

Frank Bongers

Inkl.
XForms
und
XFrames

XHTML, HTML und CSS

Handbuch und Referenz

- Syntaxgrundlagen, Praxis, Elementreferenz
- Modernes Seitenlayout und standardkonforme Websites
- Barrierefreiheit, Accessibility, Mobile Webdesign

Galileo Computing

Inkl.
Barrierefreiheit
und Mobile
Webdesign

& CSS

XHTML, HTML
Handbuch und Referenz

Bongers

483



Auf einen Blick

1	(X)HTML & CSS – Einführung	21
2	(X)HTML – Die Syntaxgrundlagen	35
3	(X)HTML – die Dokumentstruktur	57
4	CSS Grundsyntax	85
5	Seitenlayout mit CSS	109
6	(X)HTML im Contentbereich	131
7	Navigationsgestaltung mit CSS	165
8	Code-Aktualisierung nach XHTML	195
9	Webdesign für mobile Clients	219
10	Barrierefreiheit und Accessibility	239
11	Werkzeuge für Webdesigner	271
12	HTML-Sprachfamilien	293
13	XForms und XFrames	359
14	(X)HTML-Elementreferenz	409
15	(X)HTML-Attributreferenz	915
16	Referenz HTML Entities	955
17	URL-Encoding und Language-Codes	971
18	Farben und Farbwerte	979
19	CSS-Referenz	991
A	W3C-Ressourcen	1137
B	Weitere Ressourcen	1165
C	Inhalt der Begleit-CD	1177

Inhalt

Vorwort	17
---------------	----

1 (X)HTML & CSS – Einführung 21

1.1	Am Anfang war HTML	21
1.1.1	Eine sehr liberale Grammatik und ihre Vorteile	22
1.1.2	Verarbeitung und Ergänzung des Quellcodes durch den Parser	23
1.1.3	Die frühen Jahre von HTML	25
1.2	HTML 4.0 und CSS	26
1.2.1	HTML 4.0 und seine drei DTD-Varianten	26
1.2.2	Die Präsentationssprache CSS	27
1.3	Der Schritt von HTML zu XHTML	28
1.3.1	Ein einfaches XHTML-Dokument	29
1.4	Die Gegenwart von HTML und XHTML	30
1.4.1	Abwandlungen von HTML und XHTML	31
1.5	Der Schritt von CSS 1.0 zu CSS 2.1	31
1.6	Die Zukunft von HTML und XHTML	32
1.7	Die Zukunft von CSS	33
1.8	Zusammenfassung	33

2 (X)HTML – Die Syntaxgrundlagen 35

2.1	Grundsätzliches über SGML- und XML-Markupsprachen	35
2.1.1	Syntax von Start- und Endmarken	36
2.1.2	Elemente und ihre Inhaltsmodelle	36
2.1.3	Attribute – Metainformation über Elemente	38
2.1.4	Entitäten – allgemeine Ersetzungsvorschriften	39
2.1.5	Inhaltsmodelle von Elementen	41
2.2	HTML als SGML-Anwendung	43
2.2.1	Die Dokumenttyp-Definition	43
2.2.2	Elementvereinbarungen für HTML	43
2.2.3	Kommentare in HTML	46
2.2.4	Das Vokabular von HTML	47
2.3	XHTML als XML-Anwendung	48
2.3.1	Das Vokabular von XHTML	54
2.4	Zusammenfassung	55

3	(X)HTML – die Dokumentstruktur	57
3.1	Die Wahl der geeigneten Sprachversion	57
3.1.1	Möglichkeit 1: Korrekt geschriebenes HTML	57
3.1.2	Möglichkeit 2: Wohlgeformtes XHTML	58
3.1.3	Möglichkeit 3: Streng konformes XHTML	59
3.1.4	Ein paar Überlegungen zu XHTML 1.1	59
3.2	Grundaufbau von HTML- und XHTML-Dokumenten	60
3.3	Aufbau und Aufgaben des Dokumentkopfes	65
3.4	Aufbau und Aufgaben des Dokumentkörpers	69
3.5	Die Doctype-Deklaration und ihre Bedeutung	76
3.6	Beispiele für HTML-Grundstrukturen mit Doctype	79
3.6.1	HTML-Dokumenttypen	79
3.6.2	XHTML-Dokumenttypen	81
3.7	Zusammenfassung	83
4	CSS Grundsyntax	85
4.1	Die Grundsyntax von CSS	85
4.1.1	Verknüpfung von CSS-Anweisungen mit einem HTML-Dokument	85
4.1.2	Aufbau einer CSS-Anweisung	90
4.1.3	Die Deklaration einer CSS-Eigenschaft	91
4.2	Selektoren und Style-Einbindung	92
4.2.1	Der Selektor – Die Grundformen	92
4.2.2	Einbindung eines externen Stylesheets	98
4.2.3	Welcher Selektor – wann, wofür?	101
4.2.4	Kommentare in Stylesheets	102
4.2.5	Cascading, Addition und Konfliktregelung bei CSS-Anweisungen	103
4.2.6	Was bedeutet das »cascading« in Cascading Stylesheets?	103
4.2.7	CSS-Eigenschaften	104
4.3	Das CSS-Boxmodell	105
4.4	Zusammenfassung	107
5	Seitenlayout mit CSS	109
5.1	Grundüberlegungen zum Seitenaufbau	109
5.2	Umsetzung der Bereiche in Quellcode – Floatlayout	114

5.3	Quelltextreihenfolge und Accessibility	119
5.3.1	Die Wahl der Quelltextreihenfolge	119
5.3.2	Einbau der Sprunglinks	121
5.4	Umsetzung der Bereiche in Quellcode – Positionslayout	122
5.4.1	Schritt 1 – Festlegung der Containerbreiten	122
5.4.2	Schritt 2 – Platz für die Navigation schaffen	123
5.4.3	Schritt 3 – Positionierung des Navigationscontainers	124
5.4.4	Absolute und relative Positionierung	126
5.4.5	CSS-Eigenschaften für Positionierungszwecke	128
5.5	Zusammenfassung	128

6 (X)HTML im Contentbereich 131

6.1	Contentmarkup mit Textabsätzen und Überschriften	131
6.1.1	Mögliche globale Grundeinstellungen per CSS	131
6.1.2	Die Überschriften	134
6.1.3	Der Textabsatzcontainer	135
6.2	Inlineelemente zur Textauszeichnung	137
6.3	Der Hyperlinkcontainer, revisited	142
6.3.1	Welche Stylingmöglichkeiten bietet CSS für Hyperlinks?	145
6.3.2	Welche HTML-Attribute werden für Hyperlinks benötigt?	145
6.4	Eingebettete Medien	146
6.4.1	Welche Stylingmöglichkeiten bietet CSS für eingebettete Medien?	148
6.4.2	Welche HTML-Attribute werden für eingebettete Medien benötigt?	148
6.5	Listen	148
6.5.1	Welche Stylingmöglichkeiten bietet CSS für Listen?	151
6.5.2	Welche HTML-Attribute werden für Listen benötigt?	151
6.6	Tabellen	151
6.6.1	Welche Stylingmöglichkeiten bietet CSS für Tabellen? ...	155
6.6.2	Welche HTML-Attribute können für Tabellen verwendet werden?	155
6.7	Formulare	156
6.8	Zusammenfassung	163

7 Navigationsgestaltung mit CSS 165

7.1	Vorüberlegungen	165
7.1.1	Grundprinzipien der Navigationsgestaltung	166
7.1.2	Arten der Navigation	167
7.2	Navigationsmarkup	169
7.2.1	Horizontale Navigation mit Trennsymbolen	169
7.2.2	Vertikale Navigation mit Zeilenumbrüchen	170
7.2.3	Vertikale Navigation mit Textabsätzen	172
7.2.4	Vertikale Navigation mit Listen	173
7.3	Vertikale Navigation	174
7.4	Navigationspräsentation	176
7.4.1	Das Display-Property – Den Anker zur Fläche machen ...	176
7.4.2	CSS-Pseudoklassen – Mouse-Over-Effekte mit CSS	179
7.5	Vertikale verschachtelte Navigation	180
7.6	Horizontale Navigation	183
7.7	Karteireitermenü auf Listenbasis	187
7.8	Grafikgestützte Navigation auf Listenbasis	190
7.9	Zusammenfassung	192

8 Code-Aktualisierung nach XHTML 195

8.1	Codeaktualisierung mit HTML Tidy	195
8.1.1	HTML Tidy als Kommandozeilen-Tool	196
8.2	GUI-Versionen von HTML Tidy	206
8.2.1	Tidy UI für Windows	206
8.2.2	Tidy GUI für Windows	208
8.2.3	HTML Tidy in HTML-Kit	210
8.3	Codeaktualisierung von Hand	213
8.3.1	Praxisbeispiel eines 30 Minuten Redesigns	214
8.4	Zusammenfassung	218

9 Webdesign für mobile Clients 219

9.1	Welche Markup-Sprache für mobile Geräte?	219
9.2	Technische Randbedingungen mobiler Geräte	220
9.3	Displays von Mobilgeräten	221
9.4	Design für mobile Geräte	222
9.4.1	Schriftgröße	224
9.4.2	Quelltextaufbau	224
9.4.3	Farben	226
9.4.4	Navigation	227

9.5	CSS für mobile Geräte	228
9.5.1	Überlegungen zu einem Stylesheet für mobile Clients ...	229
9.6	Test von Mobil-Designs mit Opera	233
9.6.1	Test in Normalansicht	233
9.6.2	Test in Small-Screen Rendering-Ansicht	235
9.7	Test von Mobil-Designs mit Openwave Phone Simulator	236
9.8	Zusammenfassung	237

10 Barrierefreiheit und Accessibility 239

10.1	Barrierefreiheit – für wen?	239
10.1.1	Warum barrierefrei programmieren?	240
10.1.2	Technologien zur Überwindung von Barrieren	243
10.2	Das Einmaleins der Barrierearmut	244
10.2.1	Seitenaufbau und Layout	245
10.2.2	Verwendung von Schriften und Farben	247
10.2.3	Quelltext und Formatierung	249
10.2.4	Barrierefreie Tabellen	251
10.2.5	Links und Navigation	253
10.2.6	Barrierefreie Formulare	257
10.2.7	Bilder, Scripte und Multimedia	264
10.2.8	Weiterleitungen, Popups, dynamische Inhalte	267
10.2.9	Barrierefreie Inhalte	269
10.3	Zusammenfassung	269

11 Werkzeuge für Webdesigner 271

11.1	Editoren	271
11.1.1	Quelltexteditoren	271
11.1.2	WYSIWYG HTML-Editoren	274
11.2	Toolbars und weitere nützliche Werkzeuge	276
11.2.1	Developer Toolbars	276
11.2.2	Wave Toolbar	281
11.2.3	Accessibility-Tools	282
11.2.4	Bildschirmlineale	283
11.2.5	Colorpicker und Farbtools	284
11.3	Tools, Plug-ins und Erweiterungen	287
11.4	Textbrowser und Textbrowser-Simulationen	288

12 HTML-Sprachfamilien	293
12.1 Die Vergangenheit: HTML-Versionen 1.0 bis 3.2	293
12.1.1 HTML Level 0	293
12.1.2 HTML 1.0	295
12.1.3 HTML 2.0	295
12.1.4 HTML 3.2	297
12.2 Verpasste Chancen: HTML+ und HTML 3.0	298
12.2.1 HTML+ (HTMLPlus)	300
12.2.2 HTML 3.0	302
12.3 HTML 4.0 und HTML 4.01	303
12.3.1 HTML 4.01 strict	305
12.3.2 HTML 4.01 transitional	307
12.3.3 HTML 4.01 frameset	307
12.4 XHTML 1.0 und XHTML 1.1	308
12.4.1 XHTML 1.0 strict	309
12.4.2 XHTML 1.0 transitional	309
12.4.3 XHTML 1.0 frameset	310
12.4.4 XHTML 1.1	310
12.5 Bausteine: Die Modularisierung von XHTML	312
12.5.1 Der Common-Attributsatz	313
12.5.2 Core-Module	314
12.5.3 Das Strukturmodul	316
12.5.4 Das Textmodul	316
12.5.5 Das Hypertextmodul	318
12.5.6 Das Listenmodul	319
12.5.7 Das Präsentationsmodul (presentation module)	320
12.5.8 Das Formular-Basismodul (Basic Forms Module)	321
12.5.9 Tabellen-Basismodul (Basic Tables Module)	323
12.5.10 Formularmodul	325
12.5.11 Das Tabellenmodul	326
12.5.12 Das Image Modul	327
12.5.13 Das Applet Modul	328
12.5.14 Das Base Modul	328
12.5.15 Das Bi-Directional Text Modul	328
12.5.16 Das Client-Side Image Map Modul	329
12.5.17 Das Edit Modul	330
12.5.18 Das Frames Modul	330
12.5.19 Das iFrame Modul	331
12.5.20 Das Intrinsic Events Modul	331
12.5.21 Das Legacy Modul	332
12.5.22 Das Link Modul	335

12.5.23	Das Meta Information Modul	335
12.5.24	Das Name Identification Modul	336
12.5.25	Das Object Modul	336
12.5.26	Das Scripting Modul	336
12.5.27	Das Server-Side Image Map Modul	337
12.5.28	Das Style Attribut Modul	337
12.5.29	Das Style Sheet Modul	338
12.5.30	Das Target Modul	338
12.6	Die Hostlanguage: XHTML Basic	339
12.7	Die Zukunft: XHTML 2.0	340
12.7.1	Übersicht über die Elemente von XHTML 2.0	341
12.7.2	Modularisierung von XHTML 2.0	342
12.7.3	Allgemeiner Attributsatz der XHTML 2.0-Elemente	344
12.8	HTML ohne W3C: HTML 5.0	346
12.8.1	Die WHAT WG	346
12.8.2	»Web Applications« – auch »HTML 5.0« genannt	346
12.8.3	Geplante Erweiterungen für HTML 5.0	347
12.9	HTML für mobile Endgeräte	350
12.9.1	cHTML	350
12.9.2	iMode 1.0	352
12.9.3	iMode 2.0	352
12.9.4	iMode 3.0	353
12.9.5	iMode 4.0	354
12.9.6	Doti 1.0	354
12.9.7	HTML 4.0 mobile	355
12.10	XHTML für mobile Endgeräte	355
12.10.1	iMode XHTML 1.0	355
12.10.2	iMode XHTML 1.1	356
12.10.3	XHTML mobile profile	357

13 XForms und XFrames 359

13.1	XForms	359
13.1.1	Anwendungen und Implementierungen von XForms	360
13.1.2	Die Elemente von XForms	361
13.1.3	Der Namensraum von XForms	362
13.1.4	Die Module von XForms	362
13.1.5	Datentypen in XForms	368
13.1.6	XPath in XForms	368
13.1.7	XForms-Funktionen	369
13.1.8	Events in XForms	371
13.1.9	CSS-Selektoren für XForms	374

Inhalt

13.1.10	Aufbau eines XFormulars	375
13.1.11	Obligatorische und optionale Eingaben	378
13.1.12	Reglementierung der Eingabedaten durch XML-Schema	379
13.1.13	Versenden der Formulardaten – Der Submit-Button	384
13.1.14	Mehrfaches Submit	385
13.1.15	Mehrere Formulare	386
13.1.16	Beziehung zwischen Feldern und Dateninstanz	387
13.1.17	Beispiel 1: Ein- und Ausgabefelder in XFormularen	389
13.1.18	Beispiel 2: Eingabe eines Wertes innerhalb eines Wertebereichs	394
13.1.19	Beispiel 3: Eingabe eines Datums per XForm	396
13.2	XFrames	399
13.2.1	Die Elemente von XFrames	399
13.2.2	Beispiele für XFrames-Dokumente	404
13.2.3	Aufruf eines XFrames-Dokuments	406
13.3	Zusammenfassung	407

14 (X)HTML-Elementreferenz 409

14.1	a	411
14.2	abbr	430
14.3	acronym	433
14.4	address	437
14.5	applet	439
14.6	area	446
14.7	b	454
14.8	base	458
14.9	basefont	463
14.10	bdo	466
14.11	big	473
14.12	blockquote	476
14.13	body	481
14.14	br	493
14.15	button	498
14.16	caption	503
14.17	center	507
14.18	cite	510
14.19	code	512
14.20	col	515
14.21	colgroup	521
14.22	dd	527

14.23 del	530
14.24 dfn	533
14.25 dir	536
14.26 div	539
14.27 dl	549
14.28 dt	554
14.29 em	557
14.30 embed	560
14.31 fieldset	569
14.32 font	573
14.33 form	580
14.34 frame	592
14.35 frameset	604
14.36 h1 – h6	615
14.37 head	618
14.38 hr	621
14.39 html	626
14.40 i	633
14.41 iframe	635
14.42 img	643
14.43 input	655
14.44 ins	671
14.45 isindex	674
14.46 kbd	679
14.47 label	681
14.48 legend	685
14.49 li	688
14.50 link	693
14.51 listing	704
14.52 map	706
14.53 marquee	708
14.54 menu	715
14.55 meta	719
14.56 nextid	738
14.57 noframes	740
14.58 noscript	744
14.59 object	748
14.60 ol	759
14.61 optgroup	765
14.62 option	768
14.63 p	772

Inhalt

14.64	param	775
14.65	plaintext	779
14.66	pre	781
14.67	q	785
14.68	s	789
14.69	samp	791
14.70	script	794
14.71	select	803
14.72	small	809
14.73	span	812
14.74	strike	815
14.75	strong	817
14.76	style	819
14.77	sub	826
14.78	sup	828
14.79	table	830
14.80	tbody	845
14.81	td	851
14.82	textarea	863
14.83	tfoot	871
14.84	th	875
14.85	thead	887
14.86	title	892
14.87	tr	894
14.88	tt	900
14.89	u	903
14.90	ul	905
14.91	var	910
14.92	xmp	912
15 (X)HTML-Attributreferenz		915
15.1	Standardattribute in HTML und XHTML	915
15.2	Standard Eventhandler-Attribute	919
15.2.1	Eventhandler auf Applikations- und Interaktionsebene	920
15.2.2	Allgemein einsetzbare Eventhandler	921
15.3	Microsoft-Erweiterungen – Eventhandler	928
15.4	Microsoft-Erweiterungen – Standardattribute	947
15.4.1	Access-Attribute	947
15.4.2	Databinding-Attribute	949
15.4.3	Edit-Attribute	951

16 Referenz HTML Entities	955
16.1 HTML-Entities und Character-Entities	955
17 URL-Encoding und Language-Codes	971
17.1 URL-Encoding	971
17.2 Language Codes nach RFC 1766	974
18 Farben und Farbwerte	979
18.1 Farbnamen und Farbwerte	979
19 CSS-Referenz	991
19.1 Grundlagen	991
19.1.1 Formatierungsregeln	991
19.1.2 Position der Styledefinition	995
19.1.3 Präferenz und Cascading-Prinzip	999
19.1.4 Kommentare in Stylesheets	1002
19.2 CSS At-Rules	1003
19.3 CSS Selektoren CSS 1 und 2	1006
19.3.1 Attributabhängige Selektoren	1013
19.3.2 Pseudoelemente	1014
19.3.3 Pseudoklassen	1017
19.3.4 Pseudo-Kontextselektoren	1021
19.4 Werte und Maßeinheiten	1023
19.5 Properties CSS 1 und 2	1028
19.5.1 Textproperties	1028
19.5.2 Fontproperties	1038
19.5.3 Farben und Hintergründe	1044
19.5.4 Boxproperties	1053
19.5.5 Kontur-Properties	1067
19.5.6 Positioning-Properties	1072
19.5.7 Display-Property	1090
19.5.8 Listenstyle-Properties	1098
19.5.9 Tabellen-Properties	1103
19.5.10 Sonstige Properties	1111
19.5.11 Contentgenerierung	1114
19.5.12 Seitenformatierung (Druckausgabe)	1117

Inhalt

19.6	Browserspezifische Properties	1121
19.6.1	Scrollbar-Properties (Internet Explorer und Opera)	1121
19.7	Doctype-Switching und das CSS-Boxmodell	1127
19.7.1	Debugging von CSS	1132

Anhang..... 1135

A	W3C-Ressourcen	1137
A.1	W3C-Spezifikationen	1137
A.1.1	W3C Ressourcen zu HTML	1139
A.1.2	W3C Ressourcen zu XHTML 1	1141
A.1.3	W3C Ressourcen zu XHTML 2	1145
A.1.4	W3C Ressourcen zu CSS 1 und 2	1146
A.1.5	W3C Ressourcen zu CSS 3	1147
A.1.6	Weitere CSS-Ressourcen	1155
A.1.7	W3C Ressourcen zu XML	1156
A.1.8	W3C-Ressourcen zu SVG	1158
A.1.9	W3C Ressourcen zu Accessibility	1160
A.1.10	Weitere W3C-Ressourcen	1161
B	Weitere Ressourcen	1165
B.1	WHAT WG Initiative	1165
B.2	Mobiles Web	1166
B.3	Browser und Browserspezifikationen	1167
B.4	Webdeveloper-Tools	1169
B.4.1	Validatoren	1169
B.4.2	Toolbars	1170
B.4.3	Accessibility Tools	1171
B.5	Foren und Webzines	1172
B.6	Editoren	1173
B.6.1	Quelltexteditoren	1173
B.6.2	WYSISWYG-HTML-Editoren	1175
C	Inhalt der Begleit-CD	1177
C.1	Quellcodebeispiele – Einführungskapitel	1177
C.2	Quellcodebeispiele – Referenzkapitel HTML	1177
C.3	Tools und Werkzeuge	1177
C.3.1	Hilfsprogramme	1177
C.3.2	Editoren	1177
	Index	1179

Vorwort

Da es über mein Lieblingsthema HTML schon eine Vielzahl von Publikationen gab, sollte und musste dieses Buch »anders« sein als die bislang verfügbaren. Aus meiner langjährigen Erfahrung als Webdesigner und durch meine Unterrichtstätigkeit als Dozent war mir bewusst, dass es das komplette HTML-Buch bis heute nicht gibt.

Ein vollständiges Buch über HTML

Zusammen mit meinem Lektor kamen wir zu dem Schluss, dass es um eine wirkliche Vollständigkeit der Darstellung gehen sollte, die alle Aspekte von XHTML und HTML umfassen sollte. Auch CSS, ohne das im Bereich Webdesign heutzutage nichts mehr denkbar ist, sollte eingeschlossen werden. Hätte ich mir zu jenem Zeitpunkt eine Vorstellung von der mit diesen Anforderungen verbundenen Rechercharbeit gemacht, wäre dieses Buch vielleicht nicht in dieser Form geschrieben worden.

Ein praxisnahes Buch über HTML

Zahlreiche Entdeckungen und Wiederentdeckungen konnten wegen der (dann nun doch) beschränkten Seitenzahl nicht in dieses Buch gelangen – an dieser Stelle soll aber, zumindest ein einziges Mal, Netscapes geheimnisumwittertes Element `<hype>` erwähnt werden (es hat tatsächlich existiert). Natürlich war das Ziel dieses Buches nicht eine historische Abhandlung, sondern eine umfassende und praxisnahe Beschreibung der aktuellen Möglichkeiten, die XHTML und HTML in Form ihrer zahlreichen Elemente und Attribute bieten.

Aufgenommen in die ausführliche Referenz wurden HTML-Elemente nur dann, wenn einer der folgenden Punkte erfüllt war:

- ▶ Gültigkeit in einer der aktuellen Versionen von HTML oder XHTML
- ▶ Unterstützung in aktuellen Browsern
- ▶ Besondere Bedeutung in bestimmtem Praxiskontext

Aus diesem Grund werden neben den offiziell spezifizierten auch proprietäre Elemente wie `<embed>` oder `<marquee>` ausführlich beschrieben. Auch bei den Attributen wollte ich mich nicht auf die offiziellen beschränken. Gerade für den Internet Explorer existiert – Puristen mögen mir diese Aussage verzeihen – eine

Reihe von Erweiterungen, die höchst interessante Funktionalitäten bieten, über die es sich meiner Meinung nach lohnt, zumindest etwas zu erfahren.

Im Bereich CSS ist die Sachlage homogener – das Gewicht der Referenz im Rahmen dieses Buchs liegt auf CSS 2.1, da dies den Stand der aktuellen Browserunterstützung widerspiegelt. Eine Behandlung von CSS 3 ist zum gegenwärtigen Zeitpunkt noch müßig und unterblieb deshalb. Sie würde ohnehin ein eigenes Buch dieses Umfangs erfordern.

Ein einführendes Buch über HTML

Ohne eine anwendungsbezogene Verortung nützt eine Referenz jedoch wenig. Die ersten elf Kapitel beschäftigen sich daher mit praktischen Aspekten, die von einer grundlegenden Einführung in die HTML-Syntax und -Anwendung über Design mit CSS und XHTML bis hin zu Webdesign für Mobilgeräte reichen. Weitere Kapitel sind der Aktualisierung älterer Websites, der Barrierefreiheit sowie Tools und Werkzeugen gewidmet, die Ihnen als Webdesigner von Nutzen sein werden.

Ein Abriss über die Entwicklung der Sprachfamilien und Dialekte von HTML von den Anfängen bis hin zu XHTML 2.0 stellt den Auftakt zum umfassenden Referenzteil dar, der den Hauptteil des Buches bildet. In diesem Zusammenhang finden Sie auch eine Einführung in XForms und XFrames, die Ihnen einen Ausblick in die Zukunft der Formular- und Framesetgestaltung bietet. Auf die vollständigen Element- und Attributreferenzen folgen Abhandlungen, die HTML-Entitäten, URL-Encoding und Language Codes sowie Farben und Farbwerte betreffen. Den Abschluss des Buches bilden kommentierte Verzeichnisse der im Buch genannten Ressourcen und Spezifikationen.

Auf der Begleit-CD des Buches finden Sie neben den Quelltextbeispielen zu den Einführungskapiteln und der Referenz auch eine Reihe von nützlichen Tools und Editoren.

Die Icons in diesem Buch

Im Buch finden Sie viele Zusatzinformationen. Einige sind mit Icons gekennzeichnet, die Ihnen anzeigen, welcher Art die Information ist.

- [!] *Achtung:* Hier werden Sie auf potenzielle Fehlerquellen aufmerksam gemacht.
- [»] *Hinweis:* Diese Absätze enthalten hilfreiche Zusatzinformationen.
- [zB] *Beispiel:* Hier finden Sie Beispiele, die Ihnen zeigen, wie Sie das dargestellte Wissen in der Praxis umsetzen können.


