

Copyright (C) Open Source Press

Peter Kröner

HTML5

Webseiten innovativ und zukunftssicher

2. Auflage

Copyright (C) Open Source Press

Alle in diesem Buch enthaltenen Programme, Darstellungen und Informationen wurden nach bestem Wissen erstellt. Dennoch sind Fehler nicht ganz auszuschließen. Aus diesem Grunde sind die in dem vorliegenden Buch enthaltenen Informationen mit keiner Verpflichtung oder Garantie irgendeiner Art verbunden. Autor(en), Herausgeber, Übersetzer und Verlag übernehmen infolgedessen keine Verantwortung und werden keine daraus folgende Haftung übernehmen, die auf irgendeine Art aus der Benutzung dieser Informationen – oder Teilen davon – entsteht, auch nicht für die Verletzung von Patentrechten, die daraus resultieren können. Ebenso wenig übernehmen Autor(en) und Verlag die Gewähr dafür, dass die beschriebenen Verfahren usw. frei von Schutzrechten Dritter sind.

Die in diesem Werk wiedergegebenen Gebrauchsnamen, Handelsnamen, Warenbezeichnungen usw. werden ohne Gewährleistung der freien Verwendbarkeit benutzt und können auch ohne besondere Kennzeichnung eingetragene Marken oder Warenzeichen sein und als solche den gesetzlichen Bestimmungen unterliegen.

Dieses Werk ist urheberrechtlich geschützt. Alle Rechte, auch die der Übersetzung, des Nachdrucks und der Vervielfältigung des Buches – oder Teilen daraus – vorbehalten. Kein Teil des Werkes darf ohne schriftliche Genehmigung des Verlags in irgendeiner Form (Druck, Fotokopie, Mikrofilm oder einem anderen Verfahren), auch nicht für Zwecke der Unterrichtsgestaltung, reproduziert oder unter Verwendung elektronischer Systeme verarbeitet, vervielfältigt oder verbreitet werden.

Bibliografische Information Der Deutschen Nationalbibliothek

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

Copyright © 2011 Open Source Press, München
Gesamtlektorat: Sacha Storz
Satz: Open Source Press (L^AT_EX)
Umschlaggestaltung: Olga Saborov, Open Source Press
Gesamtherstellung: Kösel, Krugzell

ISBN 978-3-941841-34-5

<http://www.opensourcepress.de>

Inhaltsverzeichnis

1 HTML5 – Wer, wann und warum?	23
1.1 Was ist HTML5 und woher kommt es?	24
1.1.1 Eine kurze Geschichte von HTML	25
1.1.2 Was ist XHTML?	28
1.1.3 Die Rebellion der Browser-Allianz und die Entwicklung von HTML5	29
1.1.4 Das Ende von XHTML 2 und der Anfang der Unübersichtlichkeit	30
1.1.5 Ein erweiterter Begriff von HTML5	32
1.2 Wozu brauchen wir HTML5?	36
1.2.1 Vom alltäglichen Griff in die Trickkiste	37
1.2.2 Ein Beispiel: Die Trickkiste von Spiegel Online	37
1.2.3 . . . und die Zukunft ist auch schon heute	40
1.3 Kann man HTML5 heute schon einsetzen?	41
1.3.1 Was hat es mit dem Termin 2022 auf sich?	42
1.3.2 Welche Teile von HTML5 kann man heute schon verwenden?	42
1.3.3 Woher weiß ich, was ich schon verwenden kann und was nicht?	45
1.3.4 Vier Prinzipien für die Benutzung von HTML5	47
1.3.5 Sollte man ab jetzt nur noch HTML5 verwenden?	48
1.3.6 Ein Lese-Howto für den Standard-Entwurf der WHATWG	49
1.4 Frisch ans Werk	51
1.4.1 Ausrüstungscheck	52
1.4.2 Der Fahrplan	53

