
Recommendation Process 43
rel (Attribut, a) 207
rel (Attribut, link) 209, 276
Ressource 44
  Verarbeitung 53
rev (Attribut, a) 207
rev (Attribut, link) 209
right (Eigenschaft) 382
rows (Attribut, textarea) 261
rowspan (Attribut, th, td) 243

S > > >

Safari 77, 81, 275, 308
  Entwicklung 77
  testen 82
samp (Element) 222
Schaltflächen 257
Schema-Validator 150
Schreiben im Web 95, 119
Schrift 331
  Farbe 323
  Gewichtung (Dicke) 336
  Größe 337
Schriftfamilie 331
Screenreader 91, 170, 197, 199, 202, 222, 226, 236, 248
script (Element) 265
Sehbehinderungen 91, 338
select (Element) 258
selected (Attribut, option) 258
Selektoren 288
  Attributselektor 291
  ID-Selektor 290
  Klassenselektor 290
  Pseudoelemente 296
  Pseudoklassen 293
  Spezifität 304
  Typselektor 290
  Universalselektor 289
Semantik 29, 88, 144, 220, 246, 247, 250
Serifen 333
SGML 30, 55, 57, 144, 147, 150, 191, 192
Shorttags 57, 150
Sichtbarkeit 398
size (Attribut, input) 255
size (Attribut, select) 259
Skalierung 204, 212
small (Element) 229
Spaltengruppen 238
span (Attribut, colgroup, col) 239
span (Element) 250
Sprachauszeichnung 197
src (Attribut, img) 212
src (Attribut, script) 268
Standards 85
Stapelebene 383
Stapelkontext 383
Star-HTML-Hack 290, 312
Steuerelemente 254
Steuerzeichen 106
strong (Element) 222
style (Attribut) 199, 274, 304
style (Element) 274
Stylesheets
  Import 287
  Zeichenkodierung 288
Styling 271
Suchmaschinen 184, 193, 194, 209, 222, 226
summary (Attribut, table) 236
SVG 158
  Einbettung 215

T > > >

Tabellen 234, 411, 491
  Rahmenabstand 419
  Rahmenmodell 415
  Schichten 411
  Schichtmodell 411
  Überschriften 236, 422
  Zellen 235, 365
table (Element) 235
table-layout (Eigenschaft) 413

tbody (Element) 237
td (Element) 235
text-align (Eigenschaft) 341, 388
textarea (Element) 261
Textauszeichnung 27, 29
  Entwicklung 29
Textbrowser 78, 82, 172
text-decoration (Eigenschaft) 343
Texteingabefeld 255, 261
Texteinzug 341
text-indent (Eigenschaft) 341
text-transform (Eigenschaft) 346
Textverarbeitung 28
tfoot (Element) 237
th (Element) 235
thead (Element) 237
Tidy 160, 170
title (Attribut) 199, 209, 214, 225, 226, 276
title (Element) 184
Tooltipp 199, 214, 227
top (Eigenschaft) 382
tr (Element) 235
Trennung von Markup und Layout 15, 29, 61, 137, 196, 201, 205, 272, 275, 277, 443
tt (Element) 228
type (Attribut, input) 254
type (Attribut, object) 216
type (Attribut, script) 266
type (Attribut, style) 275
Typografie 109, 119, 331, 337

U > > >

Überlauf 401
Überschriften 28, 201
ul (Element) 230
Umschließender Block 350, 401
Unicode 109, 179, 281
Universalattribute 195
URI 41, 44
URL 45
URN 49
Usability 89
Usenet 15, 22
Userstylesheet 303, 333
UTF-8 109

V > > >

Validator 143, 152, 170, 439
value (Attribut, input) 255, 256, 257, 258
value (Attribut, option) 258
value (Attribut, param) 216
var (Element) 223
Variable 223
Vektorgrafik 157
Vererbung 197, 300
vertical-align (Eigenschaft) 365
Verzeichnislisting 47
ViolaWWW 66
Virtuelle Umgebungen 84
visibility (Eigenschaft) 398