Danksagungen

Ein Buch mit einem Umfang wie diesem kann sich der Autor nicht allein auf die Fahne schreiben – ohne die aktive Unterstützung vieler anderer stünde er mit einem solchen Unterfangen auf verlorenem Posten. An dieser Stelle möchte ich allen danken, ohne die dieses Projekt nie zu Ende gebracht worden wäre. An erster Stelle meiner Frau Carina Prange, die mir mit viel Zuspruch über einen weit längeren Zeitraum als verabredet und absehbar den Rücken freihielt. An zweiter Stelle steht, wie immer, mein Lektor Stephan Mattescheck, der meinen wiederholten Terminüberschreitungen ein übermenschliches Maß an Geduld entgegensetzte und mit seinem kompetenten Rat dann doch alles zu einem guten Ende brachte. Ganz herzlich danken möchte ich auch meinem Fachgutachter Carsten Möhrke, der mich auf zahlreiche Unklarheiten in meinem Manuskript hinwies und so für deren Beseitigung sorgte und meinem Korrektor Jürgen Dubau, dessen wachsamen Auge kein falsch gesetztes Komma entging.

Last not least mein Dank an alle Leser dieses Vorworts, die bis hierher durchgehalten haben. Ich hoffe, dass dieses Buch, das Sie in den Händen halten, Ihnen nützlich sein wird.

Frank Bongers

Berlin



Wo kommt es eigentlich her, dieses HTML? Warum existiert inzwischen auch ein XHTML? Und was hat es mit CSS auf sich? Dieses Kapitel bietet Antworten in Form einer kurzen Einführung als Themenverortung – durchaus auch mit einem kleinen Blick hinter die technischen Kulissen.

1 (X)HTML & CSS – Einführung

In der heutigen Zeit, wo das Internet zur Selbstverständlichkeit geworden ist, kommt man kaum umhin, früher oder später mit HTML in Berührung zu geraten – immerhin handelt es sich hier um die Sprache, die jeder Webseite zugrunde liegt. Dabei ist diese Sprache, historisch betrachtet, gar nicht ausgesprochen alt: Erst etwa 1990 wurde sie vom Engländer Tim Berners-Lee am Schweizer CERN-Institut ins Leben gerufen. Sein Ziel bestand anfänglich nur darin, die Kommunikation unter den Mitarbeitern des Instituts über ein Rechnernetzwerk zu erleichtern. Ganz nebenbei erfand er dabei noch den Webserver und den Webbrowser, schrieb das Übertragungsprotokoll HTTP für Webseiten und prägte den Begriff »URI« (Uniform Resource Identifier).¹

1.1 Am Anfang war HTML

HTML, die »Hypertext Markup Language«, wie Berners-Lee seine Schöpfung taufte, war auf **Einfachheit** getrimmt: Jeder in Frage kommende Client (sprich »Browser«) sollte in der Lage sein, ein derartiges Dokument darzustellen, ungeachtet der sonstigen Randbedingungen wie Betriebssystem oder Rechnerhardware. Die Art der Darstellung (»wie«) wurde der Anwendung überlassen, das Dokument sollte lediglich eine Informationsstruktur (»was«) beinhalten.

Informationsaustausch über Rechnernetzwerke war nicht eigentlich etwas Neues – immerhin gab es zu diesem Zeitpunkt seit längerem E-Mail und FTP –, doch wirklich neu war die »Killerapplikation« von HTML, nämlich die Möglichkeit, Verweise zwischen Dokumenten zu beschreiben. Dem Traum des globalen

¹ Im Jahr 2004 wurde Berners-Lee für seine Verdienste von der britischen Königin Elisabeth II. zum Ritter geschlagen.

Hypertext-Netzwerkes² kam man schlagartig recht nahe. Obendrein waren diese Verweise höchst intuitiv zu bedienen, nämlich per Mausklick.

Unterscheidbare Bereiche des Dokuments werden mit **Marken**, so genannten »Tags« gekennzeichnet, wobei **Startmarken** (Starttags) und **Endmarken** (Endtags) für Bereiche gesetzt werden konnten. Die Marken der Bereiche (letztere werden auch als **Elemente** der Seite bezeichnet) werden mit Winkelklammern eingeleitet und beendet. Endmarken enthalten zusätzlich einen, dem dort wiederholten Markenbezeichner vorangestellten Slash. Das Prinzip war also so einfach wie einleuchtend:

```
<Marke> ... Inhalte des Elements ... </Marke>
```

Anmerkung: Der hier zugrunde liegende Begriff »Markup« kommt nicht ursprünglich aus dem Computerbereich, sondern diente als Bezeichnung für das Vorbereiten eines Manuskript für den Druck (»to mark something up«) – gemeint sind die handschriftlichen Randbemerkungen des Korrektors, die sich an den Setzer richteten und die gewünschte Umsetzung (wie »fett«, »kursiv«, »eingerrückt«) der so markierten Bereiche der Vorlage verdeutlichen sollten.

1.1.1 Eine sehr liberale Grammatik und ihre Vorteile

Hierbei mussten nicht alle Dokumentbereiche explizit gekennzeichnet werden. Ein nach den sehr liberalen Regeln der Anfangszeit vollkommen gültiges HTML-Dokument sieht so aus (mehr ist in der Tat nicht erforderlich):

```
<title>Dies ist ein gueltiges HTML-Dokument</TITLE>
<P>Es enthaelt einen anklickbaren
<a HREF=anderes_dokument.html>Verweis</a>
auf ein anderes Dokument.<BR>
Eine neue Zeile am Ende.
```

Listing 1.1 01_liberal.html – Ein sehr liberal geschriebenes HTML-Dokument

Natürlich konnte man ein HTML-Dokument gleichen Inhaltes auch komplizierter schreiben – doch erforderlich war das nicht. Dass die Sprache dementsprechend einfach zu erlernen war und wegen der liberalen Handhabung auch unkorrekt formulierter Dokumente durch die Webbrowser eine **hohe Fehlertoleranz** bestand (und noch besteht!), mag über ihren Siegeszug entschieden haben.

Das »Feature«, das die Erlernbarkeit erleichterte, bestand darin, dass bestimmte Marken oder ganze Elemente gefahrlos weggelassen werden konnten. Ein ent-

² Wie zuvor schon von Vordenkern wie Vannevar Bush, Ted Nelson oder Douglas Engelbart erdacht, propagiert und teilweise umgesetzt.

sprechend »intelligenter« Client sollte in der Lage sein, den Code dennoch zu interpretieren und fehlende Teile gegebenenfalls hinzuzufügen.

Ein solcher Client (der Webbrowser, der hier technisch gesprochen als »Parser« arbeitet) ergänzt das vorstehende HTML-Dokument durch weitere Bereiche, die nur »implizit« vorhanden sind. Erst nach dieser Vervollständigung macht er sich daran, das Dokument darzustellen (die jetzt in Folge angesprochenen Elemente werden in den nächsten Kapiteln ausführlich erläutert).

1.1.2 Verarbeitung und Ergänzung des Quellcodes durch den Parser

Unser in der Tat sehr rudimentär gestaltetes Ausgangsdokument wird von der Parserfunktion des Browsers um die fehlenden Bestandteile ergänzt. Um das gesamte Dokument wird hierfür als **Wurzelement** ein Hüllencontainer gelegt (von »Container« spricht man, weil sich zwischen dessen Start- und Endmarke Inhalte befinden). Das Wurzelement besitzt, wie die Sprache selbst, den Bezeichner »HTML«. Zusätzlich wird das Dokument in einen logischen **Dokumentkopf** und einen logischen **Dokumentrumpf** geteilt, wobei der <TITLE>-Container dem Dokumentkopf zugeschlagen wird. Der Kopfbereich wird in einen Container <HEAD> eingeschlossen, der Rumpf in einen Container <BODY>.

Das Ergebnis, das im Arbeitsspeicher des Browsers entsteht, sieht folgendermaßen aus (Ergänzungen fett hervorgehoben). Ohne hier technisch werden zu wollen – es handelt es sich um den vom Parser generierten »Zwischencode«:

```
<HTML>
  <HEAD>
    <TITLE>Dies ist ein gueltiges HTML-Dokument</TITLE>
  </HEAD>
  <BODY>
    <P>Es enthaelt einen anklickbaren
    <A HREF="anderes_dokument.html">Verweis</A>
    auf ein anderes Dokument.<BR>
    Eine neue Zeile am Ende.</P>
  </BODY>
</HTML>
```

Listing 1.2 Der durch den Parser ergänzte HTML-Quelltext von liberal.html

Dieser Zwischencode mitsamt den Ergänzungen wird an die folgende Formatierungsstufe weitergegeben und liegt der Darstellung des Dokuments im Browserfenster zugrunde (siehe Abbildung 1.1).

Hierbei werden die Marken selbst aus der Darstellung entfernt; der Browser verwendet sie lediglich als Hinweis, wie der Elementinhalt zu handhaben ist. So

wird der Text in <TITLE> in der Titelleiste des Browserfensters gezeigt, der in <A>-Tags eingeschlossene Bereich wird zum anklickbaren Hyperlink.

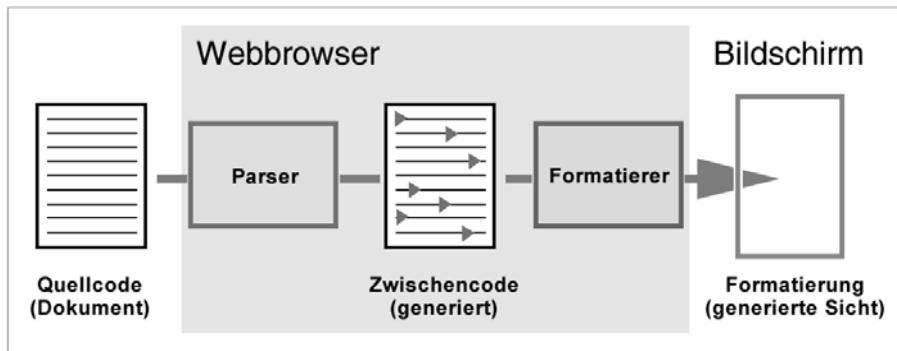


Abbildung 11 Der Webbrowser beinhaltet Parser und Formatierer.

Ein paar weitere Details werden im geparsten Quelltext sichtbar, nämlich die globale Änderung der Schreibweise der Elementnamen hin zu **Großbuchstaben**. Dies erlaubt dem Autor der HTML-Seite, die Bezeichner der Elemente im ursprünglichen Quelltext nach Belieben klein, groß oder gemischt zu schreiben.

Das Gleiche gilt für die Bezeichner so genannter »Attribute«, die Zusatzinformationen in Zusammenhang mit Elementen enthalten. Im obigen Quelltext ist dies der (dort klein geschriebene) Name des `href`-Attributs des Anker-Tags <A>: Auch dieser wird als `HREF` interpretiert. Beachten Sie, dass der Parser auch die **Anführungszeichen** um den Attributwert ergänzt, der den Dateinamen des Zieldokuments des Verweises enthält. Nicht verändert werden hingegen der Wert des Attributs und die Textinhalte der Elemente (hier der Dokumenttitel und der Textabsatz). Die **Einzelmarke**
 weist den Parser darauf hin, dass er an dieser Stelle in der Ausgabe einen Zeilenumbruch durchführen soll.

Die **Endmarke** </P> des Textabsatzes wird ergänzt und muss vom Autor nicht zwangsläufig gesetzt werden. Wollte er einen zweiten Textabsatz (englisch »paragraph«, daher der Elementname <P>) erzeugen, so müsste er allerdings für diesen zumindest eine Startmarke schreiben:

```
...
<p>Ein Textabsatz in HTML.
<p>Ein weiterer Textabsatz.
```

Dies wird vom Parser wiederum ergänzt zu

```
...
<P>Ein Textabsatz in HTML.</P>
<P>Ein weiterer Textabsatz.</P>
```

Die Option, Marken wegzulassen, stammt aus **SGML**³, der Definitionssprache für Markupssprachen, von der Berners-Lee HTML ursprünglich abgeleitet hatte. Start- oder Endmarken von Bereichen (oder sogar beide), können hier als auslassbar (*omittable*) definiert werden. Diese Festlegung wird in einer Sprachgrammatik getroffen, welche die allgemeinen Regeln der Markupssprache enthält. Auch die Syntax dieser Sprachgrammatiken ist in SGML festgelegt.

Da es um die allgemeine Formulierung des Dokumenttyps »HTML« geht, spricht man von einer **Dokumenttyp-Definition** (kurz DTD, für *document type definition*). In dieser Dokumenttyp-Definition ist geregelt, welche Elemente existieren, welche Inhalte für sie zulässig sind und welche Attribute sie besitzen dürfen. Der Parser kennt diese DTD und verwendet ihre Regeln, um ein Dokument zu überprüfen und erforderliche Ergänzungen im Quelltext vorzunehmen.

1.1.3 Die frühen Jahre von HTML

Ganz zu Anfang jedoch gab es eine festgelegte Sprachgrammatik noch nicht – wie Berners-Lee erklärte, war HTML ursprünglich nur an SGML »angelehnt«, nicht aber von diesem korrekt abgeleitet. Dies geschah formal erst im Jahr 1994 anlässlich der Sprachversion **HTML 2.0**, der eine wirkliche Dokumenttyp-Definition zugrunde lag. Initiiert durch Berners-Lee und auf Grundlage der bisherigen Arbeitsgruppen, der Internet Engineering Task Force (IETF), wurde dann das **World Wide Web Committee** (W3C) gegründet. Diese Institution sollte fortan HTML als Sprachstandard pflegen und weiterentwickeln.

Der durchschlagende Erfolg, der auf der Einfachheit von HTML beruhte, hatte in der Zwischenzeit zu einem Boom an Anwendungen und einer Explosion an Webseiten im sich entwickelnden Internet geführt. Von der bloßen Dokumentation von Information, möglichst ohne ihre Darstellung festzulegen (das war Berners-Lees Anfangsgedanke), verlagerte sich der Schwerpunkt in Richtung auf eine möglichst ansprechend präsentierte Optik der Webseiten.

Der Begriff »Webdesign« erschien auf der Bildfläche. HTML musste nun zunehmend Präsentationsaufgaben übernehmen, die das eigentliche Sprachkonzept zu verwässern drohten. Die Browserhersteller nutzten ihre Marktmacht, um die Webdesigner mit neuen HTML-Elementen zu »versorgen«, die reinen Präsentationszwecken dienten, und etablierten sie als Quasi-Standards, bis auch das W3C

³ Standard Generalized Markup Language, ISO-Standard 8879:1986; siehe <http://www.iso.org>.

sie zähneknirschend akzeptierte. Die Kompromissversion, auf die man sich Ende 1996 einigte, hieß **HTML 3.2**. Die Trennung von Struktur und Präsentation wurde ab diesem Zeitpunkt zunehmend verwässert und unterlaufen.⁴

Das »Feature« der Anfangstage, nämlich die sehr liberale Grammatik von HTML, entpuppte sich nun als Achillesferse. Zwar waren nicht alle Neuerungen jener Zeit per se »schlecht«, aber das Bedürfnis nach maschineller Verarbeitung von Informationen aus dem Internet stieg. Die Nutzergruppe erweiterte und verlagerte sich sozusagen – weg vom menschlichen Nutzer, der die Fähigkeit der Interpretation von Inhalten besitzt, hin zu auswertenden Computerprogrammen, die zur Verarbeitung eine genau geregelte Datenstruktur benötigen.

1.2 HTML 4.0 und CSS

Die Folgerung bestand darin, **Struktur** und **Präsentation** der Information wieder streng zu trennen. Die vom W3C propagierte Lösung bestand in zwei Sprachen, die jeweils ausschließlich einen der beiden Aspekte übernehmen sollten. Die eine davon war eine Neufassung von HTML, die andere nannte sich CSS.

1.2.1 HTML 4.0 und seine drei DTD-Varianten

Die Struktur Aufgabe sollte ein »gereinigtes« HTML in einer neuen Version **HTML 4.0** übernehmen, das strikt auf Präsentationselemente verzichtete. Hierfür wurde als grammatische Grundlage eine radikal entmüllte, als »**strict**«-**DTD** bezeichnete Fassung erstellt. Diese Version entspricht der orthodoxen Auffassung des W3C von einer Markupsprache.

Ein vollständiger Bruch hätte jedoch die bis dahin erstellten Webseiten zur Ungültigkeit verdammt. Aus Gründen der Abwärtskompatibilität entschied man sich, für HTML 4.0 weiterhin eine »**transitional**«-**DTD** mit allen Präsentationselementen zur Verfügung zu stellen, sowie eine dritte Version, die »**frameset**«-**DTD**, mit der ein damals neues Feature, die HTML-Framesets, formuliert werden konnten. Die hier enthaltenen Elemente und Attribute, soweit sie nur Präsentationszwecken dienen, wurden jedoch als »deprecated« (missbilligt) gekennzeichnet und ihre zukünftige Entfernung vorbehalten.

Diese Spaltung gilt auch für das wenig später veröffentlichte Versionsupdate **HTML 4.01**, das die derzeit **aktuellste Sprachversion** von HTML darstellt.

⁴ Die sehr interessante Geschichte von HTML und der verschiedenen Versionen der Sprache ist in Kapitel 12, *Sprachfamilien von HTML*, genauer beschrieben.

Aufspaltung in drei Sprachversionen ab HTML 4.0

Seit 1997 zerfällt HTML in drei Sprachzweige:

- ▶ »transitional« – enthält Präsentationselemente
- ▶ »frameset« – enthält Präsentations- und Framesetelemente
- ▶ »strict« – die orthodoxe Variante ohne Präsentationselemente

1.2.2 Die Präsentationssprache CSS

Zur Formulierung von Darstellungsanweisungen wurde eine »Präsentationssprache« entworfen, die den Namen: **CSS** (Cascading Style Sheets) erhielt. Auch diese Idee war nicht neu; das W3C hatte sie bereits 1996 zu Zeiten von HTML 3.2 propagiert. Doch die Webdesigner wollten ungern von ihren mittlerweile gewohnten Präsentationselementen lassen, die Browserhersteller wiederum taten sich schwer mit der Unterstützung von CSS. Dennoch schien der Zeitpunkt für CSS nun günstig, da der Präsentationssprache jetzt in Form von HTML 4.0 *strict* eine reine Struktursprache gegenüber stand. Die Spezifikation **CSS 1.0** wurde daher 1999 in überarbeiteter Version erneut veröffentlicht.

Das Prinzip von CSS ist einfach: An die Bestandteile eines HTML-Dokuments, also die Elementcontainer und ihre Inhalte, werden **Darstellungsanweisungen** gebunden. Die Anweisungen betreffen Schriftart und -größe, aber auch Farbe, Hintergrund oder Abmessungen eines Elements. Die »Bindung« erfolgt über eine **Mustererkennung**, der im einfachsten Fall der Bezeichner eines Elements zugrunde liegt. Die folgende Anweisung gilt für <p>-Elemente:

```
p { font-family:arial; }
```

Die Anweisung besagt, dass der Inhalt aller <p>-Container des Dokuments mit dem Schrifttyp Arial darzustellen ist. Sie sehen hier bereits, dass die Syntax von CSS eine völlig andere ist als die von HTML. Dadurch, dass die CSS-Anweisungen im Idealfall gar nicht im HTML-Dokument zu stehen haben (ein Verweis genügt), stellt dies kein Problem dar.

Besteht diese Verbindung zwischen Element und Styleanweisung, so kann im HTML-Code auf Darstellungsanweisungen verzichtet werden:

```
<p>Dieser Textabsatz wird also mit Arial dargestellt.</p>
<p>Und dieser hier auch.</p>
```

Der Vorteil, auf diesem Wege Struktur (also HTML-Code) und Präsentation (also CSS-Anweisungen) vollkommen zu trennen, ist enorm. Bei der Verwendung von

HTML-Präsentationselementen dagegen müssen die Darstellungsanweisungen im HTML-Dokument selbst stehen, was etwa so aussehen würde:

```
<p><font face="arial">
Dieser Textabsatz wird mit Arial dargestellt.</font></p>
<p><font face="arial">Und dieser hier auch.</font></p>
```

Sie sehen, dass hier wesentlich **mehr Quelltext** im HTML erforderlich ist, als die CSS-Anweisung erfordert. Zudem muss die Präsentationsanweisung (diese wird mit dem ``-Element vorgenommen) in jedem Absatz wiederholt werden. Es ist einleuchtend, dass Dokumente hierdurch erstens größer werden, zweitens schwerer zu warten sind (stellen Sie sich vor, Sie müssten die Schriftart beispielsweise überall auf Verdana ändern). Obendrein ist CSS mächtiger als die HTML-Präsentationselemente, was ein weiterer Grund ist, von ihrer Verwendung Abstand zu nehmen.

1.3 Der Schritt von HTML zu XHTML

Relativ früh kristallisierte sich heraus, dass HTML wegen seiner allzu liberalen Grammatik auf Dauer nicht tauglich bleiben würde. Um die Verarbeitung solcher Dokumente durch beliebige Anwendungen sicher zu gewährleisten, musste auf eine grundsätzlich **rigide Grammatik** aufgebaut werden. Da HTML bereits etabliert war, musste dies jedoch in einer Form erfolgen, die bestehende Browser verarbeiten können sollten – Abwärtskompatibilität war gefragt.

Die Lösung fand sich in Form der 1998 aus der Taufe gehobenen Markupsprache **XML**, die ähnlich wie SGML (von der sie im Übrigen ebenfalls abgeleitet ist) die Definition beliebiger weiterer Markupsprachen ermöglicht. In diesem Zusammenhang spricht man von »XML-Anwendungen«. Bereits 1999 erfolgte eine »Nachschöpfung« von HTML als XML-Anwendung: **XHTML 1.0**.

Aber Achtung: HTML und XHTML sind verschiedene Sprachen!

Da sich XML und SGML **unterscheiden**, gilt dies zwangsläufig auch für HTML und XHTML. Es handelt sich um einige Details, die jedoch nicht trivial sind. So bildet XHTML das Vokabular von HTML, das heißt dessen Elemente, deren Inhaltsmodelle und Attribute zwar weitestgehend nach. Doch bereits in der Grundsyntax von XML sind Unterschiede vorprogrammiert.

Ein wesentlicher Unterschied zwischen XHTML und HTML besteht darin, dass XML im Gegensatz zu SGML keine Verkürzungen zulässt, also das Weglassen von Marken. Ein XML-Dokument muss daher immer vollständig sein, weshalb eine Anwendung, die eine XML-Datei liest, wesentlich einfacher programmiert wer-

den kann. Auch ein XHTML-Dokument muss aus diesem Grund »komplett« sein – dieser Zustand wird auch als »wohlgeformt« bezeichnet.⁵

1.3.1 Ein einfaches XHTML-Dokument

Ein zum im Vorfeld vorgestellten »liberalen« HTML-Dokument gleichwertiges XHTML-Dokument ist wesentlich umfangreicher. Es besitzt, wie Sie sehen, auch eine Art »Vorspann«, der dazu dient, der verarbeitenden Anwendung mitzuteilen, welche Sprachversion (»Doctype«) genau hier vorliegt. Hier handelt es sich um die »transitional«-Version der XHTML-DTD.

Der Rest entspricht ansonsten dem Code, den der SGML-Parser im Zuge der durch ihn vorgenommenen Code-Ergänzung erzeugt hat. Für die Vollständigkeit des Quellcodes muss in diesem Falle der Autor selbst gesorgt haben – ein unvollständiges XHTML-Dokument gilt im Gegensatz zu einem (korrekt) verkürzten HTML-Dokument als fehlerhaft:

```
<!DOCTYPE html PUBLIC "-
//W3C//DTD XHTML 1.0 Transitional//EN" "http://www.w3.org/TR/xhtml1/
DTD/xhtml1-transitional.dtd">
<html xmlns="http://www.w3.org/1999/xhtml">
  <head>
    <title>Dies ist ein gueltiges HTML-Dokument</title>
  </head>
  <body>
    <p>Es enthaelt einen anklickbaren
    <a href="anderes_dokument.html">Verweis</a>
    auf ein anderes Dokument.</p>
  </body>
</html>
```

Listing 1.3 Ein nicht gerade »liberales« XHTML-Dokument

Vielleicht fallen Ihnen noch weitere Einzelheiten auf – dass beispielsweise die Elementbezeichner hier konsequent klein geschrieben werden oder dass im Starttag `<html>` ein Attribut `xmlns` mit einer »merkwürdig anmutenden« Zeichenkette hinzugefügt wurde. Dies sind Eigenheiten von XHTML, auf die im folgenden Kapitel genauer eingegangen wird.

Ansonsten gilt:

- ▶ XHTML 1.0 behält alle Element- und Attributnamen aus HTML 4.01, allerdings in Kleinschreibung.

⁵ Verzeihen Sie mir an dieser Stelle die kleine Ungenauigkeit. Im nächsten Kapitel mehr dazu.

- ▶ XHTML 1.0 existiert wie HTML 4.01 ebenfalls in den Varianten »strict«, »transitional« und »frameset«.
- ▶ Weitere Unterschiede zwischen HTML und XHTML beruhen auf den Unterschieden zwischen den »Elternsprachen« SGML und XML.

1.4 Die Gegenwart von HTML und XHTML

Während HTML auf dem Versionsstand 4.01 seit 1999 quasi »eingefroren« weiterbesteht, blieb die Entwicklung von XHTML nicht vollständig stehen. In Form von **XHTML 1.1** wurde im Jahr 2001 die Aufteilung in drei Sprachvarianten ad acta gelegt – es existiert nur in einer Form, die den bisherigen *strict*-DTDs gleicht. Die Sprache wurde formal in so genannte »Module« aufgeteilt, die jeweils eine eng umrissene Funktionalität der Sprache umfassen und jeweils aus einer Untergruppe der Elemente besteht. Aus diesem Grund wird XHTML 1.1 auch als **modulares XHTML** bezeichnet.

Die Module dienen zur Ableitung anderer Markupsprachen für Spezialzwecke, die entweder auf HTML aufbauen oder Teile von HTML beinhalten sollen. Zum anderen wurde festgelegt, dass XHTML 1.1 **als XML zu behandeln** ist (dies hat Konsequenzen für die Handhabung der Dokumente durch den Server wie den Client). Das Gesamtvokabular wurde nur durch wenige Elemente (die »Ruby«-Elemente) erweitert, blieb aber ansonsten gegenüber XHTML 1.0 »strict« konstant. Auf die Frameset-Funktionalität wurde in XHTML 1.1 verzichtet.

