

# Inhaltsverzeichnis

<b>1</b>	<b>HTML5 – Wer, wann und warum?</b>	<b>15</b>
1.1	Was ist HTML5 und woher kommt es? . . . . .	16
1.1.1	Eine kurze Geschichte von HTML . . . . .	16
1.1.2	Was ist XHTML? . . . . .	18
1.1.3	Die Rebellion der Browser-Allianz und die Entwicklung von HTML5 . . . . .	20
1.1.4	Das Ende von XHTML 2 und der Anfang der Unübersichtlichkeit . . . . .	21
1.1.5	Ein erweiterter Begriff von HTML5 . . . . .	22
1.2	Wozu brauchen wir HTML5? . . . . .	24
1.2.1	Vom alltäglichen Griff in die Trickkiste . . . . .	25
1.2.2	Ein Beispiel: Die Trickkiste von Spiegel Online . . . . .	25
1.2.3	...und die Zukunft ist auch schon heute . . . . .	27
1.3	Kann man HTML5 heute schon einsetzen? . . . . .	28
1.3.1	Was hat es mit dem Termin 2022 auf sich? . . . . .	29
1.3.2	Welche Teile von HTML5 kann man heute schon verwenden? . . . . .	29
1.3.3	Woher weiß ich, was ich schon verwenden kann und was nicht? . . . . .	32
1.3.4	Vier Prinzipien für die Benutzung von HTML5 . . . . .	34
1.3.5	Sollte man ab jetzt nur noch HTML5 verwenden? . . . . .	35
1.3.6	Ein Lese-HowTo für den Standard-Entwurf der WHATWG . . . . .	36
1.4	Frisch ans Werk . . . . .	38
1.4.1	Ausrüstungscheck . . . . .	38
1.4.2	Der Fahrplan . . . . .	39

<b>2</b>	<b>HTML5-Einstieg für Fortgeschrittene</b>	<b>41</b>
2.1	Ein erstes HTML5-Dokument . . . . .	42
2.1.1	Der Doctype . . . . .	42
2.1.2	Der Zeichensatz . . . . .	42
2.1.3	Die HTML-Syntax . . . . .	43
2.1.4	Die XHTML-Syntax . . . . .	45
2.1.5	Die Qual der Syntax-Wahl . . . . .	46
2.2	Ein erstes XHTML5-Dokument . . . . .	48
2.2.1	XHTML5 ist echtes XHTML . . . . .	49
2.2.2	XHTML-Fallstricke . . . . .	51
2.3	Inhaltliche Neuerungen von HTML5 . . . . .	53
2.3.1	Alter Wein in neuen Spezifikationen . . . . .	53
2.3.2	Neue Konzepte, Elemente und Ideen . . . . .	55
2.3.3	Neuerungen, die noch auf ihre Umsetzung warten . . . . .	56
2.3.4	Die Streichliste von HTML5 . . . . .	59
2.4	Kleine, aber interessante (und einsetzbare) Neuheiten im Schnelldurchlauf . . . . .	60
2.4.1	Neues vom <a>-Element . . . . .	60
2.4.2	Jedem Element seinen Datenspeicher . . . . .	62
2.4.3	accesskey, tabindex und WAI-ARIA . . . . .	63
2.4.4	WYSIWYG und Rechtschreibprüfung . . . . .	64
2.4.5	Neue und alte Attribute für das <script>-Element . . . . .	64
2.4.6	SVG und MathML einbetten . . . . .	67
2.5	Syntax ist nicht alles . . . . .	69
<b>3</b>	<b>Semantisches HTML5</b>	<b>71</b>
3.1	Dokumente strukturieren . . . . .	72
3.1.1	Kampf der Divitis und dem Überschriftenmangel . . . . .	72
3.1.2	Die neuen strukturierenden Elemente im Detail . . . . .	73
3.1.3	Überschriften und Abschnitte . . . . .	75
3.1.4	Beispiel: Struktur eines Blogs . . . . .	78
3.2	Auszeichnungen auf Text-Ebene . . . . .	82
3.2.1	Geänderte Elemente . . . . .	82
3.2.2	Gelöschte Elemente . . . . .	85