W > > >

W3C 30, 42, 59
  Bereiche 43
  Recommendation Process 43
W3C Core Styles 278
W3C Markup Validation Service 147, 161
WAI 43, 98
Wasserzeicheneffekt 328
WCAG 99
Web Developer Toolbar 169, 439, 440
Webbrowser 65
Webdesign 272
Webergonomie 89, 97, 272, 314
WebKit 77
Webseite
  Definition 25
Website
  Definition 25
Webstandards 85
WHATWG 60
white-space (Eigenschaft) 347
width (Attribut, applet) 219
width (Attribut, colgroup, col) 239
width (Attribut, img) 212
width (Attribut, object) 216
width (Eigenschaft) 386, 394
Wohlgeformtheit 146
word-spacing (Eigenschaft) 345
World Wide Web 39
  Entwicklung 40

Wortabstand 345
Worttrennung 342
Wurzelelement 146, 183, 190

X > > >

Xanadu 37
Xenu's Link Sleuth 165
XHTML 56
  Formatierung 187
  Grundgerüst 436
  Kommentare 177
  Modularisierung 57
  Syntax und Vokabular 175
  Tidy 160, 170
  Validator 147, 150
  Vorteile 58
XLink 39
XML 30, 146, 150, 159, 183, 191
XML Schema 151
xml:lang (Attribut) 183, 192, 197
XML-Deklaration 181
xmlns (Attribut) 183
XSL 59
  XSL-FO 59, 180
  XSLT 196

Y > > >

YAML 375

Z > > >

Zähler 425
Zeichenabstand 344
Zeichendarstellung 113
Zeicheneingabe 114
Zeichenkodierung 52, 104, 106, 160, 179, 181, 193, 281, 288
  Auswahlkriterien 108, 118
Zeichenpalette 117
Zeichenreferenzen 178
Zeichensatz 106
Zeichentabelle 117
Zeichenvorrat 106, 179, 281
Zeilenabstand 396
Zeilengruppen 237
Zeilenumbruch 204
Zentrieren
  horizontal 388
  vertikal 365
z-index (Eigenschaft) 383
Zitate 244
  Quellenangabe 249, 297
Zugänglichkeit 89, 99, 272
Zugänglichkeitsrichtlinien für Webinhalte 98
Zugriffskontrolldatei 113, 181
Zusammenfallende Randabstände 360
Zusammenfassende Eigenschaften 281, 302, 304

informit.de, Partner von Addison-Wesley, bietet aktuelles Fachwissen rund um die Uhr.

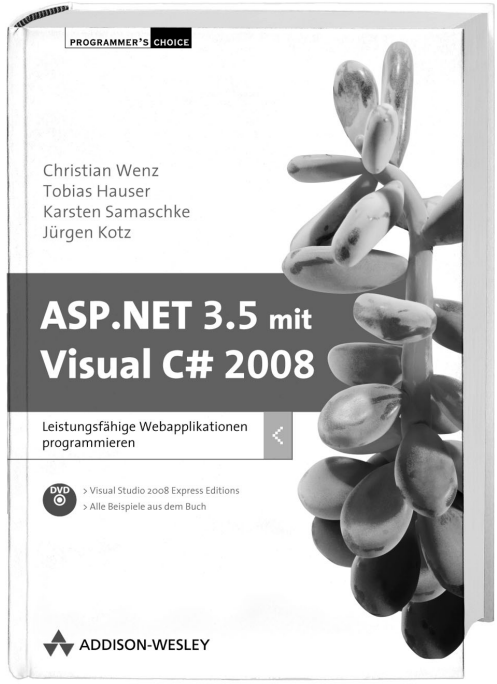
www.informit.de

In Zusammenarbeit mit den Top-Autoren von Addison-Wesley, absoluten Spezialisten ihres Fachgebiets, bieten wir Ihnen ständig hochinteressante, brandaktuelle deutsch- und englischsprachige Bücher, Softwareprodukte, Video-Trainings sowie eBooks.