Die neue Version ist jedoch nicht als Ablösung von XHTML 1.0 zu verstehen (oder hat sich zumindest nicht in diesem Sinne durchgesetzt), so dass im Augenblick folglich **sieben verschiedene offizielle Versionen von HTML und XHTML** parallel in Gebrauch sind (siehe Abbildung 1.2):

- ▶ drei Versionen von HTML 4.01 (»strict«, »transitional«, »frameset«)
- ▶ drei Versionen von XHTML 1.0 (»strict«, »transitional«, »frameset«)
- ▶ eine Version von XHTML 1.1 (entspräche einer »strict«-Version)

Nur die drei »strict«-Sprachversionen entsprechen der orthodoxen Interpretation von Markupsprachen. Die vier »transitional«- und »frameset«-Versionen von HTML und XHTML beinhalten hingegen Präsentationselemente und -attribute.

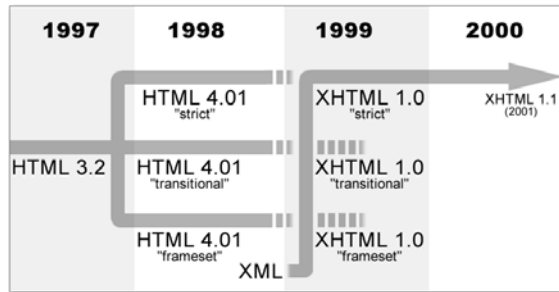


Abbildung 1.2 Die sieben aktuellen Varianten von HTML und XHTML

1.4.1 Abwandlungen von HTML und XHTML

Neben den »Großversionen« von XHTML existieren auch Abwandlungen (basierend größtenteils auf XHTML-Modulen), von denen die wichtigsten **XHTML Basic** (das »minimale« XHTML als so genannte »Host language«) und **XHTML mobile profile** sind. Letzteres ist eine vom W3C zwar abgesegnete, aber nicht initiierte Sprachversion für mobile Clients (Handys, PDAs). Vergleichbare, abgespeckte Versionen von HTML wie cHTML, HTML 4.0 mobile oder iMode HTML waren zum Teil proprietär und können inzwischen als veraltet gelten – doch deren Nachfolger iMode XHTML 1.0 und 1.1 allerdings nicht. Auf dem Mobilsektor zeichnet sich inzwischen jedoch eine Vereinheitlichung in Richtung auf XHTML mobile profile ab.

1.5 Der Schritt von CSS 1.0 zu CSS 2.1

Zu Beginn krankte **CSS 1.0** nicht an der Idee an sich, sondern an der mangelnden Unterstützung durch die Browserhersteller (es waren die Zeiten des »Browserkrieges«, wo man eher darauf achtete, sich voneinander abzusetzen, als allgemeinen Richtlinien zu folgen). Unverdrossen von den Aktualitäten hatte das W3C unterdessen die CSS-Spezifikation fortentwickelt und veröffentlichte 1998 die ambitioniertere und wesentlich umfangreichere Version **CSS 2.0**.

Eine Weile existierten zwei CSS-Versionen parallel, bis sich als unrealistisch herausstellte, von den Browserherstellern die Umsetzung von CSS 2.0 in angemessener Zeit zu erwarten. Man entschied sich stattdessen zurückzurudern – hier lässt sich eine Parallele zum geplanten HTML 3.0 und dem tatsächlich umgesetzten HTML 3.2 feststellen – und tilgte im Jahr 2002 aus CSS 2.0 einige der umstrittenen Bestandteile. Das Ergebnis wurde **CSS 2.1** getauft und stellt in der 2006 aktualisierten Fassung die derzeit aktuelle und auch weitestgehend unterstützte Version von CSS dar.

1.6 Die Zukunft von HTML und XHTML

Wie es genau mit HTML und XHTML weitergehen wird, scheint derzeit (Februar 2007) wieder einigermaßen offen. Sichtbar sind zwei konkurrierende Ansätze des W3C einerseits und eines Konsortiums namens **WHAT WG** andererseits, das sich aus Mitarbeitern von Browserherstellern und ehemaligen W3C-Mitgliedern zusammensetzt.

Die WHAT WG wirft dem W3C-Vorschlag für den HTML-Nachfolger »Weltfremdheit« vor und führt auch gute Gründe an. Dass hier Namen wie Håkon Wium Lie (Opera, CSS-Erfinder), Ian Hickson (Google) und Brendan Eich (Mozilla, JavaScript-Erfinder) vertreten sind, gibt dem Gegenvorschlag weiteres Gewicht. Unter der Bezeichnung **Web Application 1.0** propagiert die WHAT WG eine auf HTML aufbauende, abwärtskompatible Weiterentwicklung der Sprache. Diese soll parallel in einer HTML- und einer XHTML-Version existieren. Landläufig wird dieser Ansatz bereits als **HTML 5.0** bezeichnet.

Das **W3C** hat sich zu einer Neudefinition einer Markupsprache für das Internet entschlossen und diese als **XHTML 2.0** bezeichnet. Hierbei handelt es sich um eine nicht abwärtskompatible Neuschöpfung, die immerhin mit einigen Defiziten von HTML aufräumt und semantisch einen durchdachten Eindruck macht. Die Auslagerung der Frameset- und Formularfunktionalitäten in zwei externe Spezifikationen **XFrames** und **XForms** und das veränderte Vokabular von XHTML 2.0 machen jedoch eine völlig neue Browsergeneration erforderlich, die sich derzeit noch nicht abzeichnet. Obendrein gestaltet sich der Einigungsprozess schwierig – XHTML 2.0 existiert bereits in der achten Vorversion (Working Draft), ohne dass für die nahe Zukunft eine endgültige Spezifikation zu erwarten ist.

Zwischen beiden Fraktionen besteht ein Dialog, aus dem man eine Annäherung der Positionen ableiten kann. Inwiefern das W3C Teile des Vorschlags der WHAT WG für den HTML-Nachfolger übernehmen mag, ist spekulativ. Zumindest aber ist derweil der WHAT WG-Entwurf **Web Forms 2.0**, der in Konkurrenz zu **XForms** des W3C steht, bereits als W3C-Empfehlung aufgetaucht.

Weitere Informationen finden Sie hier:

- ▶ XHTML 2.0: www.w3.org/TR/xhtml2
- ▶ Web Application 1.0: www.whatwg.org/specs/
- ▶ Web Forms 2.0: www.w3.org/TR/web-forms-2/

1.7 Die Zukunft von CSS

Auch in Sachen CSS arbeitet das W3C seit 2001 an einer Weiterentwicklung, die unter der Bezeichnung CSS 3.0 firmiert. Analog zur Modularisierung von XHTML ist CSS 3.0 in verschiedene funktionale Module aufgespalten (es sind genauer gesagt 21 derartige Module), die jeweils von einer eigenen Arbeitsgruppe betreut werden.

Teile der geplanten Spezifikation sind in einigen Browsern vorwegnehmend bereits implementiert, von einer Fertigstellung als Standard ist CSS 3.0 indes noch weit entfernt. Dies mag an dem (wieder einmal) höchst ambitionierten Ansatz liegen, der einen enormen Umfang der Gesamtspezifikation zur Folge hat. Die Arbeit an den Modulen geht dabei unterschiedlich schnell voran – einige liegen seit mehreren Jahren brach, andere werden regelmäßig aktualisiert. Eine praktische Auseinandersetzung mit CSS 3.0 scheint zum gegenwärtigen Zeitpunkt noch nicht sinnvoll, die theoretisch sich abzeichnenden Möglichkeiten sind jedoch nichts weniger als faszinierend.

CSS aktuell: www.w3.org/Style/CSS/current-work

1.8 Zusammenfassung

Aus diesem Kapitel sollten Sie folgende Kernaussagen mitnehmen:

- ▶ Im Idealfall sollen Struktur und Darstellung von Information voneinander getrennt werden.
- ▶ Markupssprachen dienen der Kennzeichnung der Struktur von Dokumenten.
- ▶ Präsentationssprachen dienen der Beschreibung der Darstellung der Informationsstruktur.
- ▶ Dokumentteile werden durch Setzen von Marken (Tags) begrenzt, wobei zu Beginn eines Bereiches eine Startmarke und am Ende eine Endmarke gesetzt wird.
- ▶ Der Quellcode einer Markupssprache wird durch einen Parser verarbeitet, der das Ergebnis der Verarbeitung zur Darstellung weiterreicht.
- ▶ Für HTML-Dokumente geschehen Parsingvorgang und Darstellung innerhalb des Browsers.
- ▶ Ein SGML-konformer Parser ergänzt fehlende Marken. In SGML ist das Weglassen von Marken erlaubt, wo es die Grammatik des Dokumenttyps zulässt.

- ▶ HTML ist von SGML abgeleitet. In HTML dürfen bestimmte Marken daher weggelassen (verkürzt) werden. Weggelassene Marken werden durch den Browser vor der Darstellung ergänzt.
- ▶ Ein XML-konformer Parser ergänzt keine Marken, sondern meldet einen Fehler. Die Grundsyntax von XML erlaubt das Weglassen von Marken grundsätzlich nicht.
- ▶ XHTML ist von XML abgeleitet. In XHTML dürfen daher keine Marken weggelassen werden. Die daher obligatorische Vollständigkeit eines Dokuments erleichtert jedoch die Verarbeitung durch Software.
- ▶ HTML und XHTML dienen zur Formulierung der Struktur eines Dokuments. Beide Sprachen besitzen den gleichen Umfang an festgelegten Elementen und Attributen.
- ▶ CSS dient zur Formulierung von Darstellungsanweisungen für ein HTML-oder XHTML-Dokument. Die aktuelle Version CSS 2.1 wird weitestgehend unterstützt.
- ▶ HTML 4.0 und XHTML 1.0 existieren aus historischen Gründen in den drei Varianten »strict«, »transitional« und »frameset«. XHTML 1.1 existiert nur in einer einzigen Variante, die »strict« entspricht.
- ▶ Die als HTML 4.0 »strict« und XHTML 1.0 »strict« bezeichneten Varianten sowie XHTML 1.1 entsprechen der orthodoxen Vorstellung einer Markup-sprache, die auf Präsentation verzichtet. Die anderen beiden Varianten enthalten Präsentationselemente, auf die besser verzichtet wird.

Wie geht es weiter?

Im folgenden Kapitel 2, *(X)HTML – die Syntaxgrundlagen*, wird näher auf die **Grundsyntax** von HTML und XHTML und die **Unterschiede** zwischen beiden Sprachen eingegangen. Sie erhalten auch einen (notwendigerweise knappen) Einblick in die Formulierung von SGML- und XML-konformen DTDs. Ebenfalls vertieft wird die **Syntax von CSS**, wobei auf die Formulierung der Darstellungsanweisungen und die Funktionsweise der Mustererkennung eingegangen wird, die eine Präsentationsanweisung mit einem Zielelement verbindet.

Dieses Kapitel beschreibt die Verwendung von XHTML-Elementen zur Gestaltung des eigentlichen Dokumentinhalts. Angesprochen werden Überschriften und Textabsätze zur Gliederung, sowie Inlinenelemente zur Textgestaltung, aber auch Hyperlinks, Grafiken, Tabellen, Formulare und Listen.

6 (X)HTML im Contentbereich

Nachdem im vorangegangenen Kapitel die Grundlagen eines Dokumentlayouts beschrieben wurden (daran wird im folgenden Kapitel 7, *Navigation*, wieder angeknüpft werden), soll nun näher auf die Gestaltung des eigentlichen Dokumentinhalts und der hierfür zur Verfügung stehenden HTML-Elemente und CSS-Anweisungen eingegangen werden. Thema sind daher Elemente zur Textformatierung und Textauszeichnung, aber auch das Einbinden von Medienobjekten, oder die Erstellung und Gestaltung von Datentabellen und Formularen. Auch die Verwendung von Hyperlinks im Dokument wird vertieft.

6.1 Contentmarkup mit Textabsätzen und Überschriften

Die grundlegenden Elemente zur Inhaltsgestaltung wurden bereits erwähnt – es handelt sich um die, in Kapitel 3, *(X)HTML – die Dokumentstruktur* vorgestellten Blockelemente, wobei zunächst die Überschriften `<h1>` bis `<h6>` und der Textabsatzcontainer `<p>` im Zentrum der Betrachtung stehen sollen. Diese Container gliedern Textinhalte in logische Abschnitte, besitzen allerdings daneben auch Präsentationseigenschaften. Einige Einstellungen, die diese Grundeigenschaften, aber auch allgemeine Textdarstellung betreffen, können bereits auf globaler Ebene erfolgen.

6.1.1 Mögliche globale Grundeinstellungen per CSS

Eine Reihe von HTML-Elementen, darunter auch der Textabsatzcontainer `<p>` sowie alle Überschriftencontainer, besitzen vordefinierte Außenabstände (*margins*), die für einen Abstand zum Vorgänger- oder Folgecontainer sorgen. Sofern Sie diese Grundeigenschaften nicht beibehalten, sondern individuell festlegen möchten, ist es erwägenswert, diese zunächst global auf Null zu setzen.¹ Das glei-

che gilt für Innenabstände (*padding*s), die für einige Elemente in der Darstellung einiger Browser ebenfalls ungleich Null sind.

Nullung der Default-Margins

Für eine globale **Nullung der Default-Margins** der HTML-Präsentation setzen Sie folgende CSS-Anweisung an den Beginn Ihres Stylesheets:

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

Der Stern, der hier als Selektor dient, ist ein Platzhalter für einen beliebigen Typselektor, erfasst also alle Elemente ungeachtet ihres Bezeichners. Man nennt den Stern-Selektor deshalb auch den **universellen Selektor**. Die auf diese Weise erfassten Elemente sind in der folgenden Tabelle aufgelistet.

Beachten Sie, dass Sie nun selbst dafür Sorge tragen müssen, dass im Rahmen des Layouts erforderliche Abstände definiert werden. (Im Gegenzug kommen Ihnen allerdings nicht unvermuteterweise Defaulteigenschaften ins Gehege.) Die **zusätzliche Nullung der Paddings** vereinheitlicht das Verhalten der verschiedenen Browser gegenüber den Elementen – laut den offiziellen Spezifikationen existierten zwar keine Default-Paddings (obwohl sich `<body>` und `<td>` sowie einige weitere Elemente browserspezifisch so verhalten), es wird jedoch in keinem Fall Schaden angerichtet.

Element	Defaultmargin	Element	Defaultmargin
<code><body></code>	8px	<code><p></code>	1.12em 0
<code><h1></code>	0.67em 0	<code><blockquote></code>	1.12em 40px
<code><h2></code>	0.75em 0	<code><form></code>	1.12em 0
<code><h3></code>	0.83em 0	<code><fieldset></code>	1.12em 0
<code><h4></code>	1.12em 0	<code></code> , <code></code>	1.12em 0 1.12 em 40px
<code><h5></code>	1.5em 0 1	<code><menu></code> , <code><dir></code>	1.12em 0 1.12 em 40px
<code><h6></code>	1.67em 0	<code><dl></code>	1.12em 0 1.12 em 40px

Tabelle 6.1 Defaultmargins gemäß W3C-Spezifikation (tatsächliche Umsetzung variiert)

Globale Grundeinstellungen für den `<body>`-Container

Einige Grundeinstellungen bezüglich der im Dokument einzusetzenden **Schriften** können im `<body>`-Container global für das Dokument vorgenommen werden. In erster Linie betrifft dies den zu verwendenden **Font**, also die Schriftart. Hierzu dient das Property `font-family`, das Ihnen bereits aus Kapitel 4, CSS-

1 Einen radikaleren Ansatz bietet Tantek Çelik in: <http://tantek.com/log/2004/undohtml.css>.

Grundsyntax, her bekannt ist. Sie können mit dem `color`-Property zusätzlich auch eine **Schriftfarbe** festlegen.

```
body {
  /* Defaultschriftfarbe: */
  color: #000;
  /* Defaulthintergrundfarbe: */
  background-color: #fff;
  /* Defaultschrift: */
  font-family: arial, helvetica, sans-serif;

  /* verhindert Bug bei relativen Schriftgrößen: */
  font-size: 100.01%;
}
```

Grundsätzlich sollte für Elemente, für die eine Schriftfarbe festgelegt wird, auch eine **Hintergrundfarbe** bestimmt werden (der W3C-CSS-Validator würde das Fehlen einer solchen Definition monieren), wobei auf einen guten Kontrast geachtet werden muss. Hier wurden die Defaultfarben, die der Browser verwendet, explizit gesetzt.

Sofern **Hintergrundgrafiken** eingesetzt werden, gilt entsprechendes für die Lesbarkeit der Schrift vor dem gewählten Hintergrundbild (in diesem Fall sollte dennoch die Definition einer Hintergrundfarbe erfolgen).

Für eine **gleichmäßigere optische Wirkung** des Schriftbildes kann es sinnvoll sein, fett gesetzten Textpassagen (Überschriften oder Betonungen) einen etwas helleren (bzw. dunkleren) Farbton zu geben, um ihnen an Gewicht zu nehmen.

Ist als Defaultschriftfarbe **schwarz** (`#000`) gewählt, so kann hierfür ein dunkles Grau eingesetzt werden.

```
/* Dunkelgrau: */
h1, h2, h3, h4, h5, h6, b, strong { color:#333; }
```

Analog nimmt man bei Defaultschriftfarbe **weiß** (`#fff`) durch Einsatz eines hellen Grautones den fett gesetzten Passagen etwas Härte:

```
/* Hellgrau: */
h1, h2, h3, h4, h5, h6, b, strong { color:#eee; }
```

Last not least kann eine Definition für die Schriftgröße erfolgen, die dann als **Basisschriftgröße** für die Container innerhalb von `<body>` gilt. Hier gibt es zwei Ansätze – Sie können hier eine prozentuale Angabe machen, oder eine in **em**. Im obigen Beispiel ist eine **prozentuale Angabe** gemacht, die die Ausgangsschriftgröße (beinahe) beibehält. Die Definition

```
font-size: 100.01%;
```

beugt Bugs in älteren Browsern (Internet Explorer 5.x und Opera 6.0) vor, die zu Rundungsfehlern bei weiter innen liegenden prozentualen Schriftgrößenangaben führt. Ebenfalls denkbar (ohne jedoch den angesprochenen Bug zu vermeiden) ist eine Angabe in **em**, wobei die Basisschriftgröße auch vergrößert oder (eher erforderlich) verkleinert werden kann:

```
font-size: 0.95em; /* Basisschriftgröße nun ca. 15px */
```

6.1.2 Die Überschriften

Überschriften betiteln längere Texte nicht nur, sondern gliedern sie auch. Die zwingende Zuordnung eines Textabsatzes zu einer Hierarchiestufe (im Sinne einer Dokumentgliederung in Abschnitte) ist in HTML zwar nicht modellierbar, jedoch sollte man Überschriften grundsätzlich in folgendem Sinne einsetzen:

- ▶ Die Überschrift erster Ordnung betitelt den Dokumentinhalt
- ▶ Überschriften zweiter, dritter und weiterer Ordnungen betiteln der Reihe nach nachfolgende, niederrangigere Dokumentabschnitte
- ▶ Bei der Betitelung soll keine Überschriftenstufe übersprungen werden

Es ergibt sich eine Hierarchiefolge, wie sie in Abbildung 6.1 angedeutet ist. Achtung – die Umrandungen bezeichnen die Hierarchiestufe, nicht den Bereich des Überschriftencontainers! Beachten Sie, dass daher **keine wirkliche Zuordnung** der `<p>`-Container zu einer voranstehenden Überschrift existiert.

Erst XHTML 2.0 verspricht dieses Problem durch Einführung eines `<section>`-Containers zu lösen², der Überschriften und Absätze einer Stufe zusammenfasst (Sie könnten dies natürlich durch `<div>`-Container oder auf vergleichbarem Wege auch in HTML halbwegs nachbauen – früher wurde zu diesem Zweck auch die horizontale Trennlinie `<hr>` eingesetzt).

Die Beachtung der Überschriftenhierarchie ist eine Forderung der **Barrierefreiheit** (laut WAI 3.5, BITV 3.5), da Überschriften von Screenreadern und vergleichbaren Hilfsmitteln auch zur Kenntlichmachung der Dokumentgliederung und als Navigationshilfe einsetzen.

Ebenfalls abzulehnen ist der Einsatz von Überschriften aus Präsentationsgründen (Textvergrößerung). Analog sollte nicht vergrößerter Text an Stelle einer Überschrift eingesetzt werden. CSS bietet hinreichend Mittel, um Schriftgrößen gegebenenfalls anzupassen.

² Siehe: http://www.w3.org/TR/xhtml2/mod-structural.html#edef_structural_section.

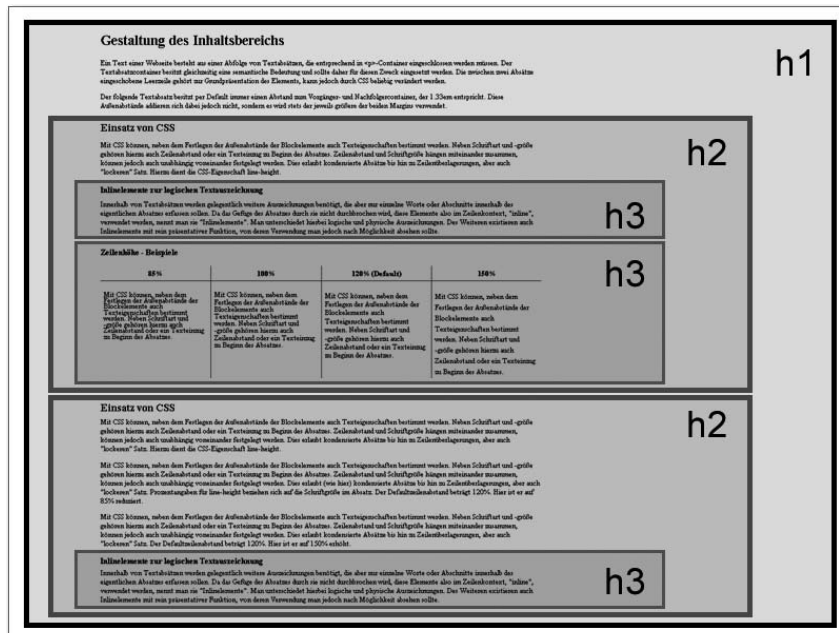


Abbildung 6.1 Überschriftenhierarchie in einem HTML-Dokument

Praxistipp – Überschriften als Linkziel

Wenn Sie den Überschriften Ihres Dokuments konsequent `id`-Attribute zuweisen, können Sie auf einfachen Wege **Sprunglinks** zu einzelnen Dokumentabschnitten erstellen.

6.1.3 Der Textabsatzcontainer

Ein Text einer Webseite besteht aus einer Abfolge von Textabsätzen, die entsprechend in `<p>`-Container eingeschlossen werden müssen. Der Textabsatzcontainer besitzt gleichzeitig eine **semantische Bedeutung** und sollte daher nur für diesen Zweck eingesetzt werden.

Die zwischen zwei Absätzen eingeschobene Leerzeile gehört zur Grundpräsentation des Elements, die jedoch durch CSS beliebig verändert werden kann. Der folgende Textabsatz besitzt per Default immer einen **Abstand** zum Vorgänger- und Nachfolgercontainer, der 1,2em entspricht, also auf der aktuell für den Container gültigen Schriftgröße basiert.

Die offiziellen Defaultmargins für den `<p>`-Container lauten wie folgt:

```
p { margin: 1.12em 0; }
```

... was sich als »Außenabstand von 1,12em nach oben und unten sowie Außenabstand 0 nach rechts und links« übersetzen lässt. Sie können selbstverständlich beliebige andere Werte vergeben, sofern Ihr Layout dies erfordert.

Die **Außenabstände** nach oben und unten zweier aufeinanderfolgender Container addieren sich dabei jedoch nicht, sondern es wird stets der jeweils größere der beiden Margins verwendet (dies gilt beispielsweise, wenn ein Textabsatz auf eine Überschrift folgt, die einen größeren Bottommargin-Wert besitzt).

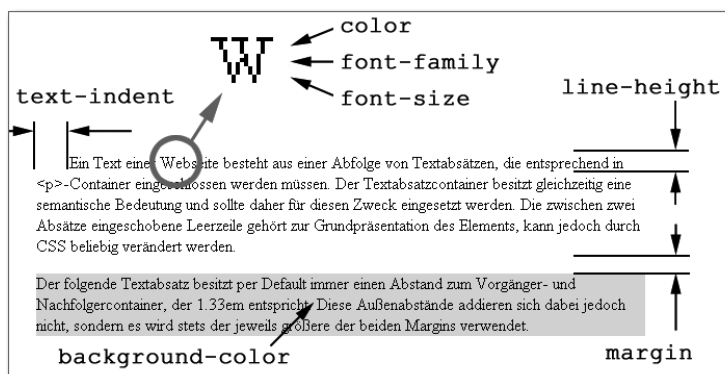


Abbildung 6.2 Die wichtigsten CSS-Parameter für Textabsätze

Mit CSS können, neben dem Festlegen der Außenabstände der Blockelemente auch Texteigenschaften bestimmt werden. Neben Schriftart und -größe gehören hierzu auch **Zeilenabstand** oder ein **Texteinzug** zu Beginn des Absatzes. Zeilenabstand und Schriftgröße hängen miteinander zusammen, können jedoch auch unabhängig voneinander festgelegt werden. Dies erlaubt eng gesetzte Absätze (»kompress«) bis hin zu Zeilenüberlagerungen, aber auch »lockeren« Satz. Hierzu dient die CSS-Eigenschaft `line-height`.

Angaben für `line-height` beziehen sich auf die Schriftgröße innerhalb des Absatzes. Die Defaultzeilenhöhe beträgt 120 % der Schriftgröße. Eine Prozentangabe unter 120 % reduziert daher die Zeilenhöhe, eine Angabe darüber vergrößert sie. Grundsätzlich sollte man mit einer Reduzierung der Zeilenhöhe vorsichtig umgehen, da dies schnell die Lesbarkeit des Textes erschwert (der Wert 100 % sollte nicht unterschritten werden). Eine Erhöhung ist im Allgemeinen unkritischer, sofern die Zeilenlänge nicht zu kurz ist, da der Textblock oberhalb von 150 % sonst optisch zu zerfallen droht (siehe Abbildung 6.3).