2	HTML5-Einstieg für Fortgeschrittene	55
2.1	Ein erstes HTML5-Dokument	56
2.1.1	Der Doctype	56
2.1.2	Der Zeichensatz	57
2.1.3	Die HTML-Syntax	57
2.1.4	Die XHTML-Syntax	63
2.1.5	Die Qual der Syntax-Wahl	64
2.2	Ein erstes XHTML5-Dokument	66
2.2.1	XHTML5 ist echtes XHTML	67
2.2.2	XHTML-Fallstricke	70
2.3	Inhaltliche Neuerungen von HTML5	71
2.3.1	Alter Wein in neuen Spezifikationen	71
2.3.2	Adaptierte Spezifikationen	72
2.3.3	Neue Konzepte, Elemente und Ideen	73
3	Kleine Änderungen, große Wirkung	75
3.1	Die Streichliste von HTML5	76
3.1.1	Gestrichen, aber spezifiziert	77
3.1.2	Nicht gestrichen, aber nicht empfohlen	77
3.2	Neue und nicht ganz so neue HTML-Attribute	77
3.2.1	Neue globale Attribute	78
3.2.2	Vereinheitlichung von class und id	78
3.2.3	Standard-Typen für <script> und <style>	79
3.2.4	Attribute von <a>, <link> und <area>	80
3.2.5	Steuerbare -Listen	83
3.2.6	WYSIWYG und Rechtschreibprüfung	87
3.2.7	Jedem Element seinen Datenspeicher	88
3.2.8	Kleines Attribut, große Wirkung	89
3.3	Praktische Auswirkungen des neuen Content Models	90
3.3.1	Links für alle!	91
3.3.2	<blockquote> ohne Absätze	91
3.4	Feintuning von Script-Ausführung	92
3.4.1	Das defer-Attribut	92
3.4.2	Das async-Attribut	94

3.4.3	async und defer im Vergleich	95
3.5	Aus der Nicht-Standard-Praxis in den Standard	96
3.5.1	getElementsByClassName()	96
3.5.2	activeElement und hasFocus()	97
3.5.3	innerHTML, outerHTML und insertAdjacentHTML()	97
3.6	Aus dem Standard in den Standard	101
3.6.1	SVG und MathML	101
3.6.2	WAI-ARIA	104
3.6.3	Ruby-Annotationen	105
3.6.4	XPath	108
3.7	Neuerungen ohne Implementierung	109
3.7.1	Frei definierbare Menüs mit <menu>	109
3.7.2	<details> und <summary>	110
3.8	Syntax ist nicht alles	111
4	Semantisches HTML5	113
4.1	Dokumente strukturieren	114
4.1.1	Kampf der Divitis und dem Überschriftenmangel	114
4.1.2	Die neuen strukturierenden Elemente im Detail	115
4.1.3	Überschriften und Abschnitte	119
4.1.4	Beispiel: Struktur eines Blogs	122
4.2	Auszeichnungen auf Text-Ebene	126
4.2.1	Geänderte Elemente	126
4.2.2	Gelöschte Elemente	133
4.2.3	Reaktivierung der (ehemaligen) Präsentationselemente	133
4.2.4	Neue Elemente	138
4.3	Markup-Empfehlungen	146
4.3.1	Tagclouds	146
4.3.2	Dialoge	146
4.3.3	Fußnoten	147
4.4	HTML-Elemente von morgen im Browser von heute	149
4.4.1	Moderne Browser	149
4.4.2	Internet Explorer 6–8	152
4.4.3	Neue Elemente und WAI-ARIA	153