wenn Sie mehr wissen wollen …

www.informit.de

THE SIGN OF EXCELLENCE



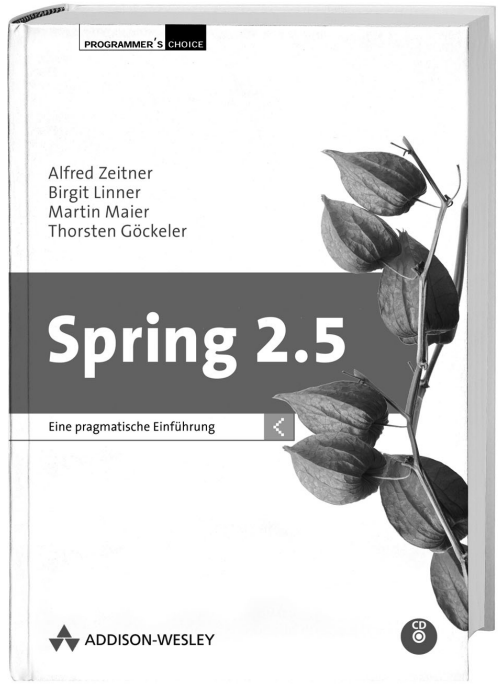
ASP.NET 3.5 baut auf dem erfolgreichen ASP.NET 2.0 auf und integriert das mächtige Framework ASP.NET AJAX. Ein-, Umsteiger und fortgeschrittene ASP.NET-Programmierer erfahren hier alles Wesentliche über das .NET Framework und Visual C#. Sie erhalten alle Tricks und Kniffe, die Sie für den erfolgreichen und professionellen Einsatz von ASP.NET benötigen: unterhaltsam, kompetent und ohne Umschweife.

Christian Wenz; Tobias Hauser; Karsten Samaschke; Jürgen Kotz
ISBN 978-3-8273-2302-6
59.95 EUR [D]

www.addison-wesley.de

[The Sign of Excellence]
ADDISON-WESLEY

THE SIGN OF EXCELLENCE



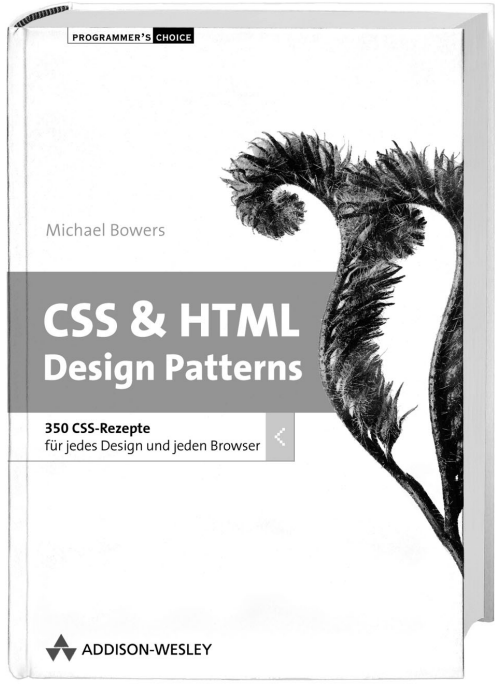
Ziel des Buches ist es, dem Leser einen schnellen und praxisnahen Einstieg im Umgang mit dem Spring-Framework zu ermöglichen. Anhand eines durchgängigen Fallbeispiels werden schrittweise Spring-Techniken eingeführt und jeweils an einem Codebeispiel erläutert. So entsteht anhand von immer komplexeren Anforderungen eines fiktiven Kunden aus einem einfachen Lagerservice ein Lagerverwaltungssystem mit Bestellfunktionalität.

Alfred Zeitner; Birgit Linner; Martin Maier; Thorsten Göckeler
ISBN 978-3-8273-2622-5
39.95 EUR [D]

www.addison-wesley.de

[The Sign of Excellence]
ADDISON-WESLEY

THE SIGN OF EXCELLENCE



Dieses Buch verfolgt einen revolutionären Ansatz: es wendet das Prinzip der Entwurfsmuster auf CSS an und entwirft so die erste Systematik zum Einsatz von Cascading Style Sheets. Der Autor kann mit rund 350 Rezepten jeden Designzweck bedienen – und alle Rezepte funktionieren mit jedem erhältlichen Browser!