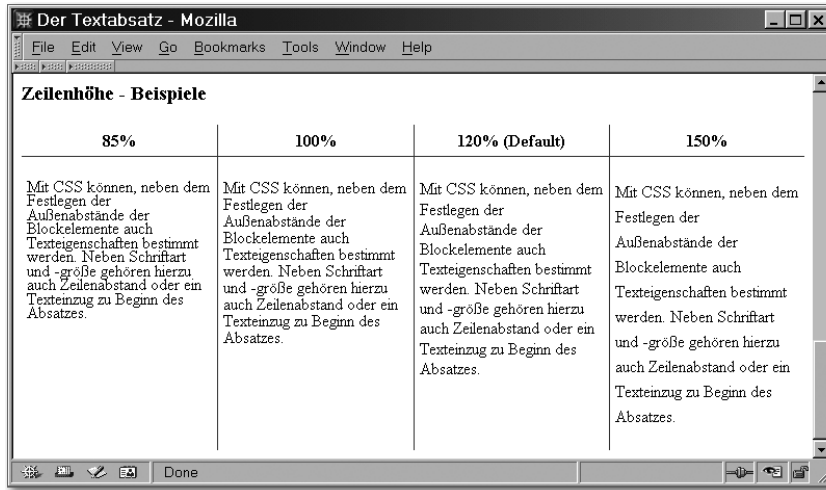


Abbildung 6.3 Verschiedene Zeilenhöhen mit line-height im Vergleich

6.2 Inlinenelemente zur Textauszeichnung

Innerhalb von Textabsätzen werden gelegentlich weitere Auszeichnungen benötigt, die aber nur einzelne Worte oder Abschnitte innerhalb des eigentlichen Absatzes erfassen sollen. Da das Gefüge des Absatzes durch sie nicht durchbrochen wird, diese Elemente also im Zeilenkontext, »inline«, verwendet werden, nennt man sie **Inlinenelemente**.

Man unterscheidet hierbei **logische** und **physische** Auszeichnungen, welche Rolle oder Bedeutung ihrer Inhalte kennzeichnen, ohne damit notwendigerweise eine Präsentation festzulegen. Des Weiteren existieren auch Inlinenelemente mit rein präsentativer Funktion, von deren Verwendung man jedoch nach Möglichkeit zugunsten von CSS absehen sollte.

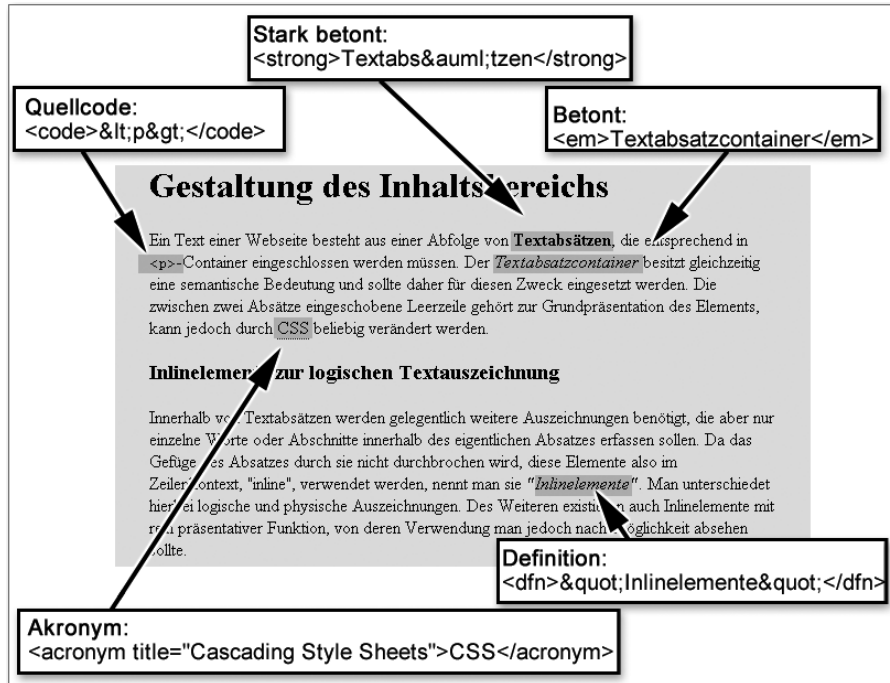


Abbildung 6.4 Verschiedene logische Textauszeichnungen

Logische und physische Auszeichnung

In der folgenden Tabelle sind Inlineelemente zur logischen und physischen Textauszeichnung aufgelistet. Ausführliche Erläuterungen zu den einzelnen Elementen finden Sie in Kapitel 14, *(X)HTML-Elementreferenz*, in den angegebenen Abschnitten – an dieser Stelle möchte ich mich auf grundlegendere Aspekte beschränken.

Element	Rolle	Erläuterung
<code><cite> ... </cite></code>	Bezeichnet die Quelle (Herkunft) eines Zitats	Abschnitt 14.18
<code><code> ... </code></code>	Bezeichnet Quellcode; gewöhnlich mit nicht proportionaler Schrift dargestellt	Abschnitt 14.19
<code><dfn> ... </dfn></code>	Bezeichnet eine Definition; gewöhnlich kursiv dargestellt	Abschnitt 14.24
<code> ... </code>	Einfache Hervorhebung; gewöhnlich kursiv dargestellt	Abschnitt 14.29
<code><kbd> ... </kbd></code>	Bezeichnet eine Tastatureingabe;	Abschnitt 14.46

Tabelle 6.2 Inlineelemente zur logischen und physischen Textauszeichnung

Element	Rolle	Erläuterung
<code><q> ... </q></code>	Bezeichnet ein Inlinezitat durch Anführungszeichen	Abschnitt 14.67
<code><samp> ... </samp></code>	"sample"; Auszug z. B. eines Programmquellcodes	Abschnitt 14.69
<code> ... </code>	Bezeichnet starke Hervorhebung; gewöhnlich fett dargestellt	Abschnitt 14.75
<code><var> ... </var></code>	Bezeichnet eine Variable	Abschnitt 14.91
Attribute (alle)	class, id, style, title (Universalattribute)	

Tabelle 6.2 Inlinenelemente zur logischen und physischen Textauszeichnung (Forts.)

Die hier vorgestellten Inlinenelemente haben gemeinsam, dass sie nicht auf eine spezielle Präsentation ihrer Inhalte abzielen (obwohl diese teilweise durchaus damit verbunden ist), sondern eine Aussage über ihre Bedeutung machen. Dies macht ihren Einsatz prinzipiell **unabhängig vom auswertenden User Agent** – ist dieser beispielsweise ein Screenreader, so kann er beispielsweise eine Betonung mit `` auswerten (durch eine Erhöhung der Leselautstärke), wohingegen eine einfache Fettauszeichnung mit `` keine zwingende semantische Bedeutung besitzt.

Element	Rolle	Erläuterung
<code> ... </code>	Neutraler Container; wichtig in Zusammenhang mit Stylesheets	Abschnitt 14.73
Attribute	class, id, style, title (Universalattribute)	

Tabelle 6.3 Der ``-Container

Eine Sonderstellung in diesem Kontext bildet der neutrale ``-Container, der als Hauptaufgabe das Tragen einer CSS-Klasse übernimmt, also eine Möglichkeit darstellt, eine reine Styleinformation ohne Zweckentfremdung eines anderen Elemente an einen Inlinebereich zu binden. Aus diesem Grunde könnte man seine Wirkung als »präsentativ« bezeichnen, jedoch kann das Element über entsprechende Klassennamen auch logische Bedeutungen vermitteln:

```
<p>Bauteil <span class="bestellnummer">08-15</span>:
<span class="preis">22.95</span></p>
```

Auch die **Barrierefreiheit** eines Textes lässt sich durch Auszeichnung mit logischen Elementen verbessern. Hier sind in erster Linie die beiden Elemente `<abbr>` und `<acronym>` zu nennen, die neben der Kennzeichnung auch gleich die Langfassung eines Begriffes zur Verfügung stellen (über ihr `title`-Attribut). Ein Screenreader kann so eine »unaussprechliche« Abkürzung durch Klartext erset-

zen, ein normaler Browser markiert die Begriffe lediglich als »mit Erläuterung hinterlegt« (dies geschieht durch Unterpunktion; die Erläuterung erscheint bei Mouseover als Tooltip – beachten Sie jedoch, dass der Internet Explorer das Element `<abbr>` erst ab Version 7 unterstützt).

Element	Rolle	Erläuterung
<code><abbr> ... </abbr></code>	Auszeichnung als Abkürzung	Abschnitt 14.2
<code><acronym> ... </acronym></code>	Auszeichnung als Akronym	Abschnitt 14.3
Attribute (alle)	class, id, style, title (Universalattribute)	

Tabelle 6.4 Logische Inlinenelemente zur Unterstützung der Barrierefreiheit

Zwei weitere logische Elemente dienen der Anbringung von **Änderungsmarken** im Text, denen auch Urheberangaben und Datum hinzugefügt werden können. Es existiert zwar eine Defaultpräsentation, die als gelöscht gekennzeichnete Passagen durchgestrichen und eingefügte mit Unterstrich darstellt, jedoch kann mit CSS auch eine beliebig andere Darstellung erzwungen werden. Folgende CSS-Befehle blenden gestrichene Passagen aus und highlighten Einfügungen durch gelben Hintergrund (der Unterstrich wird entfernt):

```
del { display:none; }
ins { text-decoration:none; background-color: yellow; }
```

Element	Rolle	Erläuterung
<code> ... </code>	"deleted"; markiert einen Textabschnitt als gestrichen	Abschnitt 14.23
<code><ins> ... </ins></code>	"inserted"; markiert einen Textabschnitt als hinzugefügt	Abschnitt 14.44
Attribute (alle)	cite, datetime	

Tabelle 6.5 Logische Inlinenelemente für Änderungsmarkierungen

Präsentative Textauszeichnung

Weniger gern gesehen ist heutzutage eine Auszeichnung, die auf eine **reine Präsentationswirkung** ausgerichtet ist – man kann mit ihnen zwar in normalen Browsern eine zu logischer Auszeichnung identische optische Wirkung erzielen, auf der Strecke bleibt jedoch die semantische Bedeutung der Inhalte.

Beispielsweise ist die Umsetzung von `Textabsätzen` und `Textabsätzen` im Browser identisch, ebenso die von `<code><p></code>` und `<tt><p></tt>`. Wo `` eine Wichtigkeit im Sinne einer Betonung festlegt, markiert `` lapidar eine (prinzipiell aus-

sagefreie) Fettauszeichnung. Das Element `<code>` weist seinem Inhalt die Bedeutung »Quelltextausschnitt« zu, wohingegen `<tt>` lediglich nicht-proportionale Schrift fordert – ohne Begründung hierfür.

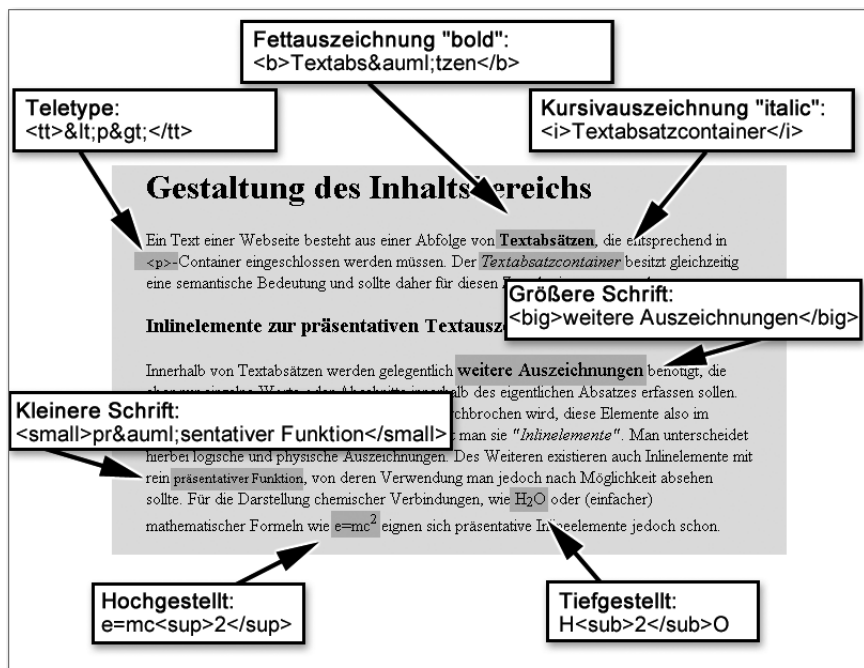


Abbildung 6.5 Verschiedene präsidentative Inlinenelemente

Generell ist gegen präsidentative Auszeichnung nichts einzuwenden, solange damit keine Aussage über die Bedeutung der Inhalte verbunden werden soll: User Agents, die nicht visuell orientiert sind, werden diese Elemente nicht sinnvoll auswerten können und gegebenenfalls ignorieren. Screenreader können das *Vorhandensein* von Präsentationsmarkup zwar ansagen, müssen dessen Interpretation (»Warum ist dieser Abschnitt fett?«) jedoch dem Nutzer überlassen.

Element	Rolle	Erläuterung
<code> ... </code>	"bold"; Fettauszeichnung;	Abschnitt 14.7
<code><big> ... </big></code>	erhöht die Schriftgröße um eine Stufe (auf ca. 120 %)	Abschnitt 14.11
<code><i> ... </i></code>	"italic"; Kursivauszeichnung	Abschnitt 14.40
<code><small> ... </small></code>	reduziert die Schriftgröße um eine Stufe (auf ca. 80 %)	Abschnitt 14.72

Tabelle 6.6 Textauszeichnungen mit vorwiegend präsidentativem Charakter

Element	Rolle	Erläuterung
<code><sub> ... </sub></code>	"subscript"; Inhalt wird tiefgestellt und kleiner präsentiert	Abschnitt 14.77
<code><sup> ... </sup></code>	"superscript"; Inhalt wird hochgestellt und kleiner präsentiert	Abschnitt 14.78
<code><tt> ... </tt></code>	"teletype"; Inhalt wird mit nicht proportionaler Schrift dargestellt	Abschnitt 14.88

Tabelle 6.6 Textauszeichnungen mit vorwiegend präsentativem Character (Forts.)

Es existieren noch weitere Elemente, die entweder aus den Anfangstagen von HTML stammen und sich überlebt haben, oder missverständlich sind. Ein Paradebeispiel ist der ``-Container, dessen Aufgabe der Schriftformatierung gänzlich durch CSS übernommen werden können (gegebenenfalls durch einen ``-Container in Verbindung mit einer CSS-Klasse). Zu nennen sind auch `<s>` und `<strike>` (hier nicht aufgeführt), an deren Stelle besser die Änderungsmarke `` eingesetzt werden sollte, oder die Unterstreichungs-marke `<u>`, die zur Verwechslung des Inhaltes mit einem Hyperlink führen kann und deren Wirkung, wenn nötig, ebenfalls durch CSS nachgebildet werden sollte.

Element	Rolle	Erläuterung
<code> ... </code>	Präsentiert den eingeschlossenen Textinhalt durch Angabe von Schriftgröße, -typ und -farbe	Abschnitt 14.32
<code><s> ... </s></code>	"strikethrough";präsentiert den eingeschlossenen Inhalt durchgestrichen	Abschnitt 14.68
<code><u> ... </u></code>	"underline"; präsentiert den eingeschlossenen Textinhalt mit Unterstrich	Abschnitt 14.89

Tabelle 6.7 Präsentationselemente, die nicht länger eingesetzt werden sollten

6.3 Der Hyperlinkcontainer, revisited

Bereits angerissen wurde der **Hyperlinkcontainer** `<a>` der zur Erstellung von Verknüpfungen zwischen Dokumenten dient. Der »Ankercontainer«, wie er auch genannt wird, zählt ebenfalls zu den Inlineelementen. Der Begriff »Anker« stammt dabei aus der Hypertexttheorie, wo mit »Ankerpunkt« das Textfragment bezeichnet wird, an welchem die eigentliche Verknüpfung »verankert« wird. Ein Hypertext-Anker besitzt demnach zwei Aspekte³:

³ Zur Theorie siehe: François Fluckiger, *Multimedia im Netz*; Prentice Hall 1996.

1. **Als Ankerpunkt:**

»Ein verknüpfbares Informationsfragment.« (z. B. der Linktext)

2. **Als Verknüpfung:**

»Ein Bezug zu einer Information.« (das Linkziel, z. B. als URI)

In HTML entspricht der Ankerpunkt dem in den Container eingeschlossenen Text, das Verknüpfungsziel wird als Wert des `href`-Attributs definiert:

```
<a href="zieldatei.html">Dies ist der Ankertext</a>
```

Das andere, mit `<a>` früher oft eingesetzte Attribut `name` hat an Bedeutung eingebüßt – es diente dazu, einen Ankercontainer auch zum Ziel eines Links machen zu können. Diese Funktion hat mittlerweile das `id`-Attribut übernommen, mit dem Vorteil, dass nun außer `<a>`-Containern beliebige Elementcontainer als Linkziel fungieren können.

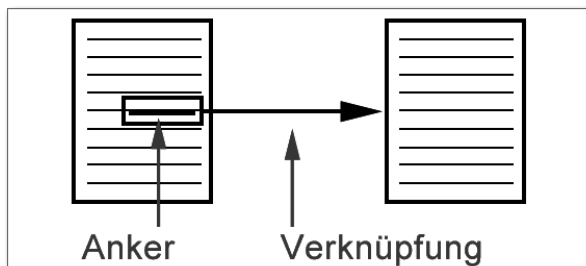


Abbildung 6.6 Der Textinhalt von `<a>` dient als so genannter »Anker«.

Element	Rolle	Erläuterung
<code><a> ... </code>	"anchor"; bezeichnet Quelle oder Ziel eines Hyperlinks	Abschnitt 14.1
Attribute (Auswahl)	accesskey, href, name, tabindex, target, title	

Tabelle 6.8 Der `<a>`-Container zum Erzeugen von Hyperlinks

Die Defaultpräsentation des Textinhalts eines Hyperlinks ist unterstrichen. Innerhalb von Fließtexten sollte dies beibehalten werden – das Styling von Navigationslinks innerhalb von Menüs ist eine andere Sache und wird im folgenden Kapitel vertieft. Kennzeichnen Sie die Links innerhalb des Inhaltsbereichs auf eine Weise, dass sie erkennbar bleiben. Sie können Styleklassen einsetzen, um verschiedene Rollen der Links zu verdeutlichen, wie siteintern, extern oder E-Mail-Link (siehe Abbildung 6.7).

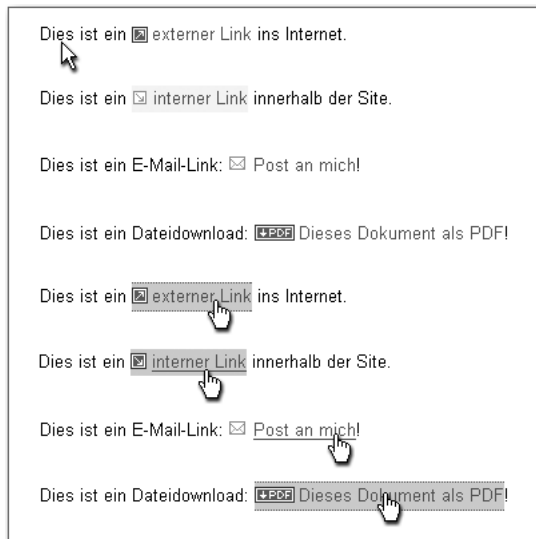


Abbildung 6.7 Hyperlinks mit Icons, unten Hover-Zustände

Den Hyperlinks in Abbildung 6.7 liegt folgender Quellcode zugrunde:

```
<p>Dies ist ein <a href="http:www.weitweg.de"
  class="extern">externer Link</a> ins Internet.</p>

<p>Dies ist ein <a href="andere_seite.html"
  class="intern">interner Link</a> innerhalb der Site.</p>

<p>Dies ist ein E-Mail-Link: <a href="mailto:x@y.de"
  class="mail">Post an mich</a>!</p>

<p>Dies ist ein Dateidownload: <a href="dokument.pdf"
  class="pdf">Dieses Dokument als PDF</a>!</p>
```

Listing 6.1 Verschiedene Hyperlinks mit Klassenattribut

Ohne CSS würden die Links präsentiert, wie in Abbildung 6.8; hier mehrere exemplarische Auszüge der verwendeten CSS-Anweisungen. Die nicht vorgestellten Klassen arbeiten analog, jedoch mit anderen Icon-Grafiken. Entsprechend müssen die `padding`-Werte angepasst werden, um die Breiten zu kompensieren.

```
a.intern {
  text-decoration:none;
  background:#f5f5f5 url('intern.gif')0em 0.2em no-repeat;
  padding:1px 0 3px 16px; color:#D13500;
}
```

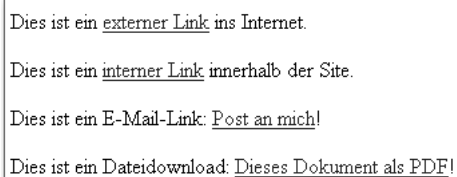
```

a:intern: hover {
  text-decoration: underline;
  background: #ffccbb url('intern_a.gif') 0em 0.2em no-repeat;
}

a:mail {
  text-decoration: none;
  background: url('mail_link.gif') 0em 0.25em no-repeat;
  padding: 1px 0 3px 18px; color: #D13500;
}

a:mail: hover { text-decoration: underline;
}

```



Dies ist ein externer Link ins Internet.

Dies ist ein interner Link innerhalb der Site.

Dies ist ein E-Mail-Link: Post an mich!

Dies ist ein Dateidownload: Dieses Dokument als PDF!

Abbildung 6.8 Die Hyperlinks von Abbildung 6.7 ohne CSS-Präsentation

6.3.1 Welche Stylingmöglichkeiten bietet CSS für Hyperlinks?

Speziell in Zusammenhang mit Hyperlinks kann das `text-decoration`-Property eingesetzt werden, um den oft ungeliebten **Unterstrich** des Links zu entfernen. Innerhalb von Fließtexten ist davon abzuraten, sofern nicht eine andere Methode (wie eine Hintergrundfarbe) eingesetzt wird, den Link optisch zu kennzeichnen. Mittels des `background-image`-Properties kann ein Link ein **Icon** in Form eines Hintergrundbildes erhalten. Schaffen Sie mit dem `padding`-Property genügend Platz, damit der Linktext das Icon nicht verdeckt.

Die interessanteste Stylingmöglichkeit per CSS bietet der Pseudoselektor `:hover`, der Mouseover-Effekte für Links erzeugen kann. Für den Hoverzustand des Links können Sie sämtliche Darstellungsparameter verändern, also Unterstrich einblenden, Farbänderung, anderes Hintergrundbild. Sie erzielen so ohne Einsatz von JavaScript dynamische Linkpräsentationen.

6.3.2 Welche HTML-Attribute werden für Hyperlinks benötigt?

Unabdingbar ist das `href`-Attribut, das die verknüpfte Ressource nennt. Einem barrierefreien Link werden Sie über das `title`-Attribut weitere Information über das Linkziel mit auf den Weg geben. Zusätzlich können Sie mit `accesskey` oder

tabindex die Zugänglichkeit über Tastatur vereinfachen. Im Rahmen von Framesets oder zum einfachen Öffnen neuer Fenster (Vorsicht, dies ist nur bedingt barrierefrei), können Sie das `target`-Attribut einsetzen. Es steht allerdings nur in den *transitional*- und *frameset*-Versionen von HTML und XHTML zur Verfügung. In XHTML 1.1. existiert es nicht.

6.4 Eingebettete Medien

Da HTML ein reines Textformat ist, können Nicht-Text-Daten nicht unmittelbar Teil des Dokuments sein. Stattdessen muss ein **Verweis** auf diese, dann extern liegende Datei vorgenommen werden, die den Browser veranlasst, die Datei an der gewünschten Stelle in das Dokumentlayout aufzunehmen. Medieneinbindungen sind also Platzhalter.

Das wichtigste dieser Elemente ist der leere ``-Tag, der eine Grafik einfügt – meist ist diese vom Typ JPG, GIF oder PNG. Die **einzubindende Datei** wird über das `src`-Attribut des Elements bestimmt, das obligatorische `alt`-Attribut stellt einen **Alternativtext** zur Verfügung, wenn die Grafik nicht angezeigt werden kann. Über die Attribute `height` und `width` werden die physischen Abmessungen der Grafik in Pixel angegeben.



Abbildung 6.9 Einfügen einer Grafik

Das Bild erscheint als Inlinenelement im Zeilenkontext, bestimmt daher also die Zeilenhöhe. Mittels des Float-Properties kann es von längeren Textpassagen umflossen werden. Da das Bild nicht mehr im Zeilenkontext steht, beeinflusst es die Zeilenhöhe nicht länger.

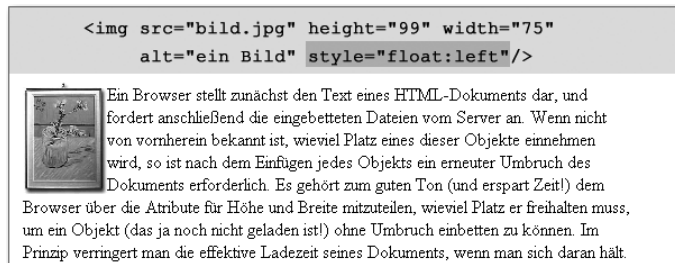


Abbildung 6.10 Ein gefloatetes Bild in einem Text.

Element	Rolle	Erläuterung
	Bindet eine Grafikdatei ein	Abschnitt 14.42
Attribute (Auswahl)	alt, border, height, longdesc, name, src, width	

Tabelle 6.9 Das -Element zum Einbetten von Grafiken

Als weitere Elemente zum Einbetten von Medien dienen die Container <object> und <iframe>, wobei <object> beliebige Medienobjekte, also Audio, Video und weitere Datenformate (auch Text oder Grafiken) einbetten kann. Über <param />-Elemente können Werte, wie Abspielparameter an ein Medienobjekt weitergegeben werden.

Sonderfälle

Element	Rolle	Erläuterung
<object> ... </object>	Fügt ein Objekt in die Seite ein	Abschnitt 14.59
Attribute (Auswahl)	archive, classid, codebase, data, height, width	
<param />	Übergibt Parameter an den umgebenden <object>-Container	Abschnitt 14.64
Attribute (Auswahl)	name, value	

Tabelle 6.10 Das <object>-Element zum Einbetten beliebiger Medien

Inlineframes dienen zum Einfügen von HTML-Dokumenten in eine Webseite. Dies geschieht durch Einbetten in einen Framerahmen, dessen Position durch den <iframe>-Container bestimmt wird. Die Verwendung von IFrames gilt aus Gründen der **Barrierefreiheit** als problematisch.