4.4.4	HTML5 und Suchmaschinen	155
4.5	Strategien für den Umgang mit dem Internet Explorer	156
4.5.1	Strategie 1: Kapitulation	156
4.5.2	Strategie 2: JavaScript	157
4.5.3	Strategie 3: XHTML5 verwenden	159
4.5.4	Strategie 4: CSS-Kunstgriffe	163
4.6	HTML5-Microdata	166
4.6.1	Gruppen, Namen und Werte	166
4.6.2	Die Microdata-Vokabulare	169
4.6.3	Die Microdata-API	170
4.6.4	Die Alternativen – RDFa und Microformats	171
4.6.5	Welches Microdata-Format benutzen?	172
4.7	HTML5 jenseits von HTML	173
5	HTML5-Formulare	175
5.1	Neue Input-Typen	176
5.1.1	<code><input type="search"></code>	176
5.1.2	<code><input type="tel"></code> , <code><input type="url"></code> und <code><input type="email"></code>	178
5.1.3	<code><input type="number"></code>	178
5.1.4	Datums- und Zeiteingaben	180
5.1.5	<code><input type="range"></code>	181
5.1.6	<code><input type="color"></code>	182
5.1.7	Implementierungsstatus der neuen Input-Typen in verschiedenen Browsern	182
5.2	Weitere neue Formularelemente	185
5.2.1	<code><keygen></code>	186
5.2.2	<code><object></code> , <code><progress></code> und <code><meter></code>	187
5.2.3	<code><output></code>	187
5.3	Neue Funktionen von Formularen und Formularelementen	189
5.3.1	Eingabefelder außerhalb von Form-Elementen	189
5.3.2	Auto-Vervollständigung	190
5.3.3	Auto-Fokus	191
5.3.4	Vorschläge mit <code><datalist></code> -Element und <code>list-</code> Attribut	191

5.3.5	Eingebaute Formularvalidierung	192
5.3.6	Mehrfach-Eingaben managen mit <code>multiple</code>	193
5.3.7	Platzhalter-Inhalte	193
5.3.8	Der dritte Checkbox-Status	194
5.3.9	Überschreibbare Formularattribute	195
5.4	Formularvalidierung	196
5.4.1	Möglichkeiten und Grenzen der Validierung	197
5.4.2	Eingebaute Validierungsregeln nutzen	199
5.4.3	Automatische Validierung	200
5.4.4	Manuelle Validierung	203
5.4.5	Eigene Validierungsregeln und -nachrichten	208
5.4.6	CSS-Exkurs: Formular- und validierungsrelevante Pseudoklassen	211
5.5	Was bringen uns HTML5-Formulare unterm Strich?	212
5.5.1	Graceful Degradation für Widgets	213
5.5.2	Aufwärtskompatible Formularvalidierung	213
5.5.3	Probleme mit HTML5-Formularen	213
5.5.4	Auf zu neuen Ufern	216
6	Die Geolocation-API	217
6.1	Positionen auslesen	218
6.1.1	Erfolgsfall	219
6.1.2	Fehlerbehandlung	219
6.1.3	Geolocation-Optionen	220
6.2	Positionsänderungen überwachen	220
6.2.1	Positionsüberwachungs-Beispiel	221
6.3	Was tun mit all den alten Browsern?	222
7	Offline-Webanwendungen	225
7.1	Kurzüberblick Offline-Technologien	226
7.2	Vorbereitungen für ein Beispiel	227
7.2.1	HTML-Gerüst	227
7.2.2	Brauchbare Browser	228
7.2.3	Speichern mit Ajax	229
7.3	Der Application Cache	231

7.3.1	Das Cache Manifest	232
7.3.2	Manifest-Events	234
7.3.3	Ersatzinhalte und Whitelists	236
7.3.4	Manifest-Management für Fortgeschrittene	238
7.3.5	Details der Application Cache API	241
7.4	Online oder offline?	243
7.4.1	Den Netzwerk-Status abfragen	243
7.4.2	Online- und Offline-Events	243
7.5	DOM Storage	244
7.5.1	Die Storage-APIs	245
7.5.2	DOM Storage als Zwischenspeicher verwenden	247
7.5.3	Ajax nur im Online-Modus	247
7.5.4	Daten mit dem Server synchronisieren	249
7.6	Weitere Möglichkeiten der Offline-Speicherung im Schnell- durchlauf	253
7.6.1	Indexed DB – Datenbank der Zukunft	253
7.6.2	Web SQL – der lebende tote Standard	254
7.6.3	User-Data-Behavior – das IE-Fossil	257
7.6.4	jStorage – das Framework	259
8	Multimedia-Elemente und eingebettete Inhalte	261
8.1	<video>- und <audio>-Elemente	262
8.1.1	Das <audio>-Element	263
8.1.2	Das <video>-Element	264
8.1.3	Weitere gemeinsame Attribute	266
8.2	Mehrere Medienquellen für ein Element	267
8.2.1	Dateiauswahl nach Media-Typ	268
8.2.2	Dateiauswahl nach Dateityp und Codec	268
8.2.3	Die Codec-Problematik	269
8.2.4	Eine „Lösung“ für alle Browser	274
8.3	Die JavaScript-API	275
8.3.1	Eigene Steuerungsbuttons	275
8.3.2	Statuswerte eines Elements	277
8.3.3	Steuerung der Abspielgeschwindigkeit	279