3.2.3	Reaktivierung der (ehemaligen) Präsentationselemente	86
3.2.4	Neue Elemente	89
3.2.5	Weitere neue Elemente und Änderungen	92
3.3	HTML-Elemente von morgen im Browser von heute	93
3.3.1	HTML5 und moderne Browser	93
3.3.2	HTML5 und die Internet Explorer 6–8	94
3.3.3	HTML5, Suchmaschinen, Screenreader und sonstige Abnehmer	99
3.4	Zugänglichkeit mit WAI-ARIA	99
3.4.1	Role- und ARIA-Attribute	100
3.4.2	Implizite ARIA-Semantik	101
3.5	Microdata in HTML5	102
3.5.1	Die Etablierten – RDFa und Microformats	102
3.5.2	HTML5-Microdata	103
3.5.3	Welches Microdata-Format benutzen?	106
3.6	Ruby-Annotationen	107
3.6.1	Ruby-Elemente in HTML5	107
3.6.2	Ruby-Markup im Browser	108
<b>4</b>	<b>HTML5-Formulare</b>	<b>111</b>
4.1	Neue Input-Elemente	112
4.1.1	search	112
4.1.2	telephone, url und email	112
4.1.3	number	113
4.1.4	Datums- und Zeiteingaben	114
4.1.5	range	116
4.1.6	color	116
4.1.7	Implementierungsstatus der neuen Input-Typen in verschiedenen Browsern	116
4.2	Weitere neue Formularelemente	119
4.2.1	<keygen>	119
4.2.2	<object>, <progress> und <meter>	120
4.2.3	<output>	121
4.3	Neue Funktionen von Formularen und Formularelementen	122
4.3.1	Eingabefelder außerhalb von Form-Elementen	122

4.3.2	Auto-Vervollständigung	123
4.3.3	Vorschläge mit <datalist>-Element und list-Attribut	124
4.3.4	Eingebaute Formularvalidierung	125
4.3.5	Mehrfach-Eingaben managen mit multiple	125
4.3.6	Platzhalter-Inhalte	126
4.3.7	Der dritte (optische) Checkbox-Status	126
4.3.8	Dynamische Formularattribute	127
4.4	Formularvalidierung	128
4.4.1	Möglichkeiten und Grenzen der Validierung	129
4.4.2	Eingebaute Validierungsregeln	130
4.4.3	Automatische Validierung	132
4.4.4	Manuelle Validierung	134
4.4.5	Eigene Validierungsregeln und -nachrichten	140
4.4.6	CSS-Exkurs: Formular- und validierungsrelevante Pseudoklassen	142
4.5	Was bringen uns HTML5-Formulare unterm Strich?	143
<b>5</b>	<b>Die Geolocation-API</b>	<b>145</b>
5.1	Positionen auslesen	146
5.1.1	Erfolgsfall	147
5.1.2	Fehlerbehandlung	147
5.1.3	Geolocation-Optionen	148
5.2	Positionsänderungen überwachen	148
5.3	Was tun mit all den anderen Browsern?	149
<b>6</b>	<b>Offline-Webanwendungen</b>	<b>151</b>
6.1	Vorbereitungen für ein Beispiel	152
6.2	Der Application Cache und das Cache Manifest	154
6.2.1	Cache Manifest anlegen	154
6.2.2	Manifest-Events	155
6.2.3	Fallbacks und Whitelists	158
6.2.4	Manifest-Tipps für Profis	160
6.2.5	Details der Application Cache API	162
6.3	DOM Storage und Online-/Offline-Events	164