Element	Rolle	Erläuterung
<code><iframe> ... </iframe></code>	Definiert einen Inline Frame	Abschnitt 14.41
Attribute (Auswahl)	frameborder, height, name, scrolling, src, width	

Tabelle 6.11 Das `<iframe>`-Element zum Einbetten von HTML-Dokumenten

6.4.1 Welche Stylingmöglichkeiten bietet CSS für eingebettete Medien?

Generell können Sie die **Box-** und **Floatproperties** einsetzen, wobei Rahmen und Außenabstände hier am wichtigsten sind. Hintergrundfarben können bei teiltransparenten Grafiken von Interesse sein, oder zum Styling des Fallbackinhalts eines `<object>`-Containers. Normalerweise wird eine Farbdefinition jedoch durch das Medienobjekt verdeckt werden.

6.4.2 Welche HTML-Attribute werden für eingebettete Medien benötigt?

Neben den spezifischen Attributen, die das Medium einbetten (wie das `src`-Attribut bei ``) sollten Sie die **physischen Abmessungen** der Objekte per HTML-Attribut angeben (`width` und `height`), da diese Informationen unabhängig von CSS vorliegen sollten, um den Renderingvorgang zu beschleunigen.

Des Weiteren wichtig und **teilweise vorgeschrieben** sind Attribute, die die Barrierefreiheit der Inhalte betreffen – hier ist in erster Linie das `alt`-Attribut von `` zu nennen, das einen Alternativtext bei fehlender oder deaktivierter Grafik angibt. Das `border`-Attribut für `` benötigen Sie eigentlich nicht, es kann jedoch von Interesse sein, den Rahmen über `border="0"` für solche Grafiken zu deaktivieren, die sich innerhalb von Hyperlink-Containern befinden. Zwar zählt `border` zu den Präsentationsanweisungen, die auch per CSS verfügbar sind, steht so aber auch ohne Stylesheet zur Verfügung.

6.5 Listen

Listen stellen in HTML eine eigene Gattung von Blockelementen dar, die, Textabsätzen gleichgestellt, innerhalb des Dokumentinhalts auftreten. In diesem Falle dienen sie dem Erzeugen von Aufzählungen, deren Listenelemente `` wahlweise nummeriert (innerhalb von »geordneten Listen«, ``) oder mit Bullets versehen werden (innerhalb von »ungeordneten Listen«, ``).

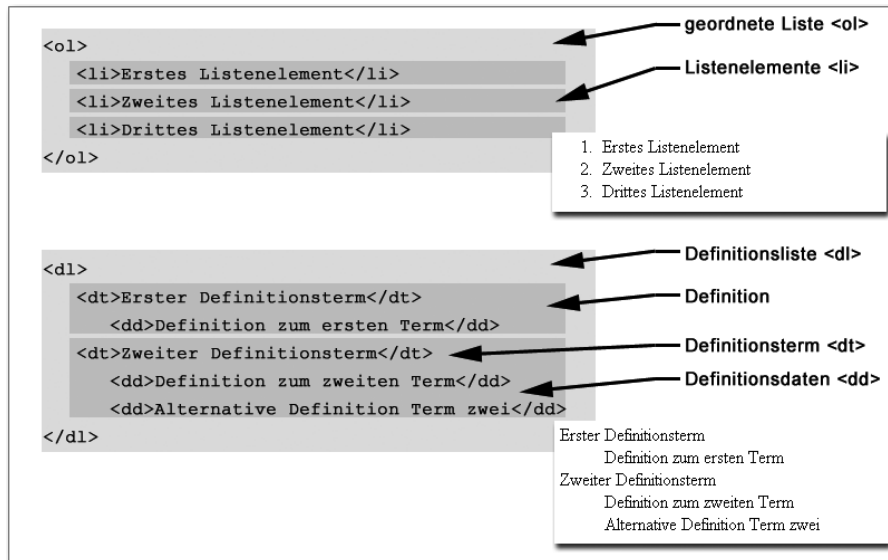


Abbildung 6.11 Geordnete Liste, Definitionsliste

Als dritter, noch gebräuchlicher Listentyp existiert die so genannte Definitionsliste `<dl>`, die allerdings keine Listenelemente vom Typ ``, sondern abwechselnd (aber ohne festgeschriebene Anzahl) definierte Begriffe (»definition term«, `<dt>`) und deren Definition (»definition data«, `<dd>`) enthält.

Definitionslisten

Für **Definitionslisten** existieren daher drei Elemente. Insgesamt wird dieser Listentyp eher selten verwendet, bietet jedoch interessante Einsatzmöglichkeiten, beispielsweise in Zusammenhang mit dem Gestalten von Formularen.

Element	Rolle	Erläuterung
<code><dl></code> ... <code></dl></code>	"definition list"; Container für eine Definitionsliste; enthält <code><dt></code> und <code><dl></code> -Container	Abschnitt 14.27
<code><dt></code> ... <code></dt></code>	"definition term"; bezeichnet den zu definierenden Ausdruck einer Definition	Abschnitt 14.28
<code><dd></code> ... <code></dd></code>	"definition data"; bezeichnet die zum vorstehenden <code><dt></code> -Container gehörende Definition	Abschnitt 14.22

Tabelle 6.12 Elemente für Definitionslisten

Geordnete und ungeordnete Listen

Die meistverbreiteten Listen sind die geordnete und die ungeordnete Liste, die sich nur durch das umgebende Containerelement unterscheiden. **Geordnete Listen** werden benötigt, wo neben einer Hierarchie auch eine **Nummerierung** der Listenpunkte gefordert ist. **Ungeordnete Listen** zeigen keine Nummerierung, sondern stellen die Listenelemente mit **Bullets** dar.

Element	Rolle	Erläuterung
<code></code> ... <code></code>	"unordered list"; Container für eine ungeordnete Liste; Listenelemente <code></code> werden durch Bullets gekennzeichnet	Abschnitt 14.90
<code></code> ... <code></code>	"ordered list"; Container für eine geordnete Liste; Listenelemente <code></code> erhalten Nummerierung mit Ziffern oder Buchstaben	Abschnitt 14.69
<code></code> ... <code></code>	"list item"; Listenelement einer ungeordneten oder geordneten Liste; Darstellung abhängig von der Art des umgebenden Listencontainers	Abschnitt 14.49

Tabelle 6.13 Elemente für geordnete und ungeordnete Listen

Hierarchien werden bei beiden Listentypen durch **Verschachtelung** erzeugt, wobei die Typen auch gemischt werden dürfen. Achten Sie bei der Verschachtelung darauf, dass die jeweils innenliegende Liste komplett in einem ``-Container der jeweils äußeren Liste enthalten sein muss (siehe Abbildung 6.12).

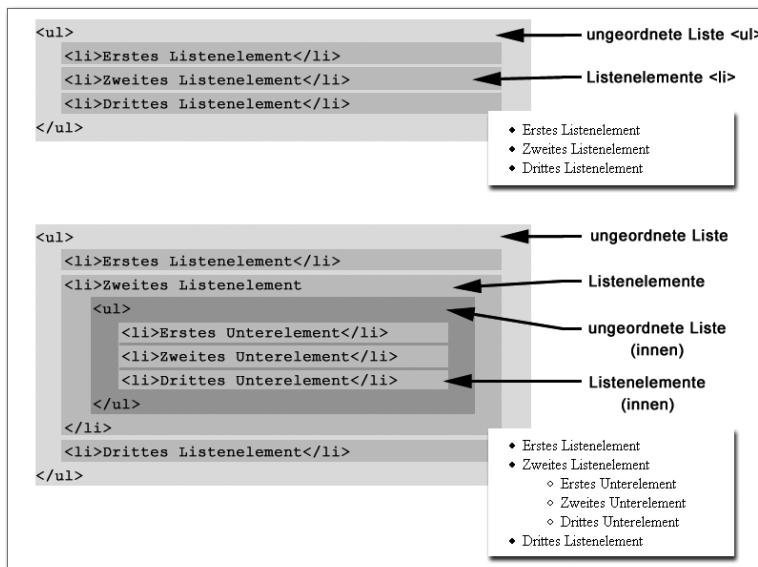


Abbildung 6.12 Ungeordnete Listen; einfach und verschachtelt

6.5.1 Welche Stylingmöglichkeiten bietet CSS für Listen?

CSS bietet speziell für Listen das Sammelproperty `list-style` und dazugehörige Unterproperties, um das Erscheinungsbild von ``- und ``-Listen anzupassen. Sie können die Aufzählungselemente deaktivieren (`list-style-type: none`) oder beliebig andere, bis hin zu grafischen Elementen, definieren.

Ansonsten gelten für Listen alle **Boxproperties**, mit denen Sie Abmessungen, Innen- und Außenabstände, Rahmen und Hintergrundfarben definieren können. In Zusammenhang mit Navigationslisten (mehr dazu im folgenden Kapitel) spielen auch das **Display-Property** und **Float-Eigenschaften** eine Rolle.

6.5.2 Welche HTML-Attribute werden für Listen benötigt?

Die kurze Antwort: keine. Es ist zwar für ``, `` und `` ein Attribut `type` definiert, um die Art des Aufzählungselements festzulegen, jedoch ist das CSS-Property `list-style-type` diesem überlegen. Allenfalls mag das Attribut `start` für `` erwägenswert sein, falls eine Nummerierung nicht bei 1 sondern mit einer anderen Ziffer beginnen soll.

6.6 Tabellen

Tabellen wurden in der Vergangenheit häufig nicht im Sinne ihrer eigentlichen Bestimmung eingesetzt, sondern, aufgrund der seinerzeit dürftigen Layout-Fähigkeiten von HTML fast immer zum Seitenlayout. Seit reine CSS-Layouts durch breite Browserunterstützung möglich sind, ist die Tabelle von Layoutaufgaben entbunden und kann wieder zu dem Zweck eingesetzt werden, für den sie eigentlich konzipiert wurde: zur Tabellierung von Daten (Abbildung 6.13).

Abgasgrenzwerte für Pkw						
Treibstoff	Schadstoff	EWG Stufe 1 (seit 1992)	EG Stufe 2 (seit 1996)	Euro 3 (seit 2000)	Euro 4 (seit 2005)	Euro 5 (Vorschlag)
Benzin	Kohlenmonoxid (CO)	2 720	2 200	2 300	1 000	1 000
	Kohlenwasserstoffe (HC)	-	-	200	100	75
	Stickoxid (NO _x)	-	-	150	80	60
	HC + NO _x	970	500	-	-	-
Diesel	Kohlenmonoxid (CO)	2 720	1 000	640	500	500
	Stickoxid (NO _x)	-	-	500	250	200
	HC + NO _x	-	700/900 DI	560	300	250
	Partikel	140	80/100 DI	50	25	5

Abbildung 6.13 Eine komplexere, mit CSS gestylte Datentabelle

Der Grundaufbau einer Tabelle ist verhältnismäßig einfach – man benötigt lediglich drei Arten von Elementen: den **Tabellencontainer** `<table>`, die **Zeilencontainer** `<tr>` und die **Datenzellencontainer** `<td>`. Sofern man Zellen in Tabellenkopffunktion benötigt, kommt ein viertes Element `<th>` hinzu, das ebenfalls eine Datenzelle darstellt, jedoch als **Headerzelle**....

Element	Rolle	Erläuterung
<code><table></code> ... <code></table></code>	Definiert eine Tabelle	Abschnitt 14.79
Attribute (Auswahl)	cellpadding, cellspacing, frame, rules, summary	
<code><tr></code> ... <code></tr></code>	Definiert eine Tabellenzeile als Unterelement von <code><table></code>	Abschnitt 14.87
Attribute (Auswahl)	align, valign	
<code><th> ... </th></code>	Definiert eine Tabellenkopfzelle als Unterelement von <code><tr></code>	Abschnitt 14.84
Attribute (Auswahl)	abbr, align, axis, colspan, height, rowspan, scope, valign	
<code><td> ... </td></code>	Definiert eine Tabellenzelle als Unterelement von <code><tr></code>	Abschnitt 14.81
Attribute (Auswahl)	abbr, align, axis, colspan, headers, rowspan, scope, valign	

Tabelle 6.14 Grundelemente für Tabellen

Sowohl Header- als auch Datenzellen stehen in Zeilencontainern `<tr>`, wobei die Headerzellen automatisch den Datenzellen ihrer Spalte als **Kopfzellen** zugeordnet werden. Textinhalte sind nur innerhalb von Header- und Datenzellencontainern gestattet.

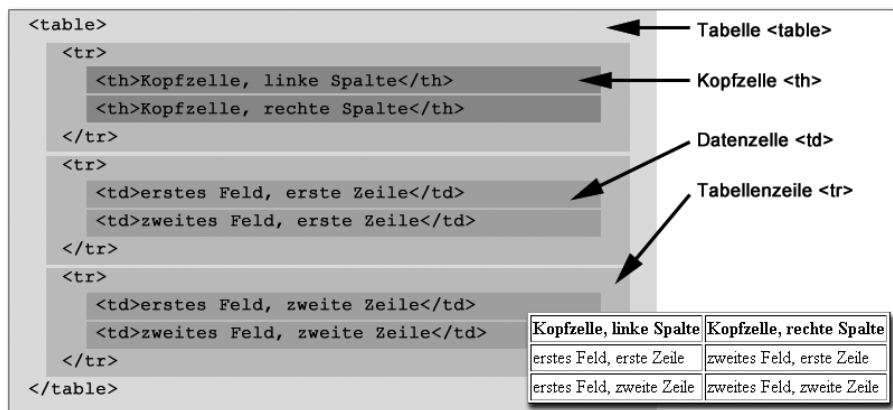


Abbildung 6.14 Grundaufbau einer Tabelle

Eine Tabelle kann jedoch noch weitaus komplexer werden. Da sich die Spalten nur indirekt aus der Zahl der Datenzellen pro Zeile ergeben, wird das Rendering vielspaltiger Tabellen erschwert, da der Browser die reale Spaltenbreite erst ermisst, wenn der jeweils größte Inhalt einer Spalte geladen ist.

Um dies zu erleichtern, existiert die Möglichkeit, zu Beginn der Tabelle die **Spalteneigenschaften** (hauptsächlich die Breite) zu definieren. Hierzu dienen die Elemente `<colgroup>` und `<col>`, wobei letztere die Eigenschaften von Einzelspalten, erstere Gruppen gleichförmiger Spalten festlegen. Ebenfalls zu Beginn der Tabelle, in diesem Fall stets als erstes Kindelement, kann eine **Tabellenüberschrift** `<caption>` stehen, die zentriert über der Tabelle erscheint.

Element	Rolle	Erläuterung
<code><caption> ... </caption></code>	Erläuternde Überschrift über (Default) bzw. unter der Tabelle; erstes Unterelement von <code><table></code>	Abschnitt 14.16
<code><colgroup> </colgroup></code>	Gruppieret mit dem <code><col></code> -Tag vorgenommene Spaltendefinitionen	Abschnitt 14.21
<code><col /></code>	Definiert die Eigenschaften einer Tabellenspalte als Teil einer Spaltengruppe (<code><colgroup></code>)	Abschnitt 14.20

Tabelle 6.15 Tabellenelemente für Überschrift und Spaltendefinitionen

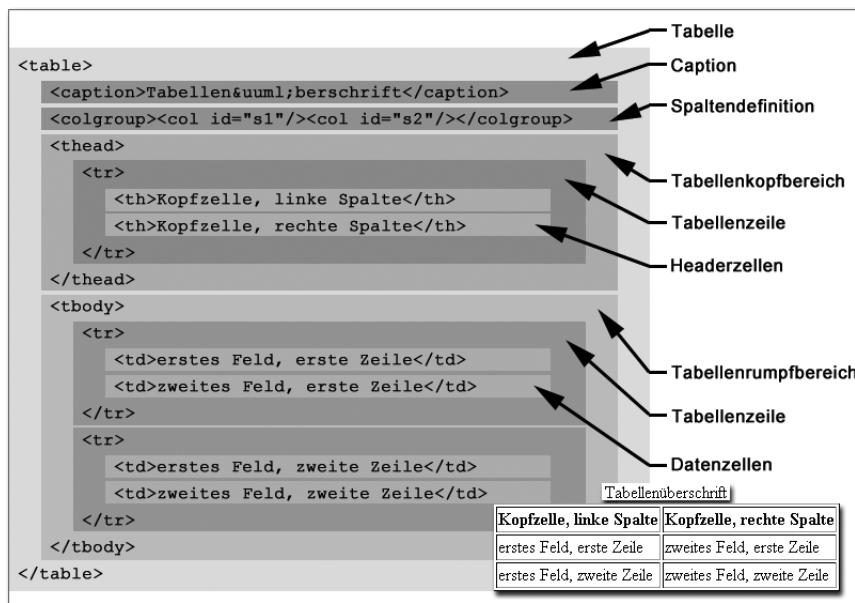


Abbildung 6.15 Aufbau einer komplexeren Tabelle

Weiterhin kann die Tabelle in Bereiche unterteilt werden. Man unterscheidet drei Arten von Bereichen: den **Tabellenkopfbereich** `<thead>`, der stets am Anfang der Tabelle steht (jedoch nach `<caption>` und Spaltendefinitionen), den **Tabellenrumpfbereich** `<tbody>`, der Zeilen `<tr>` mit Datentabellen enthält und den **Tabellenfußbereich** `<tfoot>`, der ebenfalls wieder Zeilen mit Datentabellen enthält. Der Rumpfbereich ist wiederholbar, wodurch man mehrere `<tbody>`-Zeilengruppen erzeugen kann, die bei Bedarf unterschiedlich gestylt werden können (so geschehen für Abbildung 6.13).

Element	Rolle	Erläuterung
<code><thead></code> ... <code></thead></code>	Erklärt die Tabellenzeile(n), die der Container enthält, zum Tabellenkopf	Abschnitt 14.85
<code><tfoot></code> ... <code></tfoot></code>	Erklärt die Tabellenzeile(n), die der Container enthält, zum Tabellenfuß	Abschnitt 14.83
<code><tbody></code> ... <code></tbody></code>	Erklärt die Tabellenzeile(n), die der Container enthält, zum Tabellenkörper	Abschnitt 14.80

Tabelle 6.16 Elemente zur Zeilengruppierung in Tabellen

Nicht per CSS zu erreichen sind Verbindungen zwischen Tabellenfeldern, die entweder zeilen- oder spaltenübergreifend zu definieren sind (auch Kombinationen sind möglich). Hierzu wird in der Ausgangszelle das `colspan-` oder `rowspan-` Attribut gesetzt, das die Zahl der überspannten Spalten oder Zeilen erhält (die Ausgangsspalte oder -zeile zählt mit).

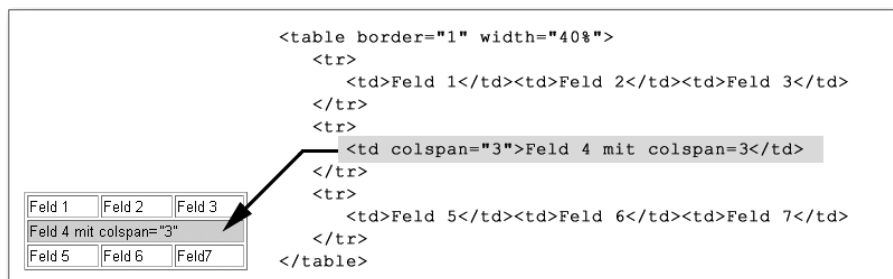


Abbildung 6.16 Horizontale Verbindung von Tabellenzellen mit `colspan`

Mit `colspan` und `rowspan` lassen sich beliebig komplexe Tabellengerüste aufbauen. Alternativ können Sie auch Tabellen verschachteln, indem Sie eine innere Tabelle in eine Datenzeile einfügen. Dies war gängige Praxis für Layouttabellen,

dürfte für reine Datentabellen jedoch selten erforderlich sein. Beachten Sie, dass verschachtelte Tabellen den Renderingvorgang bremsen.

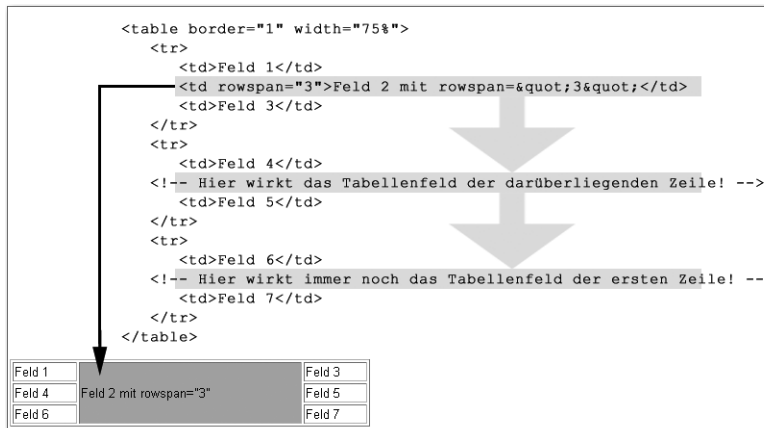


Abbildung 6.17 Vertikale Verbindung von Tabellenzellen mit `rowspan`

6.6.1 Welche Stylingmöglichkeiten bietet CSS für Tabellen?

Grundsätzlich sind die **Box-** und **Float-Properties** einsetzbar, mit denen Sie auch die Abmessungen der Tabelle festlegen sollten. **Hintergrundfarben** können Sie auf Basis der Tabelle, der Zellen, Zeilen, aber auch der Tabellenbereiche `<thead>` und `<tbody>` definieren.

Wenn Sie **Rahmen** auf Zellen oder Zeilenbasis einsetzen (dies funktioniert nicht für Zeilengruppen), führt dies zu einer möglicherweise **ungewollten Verdopplung der Rahmenbreite** bei benachbarten Zellen. Dem können Sie mit dem CSS-Property `border-collapse: collapse` abhelfen, dass Sie auf `<table>` anwenden.

Problematisch ist die Anwendung von Ausrichtungsproperties auf **Spaltendefinitionen**. Verwenden Sie stattdessen Klassen für die Datenzellen selbst oder arbeiten Sie auf Zeilenbasis.

6.6.2 Welche HTML-Attribute können für Tabellen verwendet werden?

Müssen Tabellen ohne CSS halbwegs präsentabel wirken, so wird oft ein sichtbares Tabellengitter benötigt. Das `border`-Attribut von `<table>` ist hierfür etwas grobschlächtig. Feinere Möglichkeiten bieten die Attribute `frame` und `rules`, die **Außen- und Innenlinien** getrennt definieren. Am `cellspacing`-Attribut kommen Sie derzeit noch nicht vorbei, da das korrespondierende CSS-Property `border-spacing` noch mangelhaft unterstützt wird. Das Attribut `cellpadding` kann

jedoch durch CSS ersetzt werden (sofern Sie nicht ohne CSS auskommen müssen). Für komplexe Tabellengitter können Sie `colspan` und `rowspan` ausgehend von Header- und Datenzellen einsetzen.

Das **Ausrichten von Inhalten** in der horizontalen ist mit dem CSS-Property `text-align` leicht möglich. Für die vertikale Ausrichtung ist es durchaus akzeptabel, das `valign`-Attribut weiterhin einzusetzen.

Für **barrierefreie Tabellen** sollten Sie über das `summary`-Attribut von `<table>` eine inhaltliche Beschreibung der Tabelle geben. Auch die Attribute `axis` und `scope` der Daten- und Headerzellen fallen in diese Kategorie.

6.7 Formulare

Formulare sind überall dort gefragt, wo Informationen vom Nutzer eingegeben werden sollen, oder gar Dateien von dessen lokalen Rechner auf den Webserver gelangen sollen. Formulare sind, wie Textabsätze, Listen und Tabellen ebenfalls Blockelemente, nehmen also den zur Verfügung stehenden Raum stets in voller Breite ein. Der **Grundaufbau** ist verhältnismäßig einfach – für den Anfang reichen zwei Elemente, wenn das Formular barrierefrei werden soll, kommt noch ein drittes hinzu. Sie benötigen den `<form>`-Container selbst, der alle Bestandteile des Formulars beinhaltet und das `<input>`-Element, ein leeres Element, dessen Erscheinungsbild vom Wert seines `type`-Attributs abhängt.

Input-Typ	Wirkung	Input-Typ	Wirkung
text	Texteingabefeld	password	Passwortfeld
radio	Radiobutton	checkbox	Checkboxfeld
hidden	unsichtbares Feld	file	Dateiuploadfeld
image	Grafik als Submit	button	neutraler Button
submit	Submit-Button	reset	Reset-Button

Tabelle 6.17 Die zehn Erscheinungsbilder des `<input>`-Elements

Für den **Formular-Container** werden zwei bis drei Attribute benötigt – vorgeschrieben ist jedoch lediglich das `action`-Attribut. Letzteres nennt das **Ziel des Datenversands**, also die Anwendung, die Seite oder das Script, das die Daten auswerten soll. Zusätzlich kann mit dem `method`-Attribut die Art des Datenversands festgelegt werden. Geben Sie `method="get"` an, so werden die Formulardaten als Querystring an den URL des `action`-Attributs angehängt. Für `method="post"` erfolgt die **Datenübertragung** etwas weniger offen als Teil des HTTP-Headers.

Die **Eingabefelder** benötigen, ungeachtet ihres Typs ein `name`-Attribut, das für den Datenversand gebraucht wird (teilweise wird *parallel* ein `id`-Attribut eingesetzt – geben Sie beiden Attributen in diesem Fall den gleichen Wert). Bestimmte Eingabefelder arbeiten im Verbund, sogenannten Arrays. Dies betrifft **Radiobuttons** und **Checkboxes**. Die Elemente einer solchen Gruppe benötigen, um zusammenzuarbeiten, denselben Wert des `name`-Attributs.

Besteht keine Eingabemöglichkeit in das Feld, müssen die zu übermittelnden Daten anders bestimmt werden – hierzu dient das `value`-Attribut von `<input>`, das bei für Submit- und Reset-Buttons auch die **Buttonaufschrift** bestimmt. (Da die Werte der Submit-Buttons meist nicht verschickt werden müssen, kann für diese das `name`-Attribut entfallen.) Bei **Eingabearrays** wird nur der `value` des tatsächlich angewählten Elements übertragen.