8.3.4	Eine Navigationsfunktion	280
8.3.5	Eine Fortschrittsanzeige	282
8.3.6	Neue Medienquellen laden	290
8.3.7	Events der Media-Elemente	293
8.4	Fertige HTML5-Mediaplayer	295
8.5	Zukunftsmusik	295
8.5.1	Video-Annotationen mit dem <track>-Element	296
8.5.2	Die Audio Data API	296
8.6	Flash und andere Plugin-Inhalte verwenden	297
8.6.1	Was lange währt wird endlich Standard – das <embed>- Element	297
8.6.2	<object>- und <param>-Element	298
8.6.3	<embed> oder <object>?.	299
9	Die Drag&Drop-API	303
9.1	Die API im Überblick	304
9.1.1	Anwendungsfälle für die API	305
9.1.2	Anwendungsfälle für Alternativen	306
9.2	Eine erste einfache Drag-Anwendung	306
9.2.1	Ein Element ziehbar (draggable) machen	307
9.2.2	Drag-Events	309
9.3	Drag mit Drop	313
9.3.1	Eigenheiten der Events für Drop-Ziele	314
9.3.2	Programmierung des Drop-Ziels	315
9.4	Cross-Website- und -Browser-Drag&Drop	317
9.4.1	Ziele der Events	317
9.4.2	Kreuz und quer dank dropEffect	319
9.4.3	Umsetzung im Internet Explorer	321
9.5	Datentransfers	326
9.5.1	Daten speichern und auslesen	327
9.5.2	Ziel-Auswahl anhand von Datentypen	329
9.6	Ein abschließendes Anwendungsbeispiel für Datentransfer	335
10	Die File API	337
10.1	Dateien und Dateilisten	338

10.1.1	Dateilisten	338
10.1.2	Dateien und Blobs	339
10.2	Dateien einlesen	341
10.2.1	FileReader-Methoden	341
10.2.2	FileReader-Events und -Statuseigenschaften	342
10.2.3	Fehlerbehandlung	345
10.2.4	Synchrone FileReader	346
10.3	Beispielanwendung: Der Browser als Archivierungsprogramm .	347
10.3.1	JSZip	347
10.3.2	Drag&Drop-Programmierung	348
10.3.3	Dateien einlesen	348
10.4	Browserunterstützung	350
11	Das <canvas>-Element	353
11.1	Canvas-Grundlagen	354
11.1.1	Das <canvas>-Element	354
11.1.2	Das Wesen der Canvas-APIs	356
11.1.3	Rendering Contexts	356
11.2	Der 2D-Context	357
11.2.1	Zeichenfläche, Koordinaten und Context-Zustand	357
11.2.2	Einfache Formen zeichnen	358
11.2.3	Den Context-Zustand managen	360
11.2.4	Transformationen	361
11.2.5	Globale Compositing-Einstellungen	367
11.2.6	Bereiche der Bitmap löschen	370
11.2.7	Rechteckige Rahmen	373
11.2.8	Linien und Kurven mit Pfaden zeichnen	374
11.2.9	Clipping-Masken	383
11.2.10	Texte schreiben	384
11.2.11	Bilder	390
11.2.12	Pixel-Manipulationen	392
11.2.13	Muster und Farbverläufe	395
11.2.14	Bilddaten exportieren	398
11.2.15	Schatten	399