6.3.1	Ajax für den regulären Betrieb . . . . .	165
6.3.2	Den Netzwerk-Status abfragen . . . . .	167
6.3.3	Änderungen lokal zwischenspeichern . . . . .	168
6.3.4	Daten mit dem Server synchronisieren . . . . .	171
6.3.5	Das Storage-Event . . . . .	173
6.3.6	Speicherort von DOM Storage . . . . .	174
6.4	Weitere Möglichkeiten der Offline-Speicherung . . . . .	175
6.4.1	Datenbanken in Webkit . . . . .	176
6.4.2	Storage für ältere IE: userData-Behaviour . . . . .	179
<b>7</b>	<b>Multimedia-Elemente und eingebettete Inhalte</b>	<b>181</b>
7.1	Video- und Audio-Elemente . . . . .	182
7.1.1	Das Audio-Element . . . . .	182
7.1.2	Das Video-Element . . . . .	183
7.1.3	Weitere gemeinsame Attribute . . . . .	185
7.2	Mehrere Medienquellen für ein Element . . . . .	185
7.2.1	Dateiauswahl nach Media-Typ . . . . .	186
7.2.2	Dateiauswahl nach Dateityp und Codec . . . . .	186
7.2.3	Die Codec-Problematik . . . . .	187
7.2.4	Eine „Lösung“ für alle Browser . . . . .	189
7.3	Die JavaScript-API . . . . .	190
7.3.1	Eigene Steuerungsbuttons . . . . .	190
7.3.2	Statuswerte eines Elements . . . . .	192
7.3.3	Steuerung der Abspielgeschwindigkeit . . . . .	194
7.3.4	Eine Navigationsfunktion . . . . .	195
7.3.5	Eine Fortschrittsanzeige . . . . .	197
7.3.6	Neue Medienquellen laden . . . . .	201
7.3.7	Events der Media-Elemente . . . . .	204
7.4	Flash und andere Plugin-Inhalte verwenden . . . . .	206
7.4.1	Was lange währt wird endlich Standard – das <embed>-Element . . . . .	207
7.4.2	<object>- und <param>-Element . . . . .	207
7.4.3	<embed> oder <object>? . . . . .	208

<b>8</b>	<b>Die Drag&amp;Drop-API</b>	<b>213</b>
8.1	Die API im Überblick	214
8.1.1	Anwendungsfälle für die API	215
8.1.2	Anwendungsfälle für Alternativen	216
8.2	Eine erste einfache Drag-Anwendung	216
8.2.1	Ein Element ziehbar (draggable) machen	216
8.2.2	Drag-Events	219
8.3	Drag mit Drop	223
8.3.1	Eigenheiten der Events für Drop-Ziele	224
8.3.2	Programmierung des Drop-Ziels	225
8.4	Cross-Website- und -Browser-Drag&Drop	227
8.4.1	Ziele der Events	228
8.4.2	Kreuz und quer dank dropEffect	229
8.4.3	Umsetzung im Internet Explorer	231
8.5	Datentransfers	237
8.5.1	Daten speichern und auslesen	237
8.5.2	Ziel-Auswahl anhand von Datentypen	240
8.6	Ein abschließendes (sinnvolles) Anwendungsbeispiel für Datentransfer	245
<b>9</b>	<b>Das Canvas-Element</b>	<b>249</b>
9.1	Das Canvas-Element	250
9.1.1	Canvas-Grundlagen	252
9.1.2	Transformationen	255
9.1.3	Globale Compositing-Einstellungen	261
9.1.4	Zeichenfunktionen des 2D-Context	263
9.1.5	Texte schreiben	274
9.1.6	Bilder	279
9.1.7	Die Bitmap Pixel für Pixel manipulieren	281
9.1.8	Muster	284
9.1.9	Farbverläufe	284
9.1.10	Bilddaten exportieren	286
9.1.11	Schatten	286
9.2	Probleme und Lösungen	288