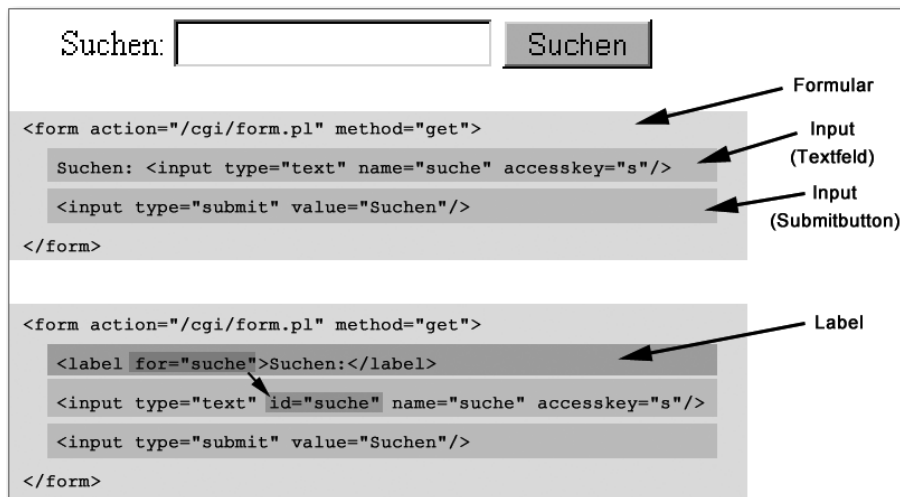


Abbildung 6.18 Einfaches Formular, unten mit hinzugefügtem `<label>`

Ein Formular gewinnt an **Barrierefreiheit**, wenn die Feldbeschriftungen über `<label>`-Container mit »ihren« Eingabefeldern logisch verknüpft werden. Dies geschieht über eine ID-IDREF-Beziehung (siehe Abbildung 6.18, unten). Eine weitere Verbesserung der Barrierefreiheit erhalten Sie durch Beifügung von `accesskey`-Attributen an `<input>`-Elemente, die so über Tastenkürzel ansprechbar werden.

Element	Rolle	Erläuterung
<code><form></code> ... <code></form></code>	Bildet einen Container für Formularelemente; bestimmt über Attribute das Formularverhalten	Abschnitt 14.33
Attribute (Auswahl)	<code>action</code> , <code>enctype</code> , <code>method</code>	
<code><input /></code>	Erzeugt verschiedene Arten von Inputelementen, deren Form über das <code>type</code> -Attribut bestimmt wird	Abschnitt 14.43
Attribute (Auswahl)	<code>accesskey</code> , <code>checked</code> , <code>disabled</code> , <code>maxlength</code> , <code>name</code> , <code>size</code> , <code>tabindex</code> , <code>type</code> , <code>value</code>	
<code><label> ...</code> <code></label></code>	Erzeugt zu einem Formularelement über eine IDREF-ID-Beziehung eine klickbare Beschriftung	Abschnitt 14.47
Attribute	<code>accesskey</code> , <code>for</code>	

Tabelle 6.18 Grundelemente des HTML-Formulars

Spätestens wenn Sie per `<input type="file"/>` einen **Dateiupload** ermöglichen wollen, müssen Sie ein **drittes Attribut** für `<form>` einfügen – in diesem Falle müssen Sie *zwingend* `enctype="multipart/form-data"` angeben, sonst können keine Dateien übertragen werden. Auch *muss* die POST-Methode zum Datenversand gewählt sein (Abbildung 6.19).

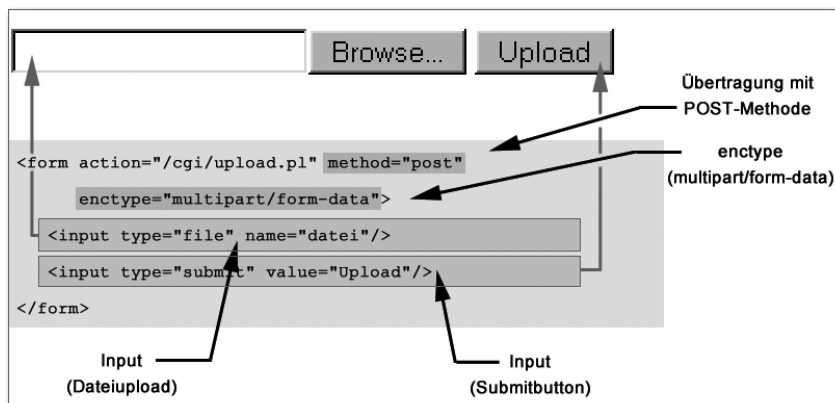


Abbildung 6.19 Formular zum Dateiupload

Neben den einfachen einzelnen Texteingabefeldern `<input type="text"/>` existiert auch ein Container-Element `<textarea>`, das für größere Textmengen konzipiert ist und mehrzeilig ausgelegt werden kann. Sofern Sie Schaltbuttons mit gestalteter Oberfläche benötigen (der Fläche können auch Grafiken hinzugefügt werden), können Sie auf den `<button>`-Container zurückgreifen.

Element	Rolle	Erläuterung
<code><button> ... </button></code>	Definiert einen Formularbutton; Containerinhalt wird auf Klickfläche dargestellt	Abschnitt 14.15
Attribute (Auswahl)	accesskey, name, tabindex, type, value	
<code><textarea> ... </textarea></code>	Erzeugt ein Texteingabefeld; Attribute legen Abmessungen fest	Abschnitt 14.82
Attribute (Auswahl)	accesskey, cols, name, rows, tabindex	

Tabelle 6.19 Formularelemente zur Generierung gestalteter Buttons und Texteingabeflächen

Neben den einfachen Typen von Inputfeldern stehen auch komplexere Eingabefelder zur Verfügung. Das `<select>`-Feld enthält eine Reihe von Optionen, die einzeln, oder (sofern bei `<select>` das Attribut `multiple` gesetzt ist) auch mehrere gewählt werden können. Da auch das `<select>`-Element zu den Eingabefeldern zählt, erhält es wieder ein `name`-Attribut (Abbildung 6.20).

Element	Rolle	Erläuterung
<code><select> ... </select></code>	Selectbox; dient als Container für Optionselemente	
Attribute (Auswahl)	multiple, name, size, tabindex	
<code><option> ... </option></code>	Erzeugt als Kindelement des Select-Containers dessen einzelne Wahloptionen	Abschnitt 14.62
Attribute (Auswahl)	disabled, selected, value	
<code><optgroup> ... </optgroup></code>	Definiert eine zusammengehörende Gruppe von Optionselementen einer Selectbox	Abschnitt 14.61
Attribute	disabled, label	

Tabelle 6.20 Elemente für Auswahlfelder und Combo-Boxen

Einzelne Optionen können mit dem `selected`-Attribut vorselektiert (ohne `multiple` wird jedoch nur eine davon berücksichtigt) oder mit `disabled` stillgelegt werden. Ist kein `value`-Attribut gesetzt, so wird der Inhalt des `<option>`-Containers als Daten verschickt.

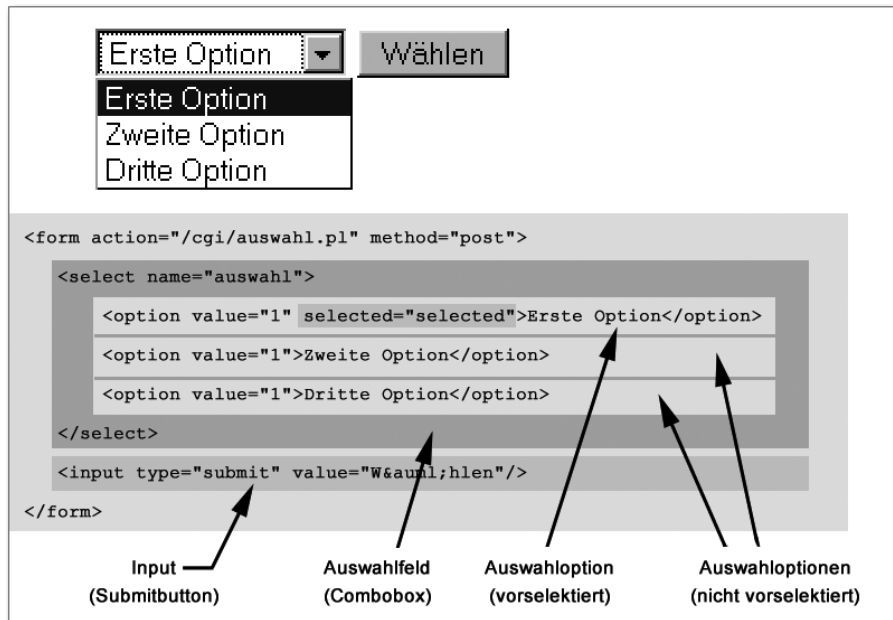


Abbildung 6.20 Formular mit Auswahlfeld <select>

Das Element <fieldset> dient zum **Gruppieren von Formularelementen**, wobei in diesem Fall eine logische Gruppierung und keine funktionale (wie bei der Bildung eines Buttonarrays) gemeint ist. Defaultmäßig werden Feldgruppen mit umgebenden Rahmen versehen. Mit <legend> kann einer Gruppe eine **Überschrift** zugewiesen werden. Auf die Funktionalität des Formulars haben Feldgruppen jedoch keinen Einfluss.

Element	Rolle	Erläuterung
<fieldset> ... </fieldset>	Definiert eine zusammengehörende Gruppe von Inputelementen eines Formulars	Abschnitt 14.31
<legend> ... </legend>	Definiert eine Überschrift einer Fieldset-Gruppe	Abschnitt 14.48

Tabelle 6.21 Elemente zum Gruppieren von Formularfeldern

Problematisch bei der **optischen Präsentation** von Formularen ist, dass es sich bei allen Formulareingabefeldern um **Inlinenelemente** handelt. Sie können sich damit behelfen, eine **Tabelle** zum Ausrichten der Felder und Beschriftungen einzusetzen, was eine sichere und bewährte (wenn auch inzwischen etwas verpönte) Methode darstellt. Sie können auch mit dem **Zeilenumbruchbefehl**
 arbeiten, um eine Zeilenstruktur zu erzielen.



Abbildung 6.21 Formular mit Feldgruppen und -überschriften

Eine gewisse Gestaltungsbasis stellt auch das Blockelement `<fieldset>` dar. Sie können per CSS eine Gesamtbreite für das Formular vergeben und die Feldgruppenbereiche gezielt stylen.

```

<form action="/cgi/user.pl" method="get">
<fieldset><legend>Userdaten</legend>
  <label for="vn">Vorname:
  <input type="text" id="vn" name="Vorname" accesskey="s"/>
  </label>
  <label for="nn">Nachname:
  <input type="text" id="nn" name="Nachname" accesskey="s"/>
  </label>
  <label for="un">Username:
  <input type="text" id="un" name="Username" accesskey="s"/>
  </label>
  <label for="pw">Passwort:
  <input type="password" id="pw" name="Passwort" accesskey="s"/>
  </label>
</fieldset>
<fieldset class="linear"><legend>Vorlieben</legend>
  <label>Pizza:

```

```


</label>
<label>Pasta:
<input type="radio" name="Vorlieben" value="1"/>
</label>
<label>Vegan:
<input type="radio" name="Vorlieben" value="1"/>
</label>
</fieldset>
<fieldset class="linear"><legend>Und ab damit!</legend>


</fieldset>
</form>

```

Listing 6.2 Quelltext zu Abbildung 6.21

Für das Formular in Abbildung 6.22 wurden dem `<label>`-Container Block-eigenschaften zugewiesen (`display:block`), was die Felder in einzelne Zeilen gruppiert. Innerhalb der beiden unteren Feldgruppen wurde dies durch einen höherrangigen CSS-Selektor widerrufen, um die Radiobuttons nebeneinander zu platzieren.

Abbildung 6.22 Umsetzung mit CSS (kapitel_06/formulare.html)

Folgendes CSS wurde in diesem Fall eingesetzt:

```

form      { width:400px;
            font-family:arial,Helvetica,sans-serif;
            font-size:0.8em;}
fieldset  { border:1px dotted #000;
            padding:0.5em 0; margin:0.5em 0;
            background-color:#ddd;}
label     { display:block;
            margin-left:1em;
            text-align:right; }

```

```

legend    { margin-left:0.5em;
            font-weight:bold;
            background-color:#fff;
            border:1px solid #000;}
input     { margin:0 1em 0 0.5em; }

/* innerhalb dieser Gruppen bleibt <label> inline: */

fieldset.linear {text-align:right; background-color:#ccc;}
fieldset.linear label {display:inline;}

```

Listing 6.3 CSS zu kapitel_06/formulare.html

6.8 Zusammenfassung

Was Sie aus diesem Kapitel mitnehmen sollten:

- ▶ Die wichtigsten Elemente zur Strukturierung von Texten sind Überschriften und Textabsatzcontainer. Überschriften sollten in hierarchischer Reihenfolge verwendet werden.
- ▶ Für die Gestaltung innerhalb von Fließtexten existieren Inlinenelemente, die ihre Inhalte nach logischen und physischen Gesichtspunkten auszeichnen
- ▶ Auf rein präsentativen Markup sollten Sie zugunsten von CSS verzichten.
- ▶ Hyperlinks sind ebenfalls Inlinenelemente. Sie besitzen eine Defaultpräsentation, können jedoch mit CSS beliebig gestaltet werden.
- ▶ Hyperlinks in Fließtext sollten stets auf den ersten Blick als Hyperlinks erkennbar bleiben.
- ▶ Mittels Verweisen können Bilder und andere Medien in HTML-Dokumente eingebettet werden.
- ▶ Listen existieren in drei Formen, von denen die ungeordnete und die geordnete Liste die meist verwendeten darstellen.
- ▶ Tabellen sollten ausschließlich zur Präsentation von tabellierten Daten eingesetzt werden.
- ▶ Formulare dienen zum Übertragen von Daten vom Client zum Server. Sie können mit CSS beliebig präsentiert werden.

Wie es weitergeht:

Im folgenden Kapitel wird die Gestaltung von Navigationen auf der Basis ungeordneter Listen erläutert. Vertieft werden darüber hinaus einige Aspekte des Seitenlayouts mit CSS.

Index

A

- Accessibility → Barrierefreiheit
Ankercontainer → Hyperlinkcontainer
Ankerpunkt 143
Application Level Eventhandler 920
Attribut 38
 Attributname 39
 Attributwert 39
 Attributwert, Begrenzer 39, 51
 leeres 51
 leeres, Begrenzer 51
Attribute
 abbr, <td> 853
 abbr, <th> 876
 accept, <form> 582
 accept, <input> 657
 accept-charset, <form> 582
 accesskey, <a> 412
 accesskey, <area> 447
 accesskey, <button> 499
 accesskey, <input> 657
 accesskey, <label> 682
 accesskey, <legend> 686
 accesskey, <textarea> 864
 action, <form> 583
 action, <isindex> (IE) 676
 action, <isindex> (NN) 677
 align, <applet> 440
 align, <caption> 505
 align, <col> 516
 align, <colgroup> 522
 align, <div> 541
 align, <embed> 561
 align, <frame> (IE) 597
 align, <h1> – <h6> 617
 align, <hr> 622
 align, <iframe> 637
 *align, * 645
 align, <input> 658
 align, <legend> 686
 align, <object> 749
 align, <p> 773
 align, <table> 832
 align, <tbody> 847
 align, <td> 853
 align, <tfoot> 872
 align, <th> 877
 align, <thead> 888
 align, <tr> 895
 alink, <body> 482
 allowtransparency, <frame> (IE) 597
 alt, <applet> 441
 alt, <area> 448
 alt, <embed> 562
 *alt, * 646
 alt, <input> 659
 application, <frame> (IE) 598
 application, <iframe> (IE) 640
 archive, <applet> 441
 archive, <object> 750
 autocomplete, <form> (IE) 586
 autocomplete, <input> (IE) 664
 autostart, <embed> 562
 axis, <td> 854
 axis, <th> 877
 background, <body> 483
 background, <table> (IE) 840
 background, <table> (NN) 842
 background, <td> (IE) 860
 background, <td> (NN) 862
 background, <th> (IE) 884
 background, <th> (NN) 886
 background, <tr> (IE) 898
 behavior, <marquee> (IE) 709
 bgcolor, <body> 483
 bgcolor, <col> (IE) 519
 bgcolor, <colgroup> (IE) 525
 bgcolor, <marquee> (IE) 709
 bgcolor, <table> 833
 bgcolor, <tbody> (IE) 848
 bgcolor, <td> 854
 bgcolor, <tfoot> (IE) 874
 bgcolor, <th> 878
 bgcolor, <thead> (IE) 890
 bgcolor, <tr> 896
 bgproperties, <body> (IE) 485
 border, <embed> 562
 border, <frameset> (NN) 609
 border, <iframe> (IE) 640
 *border, * 647
 border, <object> 750
 border, <table> 833
 bordercolor, <frame> (IE) 599

Index

- bordercolor*, <frame> (NN) 600
- bordercolor*, <frameset> (IE) 607
- bordercolor*, <frameset> (NN) 610
- bordercolor*, <iframe> (IE) 641
- bordercolor*, <table> (IE) 840
- bordercolor*, <table> (NN) 843
- bordercolor*, <td> (IE) 861
- bordercolor*, <th> (IE) 884
- bordercolor*, <tr> (IE) 898
- bordercolordark*, <table> (IE) 840
- bordercolordark*, <td> (IE) 861
- bordercolordark*, <th> (IE) 885
- bordercolordark*, <tr> (IE) 898
- bordercolorlight*, <table> (IE) 841
- bordercolorlight*, <td> (IE) 861
- bordercolorlight*, <th> (IE) 885
- bordercolorlight*, <tr> (IE) 899
- bottommargin*, <body> (IE) 485
- cellpadding*, <table> 834
- cellspacing*, <table> 835
- char*, <col> 517
- char*, <colgroup> 523
- char*, <tbody> 847
- char*, <td> 855
- char*, <tfoot> 872
- char*, <th> 878
- char*, <thead> 889
- char*, <tr> 897
- charoff*, <col> 517
- charoff*, <tbody> 848
- charoff*, <td> 855
- charoff*, <tfoot> 873
- charoff*, <th> 878
- charoff*, <thead> 889
- charoff*, <tr> 897
- charset*, <a> 413
- charset*, <link> 695
- charset*, <script> 795
- checked*, <input> 659
- cite*, <blockquote> 478
- cite*, 531
- cite*, <ins> 672
- cite*, <q> 787
- class* (Core Collection) 916
- class*, 813
- classid*, <object> 750
- clear*,
 495
- clear*, <table> (IE) 841
- code*, <applet> 441
- code*, <embed> 562
- code*, <object> (IE) 754
- codebase*, <applet> 441
- codebase*, <embed> 563
- codebase*, <object> 750
- codetype*, <object> 751
- color*, <basefont> 464
- color*, 575
- color*, <hr> (IE) 624
- cols*, <div> (NN) 542
- cols*, <frameset> 605
- cols*, <pre> (NN) 784
- cols*, <table> (NN) 843
- cols*, <textarea> 865
- colspan*, <td> 855
- colspan*, <th> 879
- compact*, <dir> 537
- compact*, <dl> 550
- compact*, <menu> 716
- compact*, 761
- compact*, 907
- content*, <meta> 720
- controls*, (IE) 650
- coord*, <a> 414
- cti*, <a> (DoCoMo) 422
- data*, <object> 751
- datetime*, 531
- datetime*, <ins> 673
- declare*, <object> 751
- défer*, <script> 795
- dir* (I18N Collection) 916
- dir*, <bdo> 468
- direction*, <marquee> (IE) 710
- disabled*, <a> (IE) 421
- disabled*, <button> 499
- disabled*, <input> 659
- disabled*, <link> (IE) 700
- disabled*, <optgroup> 766
- disabled*, <option> 769
- disabled*, <select> 805
- disabled*, <style> (IE) 822
- disabled*, <textarea> 865
- dynsrc*, (IE) 651
- dynsrc*, <input> (IE) 664
- effect*, (WebTV) 577
- email*, <a> (DoCoMo) 423
- enctype*, <form> 583
- event*, <script> (IE) 797
- EventHandler* 915
- face*, <basefont> 465
- face*, 575
- fontsize*, <body> (WebTV) 488
- font-weight*, (NN) 577

- for*, <label> 682
for, <script> (IE) 798
frame, <table> 835
frameborder, <embed> 563
frameborder, <frame> 593
frameborder, <frameset> (IE) 608
frameborder, <iframe> 638
framespacing, <frameset> (IE) 608
gutter, <div> (NN) 543
headers, <td> 856
height, <applet> 442
height, <embed> 563
height, <frame> (IE) 599
height, <iframe> 638
height, 647
height, <marquee> (IE) 710
height, <object> 752
height, <table> (IE) 841
height, <table> (NN) 843
height, <td> 857
height, <th> 879
hidden, <embed> 563
hidefocus, <a> (IE) 421
href, <a> 414
href, <area> 449
href, <base> 459
href, <link> 695
hreflang, <a> 415
hreflang, <link> 695
hspace, <applet> 442
hspace, <embed> 564
hspace, <iframe> (IE) 641
hspace, 647
hspace, <marquee> (IE) 710
hspace, <object> 752
hspace, <table> (NN) 844
http-equiv, <meta> 721
id (Core Collection) 917
id, <base> (IE) 460
id, <script> (IE) 797
id, 813
ijam, <a> (DoCoMo) 423
ilet, <a> (DoCoMo) 423
invertborder, <hr> (WebTV) 625
irst, <a> (DoCoMo) 424
ismap, 648
ismap, <input> 660
ista, <a> (DoCoMo) 424
istyle, <input> (DoCoMo) 666
istyle, <textarea> (DoCoMo) 868
iswf, <a> (DoCoMo) 424
kana, <a> (DoCoMo) 425
label, <optgroup> 767
label, <option> 769
lang (I18N Collection) 917
lang (Universalattribut) 61
language, <script> 796
leftmargin, <body> (IE) 485
link, <body> 484
longdesc, <frame> 594
longdesc, <iframe> 638
longdesc, 648
loop, <embed> 564
loop, (IE) 651
loop, <marquee> (IE) 710
loopdelay, (IE) 652
lowsrc, (IE) 652
lowsrc, (NN) 653
lowsrc, <input> (IE) 664
marginheight, <body> (NN) 487
marginheight, <frame> 594
marginheight, <iframe> 638
marginwidth, <body> (NN) 487
marginwidth, <frame> 594
marginwidth, <iframe> 638
maxlength, <input> 660
mayscript, <applet> (NN) 444
media, <link> 695
media, <style> 820
method, <form> 584
methods, <a> 415
multiple, <select> 805
n, <nextid> 739
name, <a> 415
name, <applet> 442
name, <button> 500
name, <embed> 564
name, <form> 584
name, <frame> 595
name, <iframe> 639
name, 649
name, <input> 660
name, <link> (IE) 701
name, <map> 707
name, <meta> 721
name, <object> 752
name, <param> 777
name, <select> 805
name, <textarea> 866
nohref, <area> 449
noresize, <frame> 595
noresize, <iframe> (IE) 641

Index

- nosave*, (NN) 653
noshade, <hr> 623
notab, <area> (IE) 452
notab, <input> (IE) 665
notab, <object> (IE) 755
nowrap, <body> (IE) 486
nowrap, <dd> (IE) 528
nowrap, <dt> (IE) 555
nowrap, <table> (IE) 841
nowrap, <td> 857
nowrap, <th> 880
object, <applet> 442
object, <object> (IE) 755
palette, <embed> 564
ping, <a> (WebApp) 426
playcount, <embed> 565
pluginspage, <embed> 565
pluginurl, <embed> 565
point-size, (NN) 576
profile, <head> 620
prompt, <isindex> 676
readonly, <input> 661
readonly, <textarea> 866
rel, <a> 417
rel, <link> 697
rev, <a> 417
rev, <link> 699
rightmargin, <body> (IE) 486
rows, <frameset> 607
rows, <textarea> 866
rowspan, <td> 858
rowspan, <th> 880
rules, <table> 837
runat, <script> (IE) 798
scheme, <meta> 722
scope, <td> 859
scope, <th> 881
scroll, <body> (IE) 486
scrollamount, <marquee> (IE) 711
scrolldelay, <marquee> (IE) 711
scrolling, <frame> 596
scrolling, <iframe> 639
selected, <option> 770
shape, <a> 418
shape, <area> 450
size, <basefont> 465
size, 575
size, <hr> 623
size, <input> 661
size, <select> 806
span, <col> 518
span, <colgroup> 523
src, <applet> (IE) 443
src, <embed> 565
src, <frame> 596
src, <iframe> 639
src, 649
src, <input> 662
src, <script> 796
Standardattribute 915
standby, <object> 753
start, (IE) 652
start, 761
style (Style Collection) 918
style, 814
summary, <table> 839
suppress, <a> (NN) 422
suppress, (NN) 654
tabindex, <a> 418
tabindex, <area> 450
tabindex, <button> 500
tabindex, <input> 662
tabindex, <object> 753
tabindex, <select> 806
tabindex, <textarea> 867
target, <a> 419
target, <area> 451
target, <base> 459
target, <form> 585
target, <isindex> (NN) 677
target, <link> 700
telbook, <a> (DoCoMo) 425
text, <body> 484
title (Core Collection) 918
title, <a> 420
title, <head> (IE) 620
title, <html> (IE) 629
title, 649
topmargin, <body> (IE) 487
transparency, (WebTV) 578
truespeed, <marquee> (IE) 711
type, <a> 420
type, <button> 501
type, <dir> (IE) 537
type, <dir> (NN) 538
type, <embed> 566
type, <input> 662
type, 690
type, <link> 700
type, <menu> (IE) 717
type, <menu> (NN) 718
type, <object> 753

type, 762
type, <param> 777
type, <script> 796
type, <style> 822
type, 907
units, <embed> 566
urn, <a> 420
urn, <link> 700
usemap, 649
usemap, <input> 663
usemap, <object> 754
utn, <a> (DoCoMo) 425
utn, <form> (DoCoMo) 586
valign, <caption> (IE) 505
valign, <col> 518
valign, <colgroup> 524
valign, <table> (IE) 842
valign, <tbody> 848
valign, <td> 859
valign, <tfoot> 873
valign, <th> 883
valign, <thead> 889
valign, <tr> 897
value, <button> 501
value, <input> 663
value, 691
value, <option> 770
value, <param> 777
valuetype, <param> 777
vcard_name, <input> (IE) 665
version, <html> 628
vlink, <body> 484
volume, <embed> 566
vspace, <applet> 443
vspace, <embed> 566
vspace, <iframe> (IE) 641
vspace, 650
vspace, <marquee> (IE) 712
vspace, <object> 754
vspace, <table> (NN) 844
width, <applet> 443
width, <col> 519
width, <colgroup> 525
width, <div> (NN) 543
width, <embed> 566
width, <frame> (IE) 600
width, <hr> 623
width, <iframe> 640
width, 650
width, <marquee> (IE) 712
width, <object> 754

width, <pre> 783
width, <table> 839
width, <td> 860
width, <th> 884
wrap, <pre> (NN) 784
wrap, <textarea> (NN) 868
wrap, <textarea> (IE) 867
xml:lang (I18N Collection) 919
xml:lang (Universalattribut) 61
xml:space, <pre> 783
xmlns, <html> 628