11.3 Probleme und Lösungen	400
11.3.1 Canvas für die Internet Explorer 6 bis 8	400
11.3.2 Canvas-Text in älteren Browsern	402
11.4 Canvas-Frameworks	403
11.4.1 CanvasQuery: Do-it-yourself-Framework in unter 70 Zeilen	404
11.4.2 EaselJS: Objekte und Animationen	409
11.4.3 RGraph: Graphen mit Canvas	410
11.5 Canvas 3D	411
11.5.1 Ein Beispiel für Software-3D im Eigenbau	411
11.5.2 Hardwarebeschleunigtes Canvas-3D mit WebGL	415
11.6 Fazit: Was bringt uns das <canvas>-Element?	416
12 Web Workers	419
12.1 Welches Problem lösen Web Workers?	420
12.2 Web Workers im Einsatz	421
12.2.1 Web Workers ausprobieren	421
12.2.2 Das Web-Workers-Prinzip	422
12.2.3 Wuzelbehandlung mit Web Workers	424
12.3 Shared Workers	425
12.3.1 Verbindungen herstellen	425
12.3.2 Nachrichten via Ports senden	426
12.3.3 Mehrere Verbindungen auf einmal nutzen	427
12.3.4 Mehrere Verbindungen, mehrere SharedWorkers	429
12.4 Probleme und Lösungen	430
12.4.1 Fehlerbehandlung	430
12.4.2 Web-Workers-Emulation bei fehlender Browserunterstützung	430
12.5 Fortgeschrittene Web-Workers-Techniken	432
12.5.1 Worker abbrechen	432
12.5.2 Komplexe Daten versenden	432
12.5.3 JavaScript-Bibliotheken in Workern verwenden	433
12.5.4 Verteiltes Rechnen und Sub-Workers	434
12.6 Fazit	437

13 Unterstützung für HTML5-Features erkennen und nachrüsten	439
13.1 Angewandte Feature-Erkennung	440
13.1.1 HTML5-Feature-Erkennung als HTML5-Feature	440
13.1.2 Feature-Erkennung via DOM-Eigenschaft	440
13.1.3 Feature-Erkennung via Browser	441
13.2 Was taugt Feature-Erkennung?	442
13.2.1 Nicht erkennbare Features	442
13.2.2 Vorhandensein != Funktionieren	443
13.3 Was tun, wenn ein Feature fehlt?	443
13.3.1 Graceful Degradation	444
13.3.2 Selektiver Zugang	444
13.3.3 Nachrüsten – aber wie?	445
13.4 Fertige Polyfills und Feature-Erkennungs-Referenz	447
14 Kritik, Kommentare, Ausblick und HTML 6	449
14.1 HTML5 und die Zukunft des WWW	450
14.1.1 Rüstzeug für komplexere Websites	450
14.1.2 Der Browser als Anwendungsplattform	451
14.1.3 Im Konzert mit ECMAScript 5/Harmony und CSS3	451
14.2 Kritik an HTML5	452
14.2.1 Kritik am WHATWG-Prozess	452
14.2.2 Kritikpunkte der „HTML5 Super Friends“	454
14.2.3 Kritik von Douglas Crockford	455
14.2.4 Die Komplexität von HTML5	456
14.2.5 Proprietäre Techniken und Zugänglichkeitsproblematik	457
14.2.6 Wer braucht Webapps?	458
14.3 Alternativen zu HTML5?	459
14.3.1 HTML5 im Vergleich mit XHTML 2	459
14.3.2 HTML5 im Vergleich mit Flash	461
14.3.3 Sind Ex-post-facto-Standards der einzige Weg?	462
14.4 Wann kommt HTML 6?	463
14.4.1 Das <device>-Element und seine Folgen	463
14.4.2 Wird HTML5 je Webstandard?	464