9.2.1	Canvas im Internet Explorer . . . . .	288
9.2.2	Canvas-Text in älteren Browsern . . . . .	289
9.3	Canvas 3D . . . . .	290
9.3.1	Ein Beispiel für Software-3D im Eigenbau . . . . .	291
9.3.2	Hardwarebeschleunigtes Canvas-3D mit WebGL . . . . .	294
9.3.3	Wer braucht hardwarebeschleunigtes 3D im Browser? . . . . .	304
<b>10</b>	<b>Kommentare, Ausblick und HTML6</b>	<b>305</b>
10.1	HTML5 und die Zukunft des WWW . . . . .	306
10.1.1	Rüstzeug für komplexere Dokumente . . . . .	306
10.1.2	Der Browser als Anwendungsplattform . . . . .	306
10.2	Kritik an HTML5 . . . . .	307
10.2.1	Kritik am WHATWG-Prozess . . . . .	307
10.2.2	Kritikpunkte der „HTML5 Super Friends“ . . . . .	309
10.2.3	Wer braucht Webapps? . . . . .	310
10.2.4	HTML5 im Vergleich zu XHTML 2 . . . . .	311
10.2.5	Proprietäre Techniken und Zugänglichkeitsproblematik . . . . .	313
10.3	HTML6 und Nachfolger . . . . .	315
10.3.1	Das <device>-Element und seine Folgen . . . . .	316
10.3.2	Wird HTML5 je Webstandard? . . . . .	316
10.3.3	Schöne neue HTML-Welt . . . . .	317
<b>Anhang</b>		<b>319</b>
<b>A</b>	<b>Links zum Thema HTML5</b>	<b>321</b>
A.1	Ein kompaktes HTML5-Changelog . . . . .	322
<b>B</b>	<b>HTML5-Einstieg</b>	<b>329</b>
B.1	Ein HTML5-Dokument . . . . .	329
B.2	Ein XHTML5-Dokument . . . . .	330
B.3	Unterschiede zwischen HTML5 und XHTML5 . . . . .	330
B.4	Optionale HTML-Tags . . . . .	331
<b>C</b>	<b>Semantisches HTML5</b>	<b>333</b>
C.1	Neue strukturierende HTML5-Elemente . . . . .	333

C.2	HTML5-Elemente im Internet Explorer 6 bis 8 aktivieren . . . .	334
C.3	HTML5-Elemente mit CSS stylen . . . . .	334
C.4	HTML5-Elemente und implizierte ARIA-Semantik . . . . .	335
C.5	HTML5-Elemente mit überschreibbarer implizierter ARIA-Semantik . . . . .	338
C.6	Microdata . . . . .	339
C.7	Ruby-Annotationen . . . . .	340
<b>D</b>	<b>Formulare</b>	<b>341</b>
D.1	Formularelemente allgemein . . . . .	341
D.2	Das input-Element . . . . .	343
D.3	Das keygen-Element . . . . .	346
D.4	Das output-Element . . . . .	346
D.5	Validierungs-API . . . . .	347
<b>E</b>	<b>Geolocation-API</b>	<b>351</b>
E.1	getCurrentPosition()-Methode . . . . .	351
E.2	Änderungen überwachen . . . . .	353
E.3	Geolocation-Beispiel . . . . .	353
<b>F</b>	<b>Offline-Webanwendungen</b>	<b>355</b>
F.1	Kompatibilitätstabelle . . . . .	355
F.2	Manifest-Beispiel . . . . .	356
F.3	Manifest-Einbindung . . . . .	356
F.4	Application Cache API . . . . .	356
F.5	Netzwerk-Status . . . . .	358
F.6	Netzwerk-Events . . . . .	358
F.7	Storage-APIs . . . . .	358
F.8	Storage-Event . . . . .	359
<b>G</b>	<b>Media-Elemente</b>	<b>361</b>
G.1	Kompatibilität . . . . .	361
G.2	Media-Elemente . . . . .	362
G.3	source-Element . . . . .	362
G.4	JavaScript-API . . . . .	362
G.5	embed-Element . . . . .	366



G.6	object-Element	367
G.7	param-Element	367
G.8	Ein kompletter HTML5-Mediaplayer	367
<b>H</b>	<b>Drag&amp;Drop-API</b>	<b>373</b>
H.1	Kompatibilitätstabelle	373
H.2	Elemente ziehbar machen	374
H.3	Drag&Drop-Events	374
H.4	dataTransfer-Objekt	375
H.5	effectAllowed und dropEffect	376
<b>I</b>	<b>Canvas-Kurzreferenz</b>	<b>377</b>
I.1	Canvas-Element	377
I.2	Kompatibilitätstabelle	378
I.3	Browserunterstützung aktivieren	378
I.4	2D-Context	379
I.5	Transformationen	379
I.6	Compositing	380
I.7	Farben, Verläufe und Schatten	381
I.8	Rechtecke	382
I.9	Pfade	382
I.10	Linien	383
I.11	Bilder	384
I.12	Pixel-Manipulationen	384
I.13	Text	385
I.14	WebGL-Beispielanwendung	386