Attributsammlungen

Common 915

Core 915

I18N 915

Style 915

Außenabstand → CSS-Boxproperties, margin

Auszeichnungsbefehl → Marke

Auszeichnungskommando → Marke

B

Barrierefreie Informationstechnik-
Verordnung (BITV) 241

Barrierefreiheit 239

Accessibility Tools 282

Accessibility-Attribute (IE) 947

Ausgabegerät 242

Barrierearmut 241

betreffene Gruppen 240, 241

Bilder 264

Bildschirmlupe 243

BITV 241

BITV-Selbstbewertung 245

Color Contrast Analyzer 248, 282

Datentabellen 251

Delorie Lynx Viewer 291

Downloadlink 268

Dreamweaver-Plug-in 244

Eventhandler 920

Eventhandler, geräteunabhängige 267

Events, korrespondierende 920

Farbkontraste 248

Formulare 257

Formulare, CSS-Gestaltung 262

Formulare, Eingabefelder 259

Formulare, Grundstruktur 258

Kurzcheck Bilder und Grafiken 265

Kurzcheck Datentabellen 253

Kurzcheck Download-Dateien 268

Index

- Kurzcheck dynamische Inhalte* 268
 - Kurzcheck Farben* 248
 - Kurzcheck Frames* 247
 - Kurzcheck Imagemaps* 266
 - Kurzcheck Layout* 246
 - Kurzcheck Layouttabellen* 247
 - Kurzcheck Links im Textinhalt* 256
 - Kurzcheck Listen* 250
 - Kurzcheck Markup von Inhalten* 249
 - Kurzcheck Multimedia* 266
 - Kurzcheck Navigation* 255
 - Kurzcheck Popup-Fenster* 267
 - Kurzcheck Schriften* 248
 - Kurzcheck Skripte* 267
 - Kurzcheck Seitenaufbau* 246
 - Kurzcheck Sprache* 250
 - Kurzcheck Verständlichkeit der Inhalte* 269
 - Kurzcheck Weiterleitungen* 267
 - Multimedia* 264
 - Navigation* 253
 - Navigation, Taststurbedienung* 254
 - Nutzersicht* 242
 - Regelungen, gesetzliche* 241
 - Screenreader* 243
 - Sitemap* 256
 - Technologien* 243
 - Textäquivalente* 264
 - Vorteile* 240
 - Wave Toolbar* 281
 - Web Accessibility Initiative (WAI)* 241
 - Web Accessibility Toolbar* 248, 280, 291
 - Web Content Accessibility Guidelines (WCAG)* 241
 - Yellowpipe Lynx Viewer* 290
 - Basisschriftgröße 133
 - Basistextrichtung 916
 - Beispiel
 - HTML 4.01 frameset* 80
 - HTML 4.01 strict* 80
 - HTML 4.01 transitional* 79
 - XHTML 1.0 frameset* 82
 - XHTML 1.0 strict* 81
 - XHTML 1.0 transitional* 81
 - XHTML 1.1* 82
 - XHTML 1.1 mit Schema-Locator* 83
 - Bereichsnavigation 168
 - Bezeichner
 - lokaler* 52
 - qualifizierter* 52
 - Bildschirmlupe 243
 - Braille-Display 243
 - Breadcrumb-Navigation 169
 - Browser
 - Browserarchiv* 1168
 - Browserarchiv, www.evolt.org* 299
 - Camino* 1167
 - Cello* 299
 - Chimera* 299
 - Evolt.org* 1168
 - Firefox* 288, 348, 359, 1167
 - Internet Explorer* 1167
 - Lynx* 289, 1168
 - Mosaic* 299
 - Mozilla* 1167
 - Safari* 348
 - SeaMonkey* 1167
 - Textbrowser-Simulationen* 288
 - Viola* 299
 - X-Smiles* 359
- ## C
-
- Cascading-Prinzip 103, 999
 - Präferenz in der Präferenzkaskade* 999
 - Spezifität der Selektoren* 1000
 - CDATA-Textinhalt 42
 - Character-Referenz 40
 - Checkboxen 157
 - closing delimiter 36
 - CMYK → Farben, Vierfarbmodell
 - CSS At-Rules 1003
 - @charset* 1003
 - @font-face* 1004
 - @import* 1004
 - @media* 1005
 - @page* 1006
 - CSS für mobile Geräte 228
 - Abbildungen* 230
 - Ankercontainer* 230
 - Dokumentbereiche* 229
 - Formulare* 232
 - Listen* 232
 - Margins* 229
 - media=* 228
 - Menüs* 230
 - Navigation* 230
 - Schriftfamilie* 231
 - Skiplinks* 230
 - Tabellen* 231
 - Text* 230
 - Wrapper-Container* 229

- CSS Mobile Profile 228
- CSS-Boxmodell 105
 - Außenabstand* 105
 - Box-Properties* 105
 - Innenabstand* 105
 - Innenbreite* 105
 - Innenhöhe* 105
 - Rahmeneigenschaften* 105
- CSS-Boxproperties
 - margin* 1055
- CSS-Formatierungsregeln 91, 991
 - !important-Schalter* 993
 - Reihenfolge* 94, 993
- CSS-Kommentarsyntax 102, 1002
- CSS-Maßeinheiten 1023
 - em* 1023
 - ex* 1026
 - Längenangaben* 1023
 - Längenangaben, absolut* 1023
 - Längenangaben, relativ* 1023
 - Prozentangaben* 1026
- CSS-Properties 1028
 - background* 1044
 - background-attachment* 1045
 - background-color* 1046
 - background-image* 1047
 - background-position* 1048
 - background-position, Längenangaben* 1048
 - background-position, Prozentangaben* 1048
 - background-position, Schlüsselwörter* 1049
 - background-repeat* 1045, 1050
 - border* 1056
 - border-collapse* 1103
 - border-color* 1057
 - border-spacing* 1104
 - border-style* 1058
 - border-width* 1061
 - bottom* 1072
 - Boxproperties* 1053
 - Boxproperties, Anwendbarkeit* 1053
 - Boxproperties, Containerinhalt* 1053
 - Boxproperties, Inhaltsbereich* 1054
 - browserspezifische* 1121
 - caption-side* 1106
 - clear* 1074
 - clip* 1075
 - color* 1052
 - content* 1114
 - Contentgenerierung* 1114
 - counter-increment* 1115
 - counter-reset* 1116
 - cursor* 1111
 - direction* 1028
 - display* 1090
 - Display-Property* 1090
 - empty-cells* 1108
 - Farben* 106
 - Farben und Hintergründe* 1044
 - float* 1076
 - font* 1038
 - font-family* 1038
 - Fontproperties* 1038
 - font-size* 1039
 - font-size-adjust* 1040
 - font-stretch* 1041
 - font-style* 1041
 - font-variant* 1042
 - font-weight* 1043
 - font-weight, Fettegrad* 1043
 - height* 1063
 - Hintergrundeigenschaften* 106
 - Kontur-Properties* 1067
 - left* 1079
 - letter-spacing* 1029
 - line-height* 1029
 - Listenstyle-Properties* 1098
 - list-style* 1098
 - list-style-image* 1099
 - list-style-position* 1100
 - list-style-type* 1100
 - margin* 1055
 - max-height* 1065
 - max-width* 1066
 - min-height* 1067
 - min-width* 1067
 - orphans* 1120
 - outline* 1067
 - outline-color* 1068
 - outline-style* 1069
 - outline-width* 1071
 - overflow* 1080
 - padding* 1054
 - page* 1117
 - page-break-after* 1118
 - page-break-before* 1119
 - page-break-inside* 1119
 - position* 1082
 - Positioning-Properties* 1072
 - quotes* 1116

Index

- right* 1083
 - scrollbar-3dlight-color* 1122
 - scrollbar-arrow-color* 1122
 - scrollbar-base-color* 1123
 - scrollbar-darkshadow-color* 1124
 - scrollbar-face-color* 1124
 - scrollbar-highlight-color* 1125
 - Scrollbar-Properties* 1121
 - scrollbar-shadow-color* 1126
 - scrollbar-track-color* 1126
 - Seitenformatierung* 1117
 - Sonstige Properties* 1111
 - Tabellen-Properties* 1103
 - table-layout* 1109
 - text-align* 1030
 - text-decoration* 1030
 - text-indent* 1031
 - text-indent, hängender Einzug* 1032
 - Textproperties* 1028
 - text-shadow* 1033
 - text-transform* 1033
 - top* 1084
 - unicode-bidi* 1034
 - vertical-align* 1035
 - visibility* 1086
 - white-space* 1036
 - widows* 1120
 - width* 1064
 - word-spacing* 1037
 - z-index* 1089
 - CSS-Pseudoelemente 1014
 - :after* 1015
 - :before* 1015
 - :first-letter* 1016
 - :first-line* 1016
 - CSS-Pseudoklassen 1017
 - :active* 1018
 - :focus* 1018
 - :hover* 1019
 - :lang* 1020
 - :link* 1020
 - :visited* 1021
 - CSS-Pseudo-Kontextselektoren 1021
 - :first-child* 1021
 - CSS-Regel
 - Basis-URI* 1047
 - CSS-Selektoren 92
 - Adjacent-Sibling-Selektor* 1012
 - attributabhängige* 1013
 - Child-Selektor* 1011
 - Descendant Selektor* 1011
 - Gruppenselektor* 1008
 - Gruppierung* 93
 - ID-Selektor* 97, 1010
 - Klassenselektor* 1008
 - Klassenselektor, frei* 95
 - Klassenselektor, freier* 1009
 - Klassenselektor, gebunden* 96
 - Klassenselektor, gebundener* 1009
 - Klassenselektor, Mehrfachzuweisung* 1009
 - Klassenselektor, Wahl des Bezeichners* 1010
 - Typselektor* 93, 1007
 - Universeller Selektor* 1007
 - Wahl des Selektors* 101
 - CSS-Styledefinition
 - Dokumentstyle* 88, 996
 - Inlinestyle* 90, 995
 - Position* 85, 995
 - Stylesheet, importiert* 88, 997
 - Stylesheet, verlinkt* 87, 998
 - CSS-Werte 1023
 - Farben* 1027
 - Farbnamen* 1027
 - Hexadezimalfarbangabe* 1027
 - Hexadezimalfarbangabe, gekürzt* 1027
 - RGB-Wert, numerisch* 1027
 - RGB-Wert, prozentual* 1028
- ## D
-
- Databinding-Attribute (IE) 949
 - Default-Namensraum 53
 - Definition
 - Attribute, Common Collection* 915
 - Barrierefreiheit* 61
 - Doctype-Switching* 1127
 - Floaten* 116
 - Hyperreferenz* 87
 - Markup* 22
 - MIME-Typ* 58
 - Typgültigkeit* 60
 - valid* 60
 - Validator* 60
 - Validierung* 60
 - W3C Candidate Recommendation* 1138
 - W3C Proposed Edited Recommendation* 1138
 - W3C Proposed Recommendation* 1138
 - W3C Recommendation* 1138
 - W3C Working Draft* 1138

- W3C Working Group Note* 1138
 - well-formed* 60
 - Wohlförmtheit* 60
 - Definitionsliste 149
 - Design für mobile Geräte 222
 - Accesskeys* 227
 - Bilder* 225
 - Farben* 226
 - Farbtiefe* 226
 - Grafiken* 225
 - Kontrast* 226
 - Linearisierung* 222
 - Markup-Sprache, Wahl* 219
 - Navigation* 227
 - Quelltextaufbau* 224
 - Quelltextreihenfolge* 223
 - Schriftgrößen* 224
 - Tabellen* 225
 - Test mit Openwave Phone Simulator* 236
 - Test mit Opera* 233
 - Test mit Opera, Normalansicht* 233
 - Test mit Opera, SSR* 235
 - Textauszeichnung* 225
 - Textfarbe* 227
 - Doctype-Deklaration 76
 - Aufbau* 77
 - Name* 77
 - PUBLIC* 77
 - Public-Identifizier* 77
 - Public-Identifizier, Aufbau* 78
 - SYSTEM* 77
 - System-Identifizier* 77
 - Dokumentinhalt
 - Gestaltung* 131
 - Dokumentkopf 23
 - Dokumentrumpf 23
 - Dokumenttyp-Definition 43
 - Elementvereinbarung* 43
 - Dokumenttyp-Deklaration 53
 - DTD-Variante 54
- E**
-
- Editoren 1173
 - Adobe Dreamweaver* 1175
 - Adobe GoLive CS2* 1175
 - Arachnophilia* 272, 1173
 - Bluefish* 206, 272, 1174
 - Crimson Editor* 272, 1174
 - Dreamweaver* 274
 - Edit4Win* 272, 1173
 - GoLive* 274
 - Homesite* 272
 - HTML-Kit* 206, 210
 - jEdit* 271
 - Namo Web Editor* 274
 - Notepad* 271
 - Notepad++* 272, 1173
 - Nvu* 274, 275, 1175
 - Pagespinner* 272, 1174
 - Phase 5* 272, 273
 - Phase 5 HTML-Editor* 1175
 - Quanta* 206, 272
 - Quanta+* 1174
 - SciTE* 271
 - Screem* 272, 1174
 - SuperHTML* 274
 - tsWebEditor* 206
 - Ultra Edit* 271, 1173
 - Weaverslave* 272, 1173
 - XFormation* 361
 - Element 22, 36
 - Abschluss* 49
 - Container* 38
 - Inhaltsmodell* 37
 - leeres* 38
 - Verkürzung* 48
 - Verschachtelung* 37
 - Elemente
 - <a>* 142, 411
 - <a>, accesskey* 145, 412
 - <a>, charset* 413
 - <a>, coord* 414
 - <a>, cti (DoCoMo)* 422
 - <a>, disabled (IE)* 421
 - <a>, email (DoCoMo)* 423
 - <a>, hidefocus (IE)* 421
 - <a>, href* 145, 414
 - <a>, hreflang* 415
 - <a>, ijam (DoCoMo)* 423
 - <a>, ilet (DoCoMo)* 423
 - <a>, irst (DoCoMo)* 424
 - <a>, ista (DoCoMo)* 424
 - <a>, iswf (DoCoMo)* 424
 - <a>, kana (DoCoMo)* 425
 - <a>, methods* 415
 - <a>, name* 143, 415
 - <a>, ping (WebApp)* 426
 - <a>, rel* 417
 - <a>, rev* 417
 - <a>, shape* 418
 - <a>, suppress (NN)* 422

Index

- `<a>`, *tabindex* 145, 418
- `<a>`, *target* 145, 419
- `<a>`, *telbook (DoCoMo)* 425
- `<a>`, *title* 145, 420
- `<a>`, *type* 420
- `<a>`, *urn* 420
- `<a>`, *utn (DoCoMo)* 425
- `<abbr>` 139, 430
- `<abbr>`, *title* 139
- `<acronym>` 139, 433
- `<acronym>`, *title* 139
- `<address>` 437
- `<applet>` 439
- `<applet>`, *align* 440
- `<applet>`, *alt* 441
- `<applet>`, *archive* 441
- `<applet>`, *code* 441
- `<applet>`, *codebase* 441
- `<applet>`, *height* 442
- `<applet>`, *hspace* 442
- `<applet>`, *mayscript (NN)* 444
- `<applet>`, *name* 442
- `<applet>`, *object* 442
- `<applet>`, *src (IE)* 443
- `<applet>`, *vspace* 443
- `<applet>`, *width* 443
- `<area>` 446
- `<area>`, *accesskey* 447
- `<area>`, *alt* 448
- `<area>`, *href* 449
- `<area>`, *nohref* 449
- `<area>`, *notab (IE)* 452
- `<area>`, *shape* 450
- `<area>`, *tabindex* 450
- `<area>`, *target* 451
- `` 139, 140, 141, 454
- `<base>` 69, 458
- `<base>`, *href* 459
- `<base>`, *id (IE)* 460
- `<base>`, *target* 459
- `<basefont>` 463
- `<basefont>`, *color* 464
- `<basefont>`, *face* 465
- `<basefont>`, *size* 465
- `<bdo>` 466
- `<bdo>`, *dir* 468
- `<big>` 141, 473
- `<blockquote>` 72, 476
- `<blockquote>`, *cite* 478
- `<body>` 60, 481
- `<body>`, *alink* 482
- `<body>`, *background* 483
- `<body>`, *bgcolor* 483
- `<body>`, *bgproperties (IE)* 485
- `<body>`, *bottommargin (IE)* 485
- `<body>`, *fontsize (WebTV)* 488
- `<body>`, *Inhaltsmodell* 69
- `<body>`, *leftmargin (IE)* 485
- `<body>`, *link* 484
- `<body>`, *marginheight (NN)* 487
- `<body>`, *marginwidth (NN)* 487
- `<body>`, *nowrap (IE)* 486
- `<body>`, *rightmargin (IE)* 486
- `<body>`, *scroll (IE)* 486
- `<body>`, *text* 484
- `<body>`, *topmargin (IE)* 487
- `<body>`, *vlink* 484
- `
` 160, 493
- `
`, *clear* 495
- `<button>` 158, 498
- `<button>`, *accesskey* 499
- `<button>`, *disabled* 499
- `<button>`, *name* 500
- `<button>`, *tabindex* 500
- `<button>`, *type* 501
- `<button>`, *value* 501
- `<caption>` 153, 503
- `<caption>`, *align* 505
- `<caption>`, *valign (IE)* 505
- `<center>` 507
- `<cite>` 138, 510
- `<code>` 138, 512
- `<col>` 153, 515
- `<col>`, *align* 516
- `<col>`, *bgcolor (IE)* 519
- `<col>`, *char* 517
- `<col>`, *charoff* 517
- `<col>`, *span* 518
- `<col>`, *valign* 518
- `<col>`, *width* 519
- `<colgroup>` 153, 521
- `<colgroup>`, *align* 522
- `<colgroup>`, *bgcolor (IE)* 525
- `<colgroup>`, *charoff* 523
- `<colgroup>`, *span* 524
- `<colgroup>`, *valign* 524
- `<colgroup>`, *width* 525
- `<dd>` 149, 527
- `<dd>`, *nowrap (IE)* 528
- `` 74, 140, 530
- ``, *cite* 531
- ``, *datetime* 531

- <dfn> 138, 533
- <dir> 536
- <dir>, compact 537
- <dir>, type (IE) 537
- <dir>, type (NN) 538
- <div> 70, 539
- <div>, align 541
- <div>, cols (NN) 542
- <div>, Div-Layout 71
- <div>, gutter (NN) 543
- <div>, width (NN) 543
- <dl> 149, 549
- <dl>, compact 550
- <dt> 149, 554
- <dt>, nowrap (IE) 555
- 138, 557
- <embed> (Prop.) 560
- <embed>, align 561
- <embed>, alt 562
- <embed>, autostart 562
- <embed>, border 562
- <embed>, code 562
- <embed>, codebase 563
- <embed>, frameborder 563
- <embed>, height 563
- <embed>, hidden 563
- <embed>, hspace 564
- <embed>, loop 564
- <embed>, name 564
- <embed>, palette 564
- <embed>, playcount 565
- <embed>, pluginspage 565
- <embed>, pluginurl 565
- <embed>, src 565
- <embed>, type 566
- <embed>, units 566
- <embed>, volume 566
- <embed>, vspace 566
- <embed>, width 566
- <fieldset> 160, 569
- 142, 573
- , color 575
- , effect (WebTV) 577
- , face 575
- , font-weight (NN) 577
- , point-size (NN) 576
- , size 575
- , transparency (WebTV) 578
- <form> 73, 156, 580
- <form>, accept 582
- <form>, accept-charset 582
- <form>, action 156, 583
- <form>, autocomplete (IE) 586
- <form>, enctype 158, 583
- <form>, method 156, 584
- <form>, name 584
- <form>, target 585
- <form>, utn (DoCoMo) 586
- <frame> 63, 592
- <frame>, align (IE) 597
- <frame>, allowtransparency (IE) 597
- <frame>, application (IE) 598
- <frame>, bordercolor (IE) 599
- <frame>, bordercolor (NN) 600
- <frame>, frameborder 593
- <frame>, height (IE) 599
- <frame>, longdesc 594
- <frame>, marginheight 594
- <frame>, marginwidth 594
- <frame>, name 595
- <frame>, noresize 595
- <frame>, scrolling 596
- <frame>, src 596
- <frame>, width (IE) 600
- <frameset> 63, 604
- <frameset>, border (NN) 609
- <frameset>, bordercolor (IE) 607
- <frameset>, bordercolor (NN) 610
- <frameset>, cols 605
- <frameset>, frameborder (IE) 608
- <frameset>, framespacing (IE) 608
- <frameset>, rows 607
- <h1> 71
- <h1> - <h6> 615
- <h1> - <h6>, align 617
- <h1>-<h6> 134
- <head> 60, 618
- <head>, profile 620
- <head>, title (IE) 620
- <hr> 72, 621
- <hr>, align 622
- <hr>, color (IE) 624
- <hr>, invertborder (WebTV) 625
- <hr>, noshade 72, 623
- <hr>, size 623
- <hr>, width 623
- <html> 60, 626
- <html>, title (IE) 629
- <html>, version 628
- <html>, xmlns 628
- <i> 140, 141, 633
- <iframe> 147, 635

Index

- `<iframe>`, *align* 637
- `<iframe>`, *application* (IE) 640
- `<iframe>`, *border* (IE) 640
- `<iframe>`, *bordercolor* (IE) 641
- `<iframe>`, *frameborder* 638
- `<iframe>`, *height* 638
- `<iframe>`, *hspace* (IE) 641
- `<iframe>`, *longdesc* 638
- `<iframe>`, *marginheight* 638
- `<iframe>`, *marginwidth* 638
- `<iframe>`, *name* 639
- `<iframe>`, *noresize* (IE) 641
- `<iframe>`, *scrolling* 639
- `<iframe>`, *src* 639
- `<iframe>`, *vspace* (IE) 641
- `<iframe>`, *width* 640
- `` 146, 643
- ``, *align* 645
- ``, *alt* 646
- ``, *border* 148, 647
- ``, *controls* (IE) 650
- ``, *dynsrc* (IE) 651
- ``, *height* 146, 647
- ``, *hspace* 647
- ``, *ismap* 648
- ``, *longdesc* 648
- ``, *loop* (IE) 651
- ``, *loopdelay* (IE) 652
- ``, *lowsrc* (IE) 652
- ``, *lowsrc* (NN) 653
- ``, *name* 649
- ``, *nosave* (NN) 653
- ``, *src* 146, 649
- ``, *start* (IE) 652
- ``, *suppress* (NN) 654
- ``, *title* 649
- ``, *usemap* 649
- ``, *vspace* 650
- ``, *width* 146, 650
- `<input>` 156, 655
- `<input>`, *accept* 657
- `<input>`, *accesskey* 157, 657
- `<input>`, *align* 658
- `<input>`, *alt* 659
- `<input>`, *autocomplete* (IE) 664
- `<input>`, *checked* 659
- `<input>`, *disabled* 659
- `<input>`, *dynsrc* (IE) 664
- `<input>`, *id* 157
- `<input>`, *ismap* 660
- `<input>`, *istyle* (DoCoMo) 666
- `<input>`, *lowsrc* (IE) 664
- `<input>`, *maxlength* 660
- `<input>`, *name* 157, 660
- `<input>`, *notab* (IE) 665
- `<input>`, *readonly* 661
- `<input>`, *size* 661
- `<input>`, *src* 662
- `<input>`, *tabindex* 662
- `<input>`, *type* 156, 662
- `<input>`, *usemap* 663
- `<input>`, *value* 157, 663
- `<input>`, *vcard_name* (IE) 665
- `<ins>` 74, 140, 671
- `<ins>`, *cite* 672
- `<ins>`, *datetime* 673
- `<isindex>` 674
- `<isindex>`, *action* (IE) 676
- `<isindex>`, *action* (NN) 677
- `<isindex>`, *prompt* 676
- `<isindex>`, *target* (NN) 677
- `<kbd>` 138, 679
- `<label>` 157, 681
- `<label>`, *accesskey* 682
- `<label>`, *for* 682
- `<legend>` 685
- `<legend>`, *accesskey* 686
- `<legend>`, *align* 686
- `` 688
- ``, *type* 690
- ``, *value* 691
- `<link>` 67, 693
- `<link>`, *charset* 695
- `<link>`, *disabled* (IE) 700
- `<link>`, *href* 695
- `<link>`, *hreflang* 695
- `<link>`, *media* 695
- `<link>`, *name* (IE) 701
- `<link>`, *rel* 697
- `<link>`, *rev* 699
- `<link>`, *target* 700
- `<link>`, *type* 700
- `<link>`, *urn* 700
- `<listing>` 704
- `<map>` 706
- `<map>`, *name* 707
- `<marquee>` (IE) 708
- `<marquee>`, *behavior* (IE) 709
- `<marquee>`, *bgcolor* (IE) 709
- `<marquee>`, *direction* (IE) 710
- `<marquee>`, *height* (IE) 710
- `<marquee>`, *hspace* (IE) 710