Michael Bowers
ISBN 978-3-8273-2652-2
49.95 EUR [D]

www.addison-wesley.de

[The Sign of Excellence]
ADDISON-WESLEY

THE SIGN OF EXCELLENCE

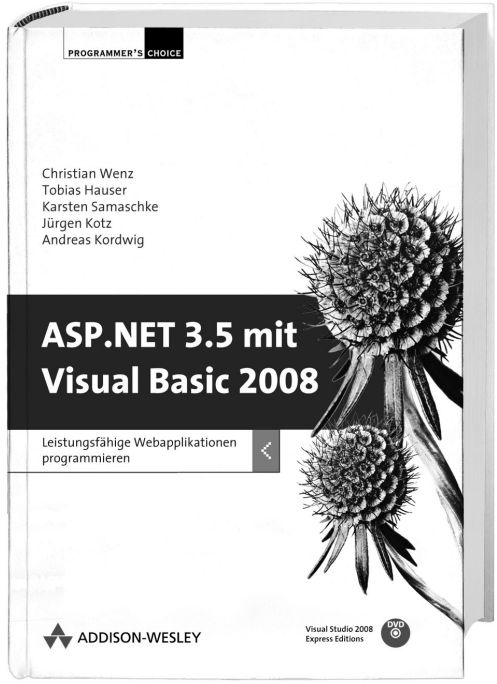
Das Buch beleuchtet die gegenwärtige Situation im Web, die verfügbaren Technologien, den Browser-Markt und Erwartungen von Anwendern. Darauf aufbauend werden die meistverbreiteten Beschreibungs- und Programmiersprachen für Websites praxisnah und anhand von konkreten Beispielen vermittelt. Das Spektrum reicht dabei von HTML über CSS, JavaScript/DOM bis zu PHP/MySQL. Der Leser wird darüber hinaus aber auch in die Konfiguration eines Webservers wie Apache eingeführt und erwirbt wichtige Grundkenntnisse im servertypischen Betriebssystem Linux. Extra für dieses Kompendium wurde ein Wiki geschaffen, dass der Autor regelmäßig pflegt, wodurch keine Frage offen bleibt.

Stefan Münz
ISBN 978-3-8273-2678-2
49.95 EUR [D]

www.addison-wesley.de

[The Sign of Excellence]
ADDISON-WESLEY

THE SIGN OF EXCELLENCE



ASP.NET 3.5 baut auf dem erfolgreichen ASP.NET 2.0 auf und integriert das mächtige Framework ASP.NET AJAX. Ein-, Umsteiger und fortgeschrittene ASP.NET-Programmierer erfahren hier alles Wesentliche über das .NET Framework und Visual Basic 9. Sie erhalten alle Tricks und Kniffe, die Sie für den erfolgreichen und professionellen Einsatz von ASP.NET benötigen: unterhaltsam, kompetent und ohne Umschweife.

Christian Wenz; Tobias Hauser; Karsten Samaschke; Jürgen Kotz; Andreas Kordwig
ISBN 978-3-8273-2682-9
59.95 EUR [D]

www.addison-wesley.de

[The Sign of Excellence]
ADDISON-WESLEY

Copyright

Daten, Texte, Design und Grafiken dieses eBooks, sowie die eventuell angebotenen eBook-Zusatzdaten sind urheberrechtlich geschützt. Dieses eBook stellen wir lediglich als persönliche Einzelplatz-Lizenz zur Verfügung!

Jede andere Verwendung dieses eBooks oder zugehöriger Materialien und Informationen, einschliesslich

- der Reproduktion,
- der Weitergabe,
- des Weitervertriebs,
- der Platzierung im Internet, in Intranets, in Extranets,
- der Veränderung,
- des Weiterverkaufs
- und der Veröffentlichung

bedarf der schriftlichen Genehmigung des Verlags.

Insbesondere ist die Entfernung oder Änderung des vom Verlag vergebenen Passwortschutzes ausdrücklich untersagt!

Bei Fragen zu diesem Thema wenden Sie sich bitte an: info@pearson.de

Zusatzdaten

Möglicherweise liegt dem gedruckten Buch eine CD-ROM mit Zusatzdaten bei. Die Zurverfügungstellung dieser Daten auf unseren Websites ist eine freiwillige Leistung des Verlags. Der Rechtsweg ist ausgeschlossen.

Hinweis

Dieses und viele weitere eBooks können Sie rund um die Uhr und legal auf unserer Website

http://www.informit.de

herunterladen