Anhang	467
A HTML5 – Wer, wann und warum?	469
A.1 Links zu HTML5	469
A.2 Ein kompaktes HTML5-Changelog	470
B HTML5-Einstieg	477
B.1 Links	477
B.2 Ein HTML5-Dokument	478
B.3 Ein XHTML5-Dokument	478
B.4 Unterschiede zwischen HTML5 und XHTML5	478
B.5 Zeichensatz festlegen	479
B.6 Optionale HTML-Tags	480
C Kleine Änderungen, große Wirkung	481
C.1 Steuerbare <code></code> -Listen	481
C.2 WYSIWYG und Rechtschreibprüfung	483
C.3 <code>data-*</code> -Attribute	483
C.4 Content Model	484
C.5 <code>defer</code> und <code>async</code> für <code><script></code>	486
C.6 Neu standardisierte Elemente, DOM-Eigenschaften und Methoden	487
C.7 Adaptierte Spezifikationen	488
C.8 Elemente für Ruby-Annotation	489
D Semantisches HTML5	491
D.1 Links	491
D.2 Neue strukturierende HTML5-Elemente	492
D.3 Neue semantische Elemente auf Textebene	492
D.4 HTML5-Elemente und implizierte ARIA-Semantik	494
D.5 HTML5-Elemente mit überschreibbarer implizierter ARIA-Semantik	497
D.6 HTML5-Elemente im Internet Explorer 6 bis 8 aktivieren	498
D.7 HTML5-Elemente standardkonform stylen	499
D.8 Microdata	500

E	Formulare	503
E.1	Links	503
E.2	Das <input>-Element	504
E.3	Neue Attribute von Eingabeelementen	506
E.4	Sonstige neue Elemente	509
E.5	Validierungs-API	511
F	Geolocation-API	519
F.1	Links	519
F.2	Browserunterstützung	519
F.3	Positionen bestimmen	520
F.4	Positionen überwachen	521
G	Offline-Webanwendungen	523
G.1	Links	523
G.2	Kompatibilitätstabelle	524
G.3	Cache Manifest	525
G.4	Netzwerk-Status	528
G.5	Netzwerk-Events	528
G.6	DOM Storage	529
H	Media-Elemente	533
H.1	Links	533
H.2	Media-Elemente	536
H.3	Das <source>-Element	537
H.4	JavaScript-API	537
H.5	embed-Element	542
H.6	object-Element	542
H.7	param-Element	543
H.8	Ein kompletter HTML5-Mediaplayer	543
I	Drag&Drop-API	547
I.1	Kompatibilitätstabelle	547
I.2	Elemente ziehbar machen	548
I.3	Drag&Drop-Events	548

I.4	dataTransfer-Objekt	549
I.5	effectAllowed und dropEffect	550
J	File API	551
J.1	Links	551
J.2	Browserunterstützung	551
J.3	FileList-Objekt	552
J.4	Blob-Objekt	552
J.5	File-Objekt	553
J.6	FileReader-Interface	553
J.7	FileErrors und FileExceptions	555
J.8	Beispielanwendung: Der Browser als Archivierungsprogramm .	555
K	Canvas-Kurzreferenz	557
K.1	Links	557
K.2	Das <canvas>-Element	559
K.3	Kompatibilitätstabelle	559
K.4	2D-Context	560
K.5	Transformationen	560
K.6	Compositing	561
K.7	Farben, Verläufe und Schatten	561
K.8	Rechtecke	563
K.9	Pfade	563
K.10	Linien	564
K.11	Bilder	565
K.12	Pixel-Manipulationen	565
K.13	Text	566
K.14	Mini-2D-Framework	567
L	WebWorkers	571
L.1	Links	571
L.2	Kompatibilitätstabelle	572
L.3	Web-Worker-API	572
L.4	SharedWorker-API	574
L.5	Web-Workers-Beispiel	574

M Unterstützung für HTML5-Features erkennen	577
M.1 Links	577
M.2 Polyfills	577
M.3 Unterstützung für HTML5-Elemente erkennen	578
M.4 Audio- und Videocodecs erkennen	580
M.5 HTML5-APIs	581