- `<marquee>`, *loop* (IE) 710
- `<marquee>`, *scrollamount* (IE) 711
- `<marquee>`, *scrolldelay* (IE) 711
- `<marquee>`, *truespeed* (IE) 711
- `<marquee>`, *vspace* (IE) 712
- `<marquee>`, *width* (IE) 712
- `<menu>` 715
- `<menu>`, *compact* 716
- `<menu>`, *type* (IE) 717
- `<menu>`, *type* (NN) 718
- `<meta>` 66, 719
- `<meta>`, *content* 720
- `<meta>`, *http-equiv* 721
- `<meta>`, *name* 721
- `<meta>`, *scheme* 722
- `<nextid>` 738
- `<nextid>`, *n* 739
- `<noembed>` 568
- `<noframes>` 63, 740
- `<noscript>` 74, 744
- `<object>` 69, 147, 748
- `<object>`, *align* 749
- `<object>`, *archive* 750
- `<object>`, *border* 750
- `<object>`, *classid* 750
- `<object>`, *code* (IE) 754
- `<object>`, *codebase* 750
- `<object>`, *codetype* 751
- `<object>`, *data* 751
- `<object>`, *declare* 751
- `<object>`, *height* 752
- `<object>`, *hspace* 752
- `<object>`, *name* 752
- `<object>`, *notab* (IE) 755
- `<object>`, *object* (IE) 755
- `<object>`, *standby* 753
- `<object>`, *tabindex* 753
- `<object>`, *type* 753
- `<object>`, *usemap* 754
- `<object>`, *vspace* 754
- `<object>`, *width* 754
- `` 148, 150, 759
- ``, *compact* 761
- ``, *start* 151, 761
- ``, *type* 151, 762
- `<optgroup>` 159, 765
- `<optgroup>`, *disabled* 766
- `<optgroup>`, *label* 767
- `<option>` 159, 768
- `<option>`, *disabled* 159, 769
- `<option>`, *label* 769
- `<option>`, *selected* 770
- `<option>`, *value* 770
- `<p>` 71, 135, 772
- `<p>`, *align* 773
- `<p>`, *Defaultmargins* 135
- `<p>`, *Inhaltsmodell* 71
- `<param>` 69, 147, 775
- `<param>`, *name* 777
- `<param>`, *type* 777
- `<param>`, *value* 777
- `<param>`, *valuetype* 777
- `<plaintext>` 779
- `<pre>` 72, 781
- `<pre>`, *cols* (NN) 784
- `<pre>`, *width* 783
- `<pre>`, *wrap* (NN) 784
- `<pre>`, *xml:space* 783
- `<q>` 138, 785
- `<q>`, *cite* 787
- `<s>` 142, 789
- `<samp>` 138, 791
- `<script>` 65, 68, 794
- `<script>`, *charset* 795
- `<script>`, *defer* 795
- `<script>`, *event* (IE) 797
- `<script>`, *for* (IE) 798
- `<script>`, *id* (IE) 797
- `<script>`, *Inhaltsmodell* 68
- `<script>`, *language* 796
- `<script>`, *runat* (IE) 798
- `<script>`, *src* 796
- `<script>`, *type* 796
- `<select>` 159, 803
- `<select>`, *disabled* 805
- `<select>`, *multiple* 159, 805
- `<select>`, *name* 159, 805
- `<select>`, *size* 806
- `<select>`, *tabindex* 806
- `<small>` 141, 809
- `` 139, 812
- ``, *class* 813
- ``, *id* 813
- ``, *style* 814
- `<strike>` 142, 815
- `` 138, 817
- `<style>` 65, 67, 819
- `<style>`, *disabled* (IE) 822
- `<style>`, *Inhaltsmodell* 67
- `<style>`, *media* 820
- `<style>`, *type* 822
- `<sub>` 141, 826

Index

- `<sup>` 141, 828
- `<table>` 73, 152, 830
- `<table>`, *align* 832
- `<table>`, *background (IE)* 840
- `<table>`, *background (NN)* 842
- `<table>`, *bgcolor* 833
- `<table>`, *border* 833
- `<table>`, *bordercolor (IE)* 840
- `<table>`, *bordercolor (NN)* 843
- `<table>`, *bordercolordark (IE)* 840
- `<table>`, *bordercolorlight (IE)* 841
- `<table>`, *cellpadding* 834
- `<table>`, *cellspacing* 835
- `<table>`, *clear (IE)* 841
- `<table>`, *cols (NN)* 843
- `<table>`, *frame* 835
- `<table>`, *height (IE)* 841
- `<table>`, *height (NN)* 843
- `<table>`, *hspace (NN)* 844
- `<table>`, *nowrap (IE)* 841
- `<table>`, *rules* 837
- `<table>`, *summary* 839
- `<table>`, *valign (IE)* 842
- `<table>`, *vspace (NN)* 844
- `<table>`, *width* 839
- `<tbody>` 845
- `<tbody>`, *align* 847
- `<tbody>`, *bgcolor (IE)* 848
- `<tbody>`, *char* 847
- `<tbody>`, *charoff* 848
- `<tbody>`, *valign* 848
- `<td>` 152, 851
- `<td>`, *abbr* 853
- `<td>`, *align* 853
- `<td>`, *axis* 854
- `<td>`, *background (IE)* 860
- `<td>`, *background (NN)* 862
- `<td>`, *bgcolor* 854
- `<td>`, *bordercolor (IE)* 861
- `<td>`, *bordercolordark (IE)* 861
- `<td>`, *bordercolorlight (IE)* 861
- `<td>`, *char* 855
- `<td>`, *charoff* 855
- `<td>`, *colspan* 855
- `<td>`, *headers* 856
- `<td>`, *height* 857
- `<td>`, *nowrap* 857
- `<td>`, *rowspan* 858
- `<td>`, *valign* 859
- `<td>`, *width* 860
- `<textarea>` 158, 863
- `<textarea>`, *accesskey* 864
- `<textarea>`, *cols* 865
- `<textarea>`, *disabled* 865
- `<textarea>`, *istyle (DoCoMo)* 868
- `<textarea>`, *name* 866
- `<textarea>`, *readonly* 866
- `<textarea>`, *rows* 866
- `<textarea>`, *tabindex* 867
- `<textarea>`, *wrap (IE)* 867
- `<textarea>`, *wrap (NN)* 868
- `<tfoot>` 871
- `<tfoot>`, *align* 872
- `<tfoot>`, *bgcolor (IE)* 874
- `<tfoot>`, *char* 872
- `<tfoot>`, *charoff* 873
- `<tfoot>`, *valign* 873
- `<th>` 152, 875
- `<th>`, *abbr* 876
- `<th>`, *align* 877
- `<th>`, *axis* 877
- `<th>`, *background (IE)* 884
- `<th>`, *background (NN)* 886
- `<th>`, *bgcolor* 878
- `<th>`, *bordercolor (IE)* 884
- `<th>`, *bordercolordark (IE)* 885
- `<th>`, *bordercolorlight (IE)* 885
- `<th>`, *char* 878
- `<th>`, *charoff* 878
- `<th>`, *colspan* 879
- `<th>`, *height* 879
- `<th>`, *nowrap* 880
- `<th>`, *rowspan* 880
- `<th>`, *scope* 881
- `<th>`, *valign* 883
- `<th>`, *width* 884
- `<thead>` 887
- `<thead>`, *align* 888
- `<thead>`, *bgcolor (IE)* 890
- `<thead>`, *char* 889
- `<thead>`, *charoff* 889
- `<thead>`, *valign* 889
- `<title>` 60, 892
- `<tr>` 152, 894
- `<tr>`, *align* 895
- `<tr>`, *background (IE)* 898
- `<tr>`, *bgcolor* 896
- `<tr>`, *bordercolor (IE)* 898
- `<tr>`, *bordercolordark (IE)* 898
- `<tr>`, *bordercolorlight (IE)* 899
- `<tr>`, *char* 897
- `<tr>`, *charoff* 897

- `<tr>`, *valign* 897
- `<tt>` 900
- `<u>` 142, 903
- `` 74, 148, 150, 905
- ``, *compact* 907
- ``, *type* 151, 907
- `<var>` 138, 910
- `<xmp>` 912
- Strukturelemente 60
- Elementname 44
- Encoding
 - Escapesequenzen 971
 - Markierungszeichen 972
 - Trennzeichen 972
 - URL-Encoding 971
 - Zeichen, alphanumerische 972
 - Zeichen, nicht-druckbare 973
 - Zeichen, nicht-reservierte 972
 - Zeichen, reservierte 972
 - Zeichen, verbotene 973
- Endmarke 22, 36
- Endtag 22
- Endtag → Endmarke
- Entität 40
 - vordefinierte in SGML 40
- Entities
 - Akzentzeichen 956
 - Buchstabenformen, internationale 961
 - Character-Entities 968
 - Großbuchstaben mit Akzentzeichen 957
 - Großbuchstaben, griechische 959
 - Interpunktionszeichen 961
 - Kleinbuchstaben mit Akzentzeichen 958
 - Kleinbuchstaben, griechische 960
 - Ligaturen 959
 - Pfeile 965
 - Spationierung 966
 - Steuerzeichen 955
 - Symbole in Buchstabenform 965
 - Symbole, höhere mathematische 962
 - Symbole, kommerzielle 956
 - Symbole, mathematische 962
 - Symbole, Währungen 966
 - Symbole, weitere 965
 - Symbole, Zahlen 967
 - Whitespace 966
 - Zahlen 967
 - Zeichen, typographische 963
- Ersetzungsvorschrift → Entität
- Eventhandler 915
 - Applikationsebene 920
 - geräteabhängige 920
 - geräteunabhängige 920
 - Interaktionsebene 920
 - onabort (IE) 928
 - onactivate (IE) 929
 - onafterprint (IE) 929
 - onafterupdate (IE) 929
 - onbeforeactivate (IE) 930
 - onbeforecopy (IE) 930
 - onbeforecut (IE) 930
 - onbeforedeactivate (IE) 931
 - onbeforeeditfocus (IE) 931
 - onbeforepaste (IE) 932
 - onbeforeprint (IE) 932
 - onbeforeunload (IE) 932
 - onbeforeupdate (IE) 933
 - onblur (Intrinsic) 921
 - onbounce (IE) 933
 - oncellchange (IE) 934
 - onchange (Intrinsic) 921
 - onclick (Common) 922
 - oncontextmenu (IE) 934
 - oncopy (IE) 934
 - oncut (IE) 934
 - ondataavailable (IE) 935
 - ondatasetchanged (IE) 935
 - ondatasetcomplete (IE) 935
 - ondblclick (Common) 923
 - ondeactivate (IE) 936
 - ondrag (IE) 936
 - ondragend (IE) 936
 - ondragenter (IE) 936
 - ondragleave (IE) 937
 - ondragover (IE) 937
 - ondragstart (IE) 937
 - ondrop (IE) 937
 - onerror (IE) 938
 - onerrorupdate (IE) 938
 - onfilterchange (IE) 938
 - onfinish (IE) 938
 - onfocus (Intrinsic) 923
 - onfocusin (IE) 939
 - onfocusout (IE) 939
 - onhelp (IE) 939
 - onkeydown (Common) 923
 - onkeypress (Common) 924
 - onkeyup (Common) 924
 - onlayoutcomplete (IE) 940
 - onload (Intrinsic) 924
 - onlosecapture (IE) 940
 - onmousedown (Common) 925

Index

onmouseenter (IE) 940
onmouseleave (IE) 941
onmousemove (Common) 925
onmouseover(Common) 926
onmouseup (Common) 925, 926
onmousewheel (IE) 941
onmove (IE) 941
onmoveend (IE) 942
onmovestart (IE) 942
onpaste (IE) 942
onpropertychange (IE) 942
onreadystatechange (IE) 943
onreset (Intrinsic) 926
onresize (IE) 943
onresizeend (IE) 944
onresizestart (IE) 944
onrowenter (IE) 944
onrowexit (IE) 945
onrowsdelete (IE) 945
onrowsinserted (IE) 945
onscroll (IE) 946
onselect (Intrinsic) 927
onselectstart (IE) 946
onstart (IE) 946
onstop (IE) 947
onsubmit (Intrinsic) 927
onunload (Intrinsic) 927
Schreibweise 921

Events

äquivalente Events 920

F

Farben 1027
8-Bit Palette 981
browsersafe 981
CLUT-Palette 981
CSS3 Color Module 983
Differenz PC/Mac 982
Farbbezeichner 982
Farbbezeichner CSS3 988
Farbraum 982
Farbsynthese 981
Farbsynthese, additive 982
Farbsynthese, subtraktive 981
Farbwerte, neutrale 983
Farbzuweisungen in CSS 980
Farbzuweisungen in HTML 979
Grundfarben 983
in CSS 1027
nicht druckbare 982

RGB-Dezimalcode (CSS) 980
RGB-Farbmodell 982
RGB-Hexadezimalcode 982
RGB-Hexadezimalcode, verkürzt (CSS)
980
RGB-Prozentangabe (CSS) 980
Vierfarbmodell 981
Windows VGA Farbpalette 983
X11 Farbnamen 982

Fettegrad 1043

Firmen

Apple 304
Netscape 297
NTT DoCoMo 351
Opera 304
TTN 351, 354

Floatlayout 114

Foren

SelfHTML 1172
XHTML Forum 1172

Formatierer 23

Formulare 156

Buttonaufschrift 157

G

Grafiken

Einbindung 146

H

Hersteller

Nokia 1167

Hilfsnavigation 170

Hintergrundfarbe 133

Hintergrundgrafiken 133

Host Language → XHTML Host Language

HTML-Editoren

TopStyle Pro 206

HTML-Versionen

cHTML 31, 350

Doti 1.0 354

HTML 1.0 295

HTML 2.0 25, 295

HTML 3.0 299, 302

HTML 3.2 26, 297

HTML 4.0 26, 303

HTML 4.0 mobile 31, 355

HTML 4.01 26

HTML 4.01 frameset 307

HTML 4.01 strict 305

HTML 4.01 transitional 307
 HTML 5.0 304, 346
 HTML 5.0, geplante Erweiterungen 347
 HTML 5.0, Testimplementierungen 348
 HTML Level 0 293
 HTML+ (HTMLPlus) 299, 300
 iMode 1.0 351
 iMode 2.0 352
 iMode 3.0 353
 iMode 4.0 353
 iMode HTML 31
 iMode XHTML 31
 iMode XHTML 1.0 355
 iMode XHTML 1.1 356
 mobile Endgeräte 350
 Web Applications 1.0 346
 WML 2.0 357
 XHTML 1.0 28, 308
 XHTML 1.0 frameset 310
 XHTML 1.0 strict 309
 XHTML 1.0 transitional 309
 XHTML 1.1 30, 310
 XHTML 2.0 340, 359
 XHTML 2.0, modulares 342
 XHTML Basic 31, 339, 357
 XHTML mobile profile 31, 356
 XHTML, modulares 312
 Hyperlink 24
 Defaultpräsentation 143
 Hyperlinkcontainer 142, 143
 Hyperreferenz
 absoluter Pfad 87
 Basis-URI 87
 Dateiname 87
 Pfadangabe 87
 relativer Pfad 87
 Hypertextsystem 165

I

iMode HTML 351
 iMode XHTML 355
 Inhaltsbereich 168
 Inhaltsgruppe 42
 Inhaltsmodell 37, 41
 CDATA-Textinhalt 41
 Elementinhalt 41
 Gemischter Inhalt 41
 Inhaltsgruppe 42
 Inhaltsgruppe, Kardinalität 42
 PCDATA-Textinhalt 41

Schlüsselworte 41
 Inlinenelemente 137
 logische 137
 physische 137
 präsentative 137
 Innenabstand → CSS-Boxproperties, padding
 Institution
 What WG 952
 Institutionen
 CERN-Institut 21
 IETF (Internet Engineering Task Force) 296
 Internet Engineering Task Force (IETF) 25
 NILS, www.nils.org.au 280, 282
 Open Mobile Alliance 220, 356, 1167
 Openwave 1166
 Vision Australia 280
 W3C, World Wide Web Consortium 299
 Wap Forum 1167
 Web for all 280, 282
 WHAT Working Group 346
 World Wide Web Committee (W3C) 25
 Interaction Level Eventhandler 920
 Internationalisierung 915
 Internet Engineering Task Force (IETF) 25

K

Kommentare
 HTML 46
 in Stylesheets 102, 1002
 Konsistenz 112

L

Language Code 974
 Listen
 geordnete 150
 ungeordnete 150

M

Marke 36
 Auslassbarkeit 25, 46
 Begrenzer 36
 closing delimiter 36
 Delimiter 36
 Name 36
 open delimiter 36
 Marker-Attribut → Attribut

Index

Markupsprache 35
 Dokumenttyp-Definition 43
MIME-Typ 57
Mobilgeräte 219
 Arbeitsspeicher 219
 Dateneingabe 220
 Displaygröße 219
 Displaygrößen 221
 Eingabegerät 221
 Event-Unterstützung 221
 Prozessoren 219, 220
 Sonderfunktionen 221

N

Namensraum 52
 XHTML 52
Navigation 165
 Ad Hoc 167
 Globale 167
 Grafikgestützte 190
 Hierarchische 167
 Lokale 167
 mit Listen 173
 mit Textabsätzen 172
 mit Zeilenumbrüchen 171
 ungeordnete Liste 173
Navigationsgestaltung 166

O

open delimiter 36
Open Mobile Alliance 220, 356

P

Parameter-Entity 45
Parser 23
 Zwischencode 23
PCDATA-Textinhalt 42
Positionierung
 absolute 126
 Bezugskoordinatensystem 124
 feste 126
 relative 126, 127
 statische 126
Positionslayout 122
Programmiersprache 35

Q

Quelltext
 Reorganisation 120
Quelltextreihenfolge 119

R

Radiobuttons 157
Ruby 30, 311
Ruby-Spezifikation 312

S

Schema-Locator 83
Screenreader 243
Servicebereich 168
SGML 25
SGML-konforme Deklaration 44
SGML-Vereinbarung
 leere 47
Skiplinks → Sprunglinks
Small Screen Rendering Modus 233
Sprunglinks 120
Standardattribute 915
 Accessibility (IE) 947
 accesskey (IE) 948
 atomicselection (IE) 951
 class 916
 contenteditable (IE) 952
 Databinding (IE) 947, 949
 datafld (IE) 950
 dataformatas (IE) 950
 datapagesize (IE) 951
 datasrc (IE) 949
 dir 916
 Editing (IE) 947, 951
 EventHandler 919
 hidefocus (IE) 948
 id 917
 lang 917
 onblur 921
 onchange 921
 onclick 922
 ondblclick 923
 onfocus 923
 onkeydown 923
 onkeypress 924
 onkeyup 924
 onload 924
 onmousedown 925

onmousemove 925
onmouseout 925
onmouseover 926
onmouseup 926
onreset 926
onselect 927
onsubmit 927
onunload 927
style 918
tabindex (IE) 949
title 918
unselectable (IE) 953
xml:lang 919
 Startmarke 22, 36
 Starttag → Startmarke
 Steuerzeichen
 Maskierung 40
 Stylesprachen
 DSSSL *lite* 302
 DSSSL *Online* 302

T

Tabellen 151
 Textezug 136
 Textinhalt
 CDATA 41
 PCDATA 41
 Tools
 Absolute Color Picker 286
 Accessibility Tools 282
 Accessibility-CSS 1171
 Accessibility-Toolbar 1171
 Bildschirmleiste 283
 Browser-Plug-ins 287
 Color Contrast Analyzer 244, 248, 282
 Color Spy 285
 Colorblind Web Page Filter 1171
 ColorZilla (Firefox) 284
 Delorie Lynx Viewer 291
 Developer Toolbars 276
 DOM-Inspector (Mozilla) 287
 Farbkontrast Analyzer 282
 Firebug (Firefox) 288
 HTML Tidy 196
 HTML Tidy, accessibility-check 204
 HTML Tidy, add-xml-decl 203
 HTML Tidy, Batch-Verarbeitung 205
 HTML Tidy, char-encoding 204
 HTML Tidy, Clean 200
 HTML Tidy, Dateiausgabe 199

HTML Tidy, Dateimodifikation 200
 HTML Tidy, doctype 204
 HTML Tidy, drop-empty-paras 202
 HTML Tidy, drop-font-tags 202
 HTML Tidy, enclose-text 202
 HTML Tidy, GUI-Versionen 206
 HTML Tidy, Inegration in HTML-Kit 210
 HTML Tidy, Kommandozeile 197
 HTML Tidy, Konfigurationsdatei 205
 HTML Tidy, logical-emphasis 203
 HTML Tidy, lower-literals 203
 HTML Tidy, Optionen 201
 HTML Tidy, output-xhtml 203
 HTML Tidy, quote-ampersand 203
 HTML Tidy, quote-marks 203
 HTML Tidy, Stapelverarbeitung 205
 HTML Tidy, Tidy GUI (Windows) 206, 208
 HTML Tidy, Tidy UI (Windows) 206
 HTML Tidy, write-back 202
 HTML Tidy, XHTML-Ausgabe 199
 Inspect Element (Mozilla) 287
 InspectThis (Mozilla) 287
 Internet Explorer Developer Toolbar 278
 Measure It (Firefox) 283
 Openwave Phone Simulator 236
 Styleswitcher 1171
 Wave Toolbar 281
 Web Accessibility Toolbar 244, 280, 291
 Web Developer Toolbar 276
 Webdeveloper-Toolbar 1170
 Yellowpipe Lynx Viewer (Firefox) 290
 Yellowpipe Lynx Viewer (IE) 290

U

Überschriftenhierarchie 134
 Universalattribut 90
 Universalattribute → Standardattribute
 URI-String 971
 Bestandteile, funktionale 972
 Portbezeichner 972
 Querystring 972
 Zeichen, verbotene 973

V

Validatoren 1169
 Validome HTML Validator 1170
 W3C CSS-Validator 1169

Index

W3C Feed Validator 1169
W3C HTML Tidy 1170
W3C Link Checker 1170
W3C XHTML-Validator 1169
WDG HTML Validator 1169
Validierung 59
Verkürzungsvereinbarung 46

W

Web Accessibility Initiative (WAI) 241
Web Applications 304, 347
Web Content Accessibility Guidelines (WCAG) 241
Web Controls 347
Web Forms 347
Webbrowser 23
Webzines
 A List Apart 1172
 CSS Zen Garden 1172
Werte und Maßeinheiten 1023
Wurzelement 23, 52
 XHTML 51

X

XForms 359
 Aufbau eines XFormulars 375
 Auswahlfelder 380
 Auswahlfelder, Einfachauswahl 380
 Auswahlfelder, Mehrfachauswahl 380
 CSS-Selektoren 374
 CSS-Selektoren, CSS3 374
 CSS-Selektoren, XForms 374
 Datentypen 368
 Datentypen, eigene 368
 Datentypen, XML-Schema 368
 Datumfeld 396
 Elemente 361
 Entwicklungsumgebungen 361
 Entwicklungsumgebungen, XFormation 361
 erweitern 365
 Events 371
 Events, Error Indications 372
 Events, Eventdefinitionen 372
 Events, Initialization Events 372
 Events, Interaction Events 372
 Events, Notification Events 372
 Funktionen 369
 Funktionen, Auswertungskontext 370
 Funktionen, Boolesche 370
 Funktionen, Datum und Zeit 371
 Funktionen, DOM-Manipulation 369
 Funktionen, Nodesets 371
 Funktionen, Propertyvermittlung 371
 Funktionen, Zahlenwerte 370
 Hints 378
 Implementierungen 360
 Implementierungen, Chiba Project 361
 Implementierungen, clientseitige 360
 Implementierungen, FormFaces 360
 Implementierungen, serverseitige 361
 Implementierungen, X-Smiles 360
 Mehrfaches Submit 385
 Module 362
 Module, Action Module 364
 Module, Core Module 362
 Module, Extension Module 365
 Module, Form Controls Module 363
 Module, Group Module 366
 Module, MustUnderstand Module 366
 Module, Repeat Module 367
 Module, Switch Module 367
 Namensraum 362
 Range 394
 Reglementierung durch XML-Schema 379
 Übertragungsmethoden 384
 Vorgabewerte 377
 XPath 368
XFrames 64, 359, 399
 Aufruf eines XFrames-Dokuments 406
 Elemente 399
 Elemente, frame 404
 Elemente, frames 400
 Elemente, group 402
 Elemente, head 400
 Elemente, style 401
 Elemente, title 401
XHTML
 Host language 31
 Kleinschreibung 51
 modulares 30
XHTML 2.0-Attributmodule
 Bi-Directional 345
 Core 344
 Edit 345
 Embedding 345
 Events 345
 Hypertext 345
 I18N 344
 Image Map 345
 Media 345
 Metainformation 345

- Role* 345
 - Style* 345
 - XForms* 345
 - XHTML 2.0-Attributsammlungen
 - Common* 344
 - XHTML 2.0-Module
 - Core Module* 342
 - Document Modul* 342
 - extern definierte* 344
 - Handler Modul* 343
 - Hypertext Modul* 343
 - Image Modul* 343
 - List Modul* 343
 - Metainformation Modul* 343
 - Object Modul* 343
 - Role Access Modul* 343
 - Ruby Modul* 344
 - Structural Modul* 343
 - Style Sheet Modul* 343
 - Tables Modul* 344
 - Text Modul* 343
 - XForms Modul* 344
 - XML Events Modul* 344
 - XHTML Host Language 315
 - XHTML Basic* 339
 - XHTML Modularization 312
 - XHTML-Attributsammlungen
 - Attributsammlungen, leere* 314
 - Common* 313
 - Core* 313
 - Events* 314
 - I18N* 313
 - Style* 314
 - XHTML-Dokument
 - streng konformes* 59
 - XHTML-Module
 - Applet Modul* 328
 - Base Modul* 328
 - Basic Forms Modul* 321
 - Basic Forms Modul, Attribute* 322
 - Basic Tables Modul* 323
 - Basic Tables Modul, Attribute* 324
 - Bi-Directional Text Modul* 328
 - Client Side Image Map Modul, Attribute* 329
 - Client-Side Image Map Modul* 329
 - Core-Module* 314
 - Edit Modul* 330
 - Formularmodul* 325
 - Frames Modul* 330
 - Hypertextmodul* 318
 - IFrame Modul* 331
 - Imagemodul* 327
 - Intrinsic Events Modul* 331
 - Konformität* 315
 - Konformität, Host Language* 315
 - Konformität, Integration Set* 315
 - Legacy Modul* 332
 - Legacy Modul, Attribute* 333
 - Link Modul* 335
 - Listenmodul* 319
 - Meta Information Modul* 335
 - Name Identification Modul* 336
 - Object Modul* 336
 - Präsentationsmodul* 320
 - Ruby-Modul* 311
 - Scripting Modul* 336
 - Server Side Image Map Modul* 337
 - Strukturmodul* 316
 - Style Attribut Modul* 337
 - Style Attribut Modul, Ausnahmen* 337
 - Style Sheet Modul* 338
 - Tabellenmodul* 326
 - Target Modul* 338
 - Target Modul, benötigte Zusatzmodule* 338
 - Target Modul, betroffene Elemente* 338
 - Textmodul* 316
 - XHTML-Namensraum 52
 - Namensraumstring* 52
 - XML-Events 372
 - XML-konform 58
 - XML-konforme Deklaration 48
 - XML-Schema 83
- ## Z
-
- Zeilenabstand 136
 - Zeilenkontext 116
 - Zwischencode 23