

Mit
Stellenanzeigen



KOMPAKT

WEBDESIGN

2/2012

Ein Sonderheft des Magazins für professionelle Informationstechnik, www.ix.de

€ 8,99

HTML5 und CSS3

Was neu an HTML5 ist:

Animationstools

Web Workers, Websockets

Datenbanken im Browser

Formulare, Multimedia-Integration

Cascading Stylesheets 3:

Media Queries und Selektoren

**Animationen, Transitions,
Transformationen**

LessCSS und Modernizr

Mobiles Web:

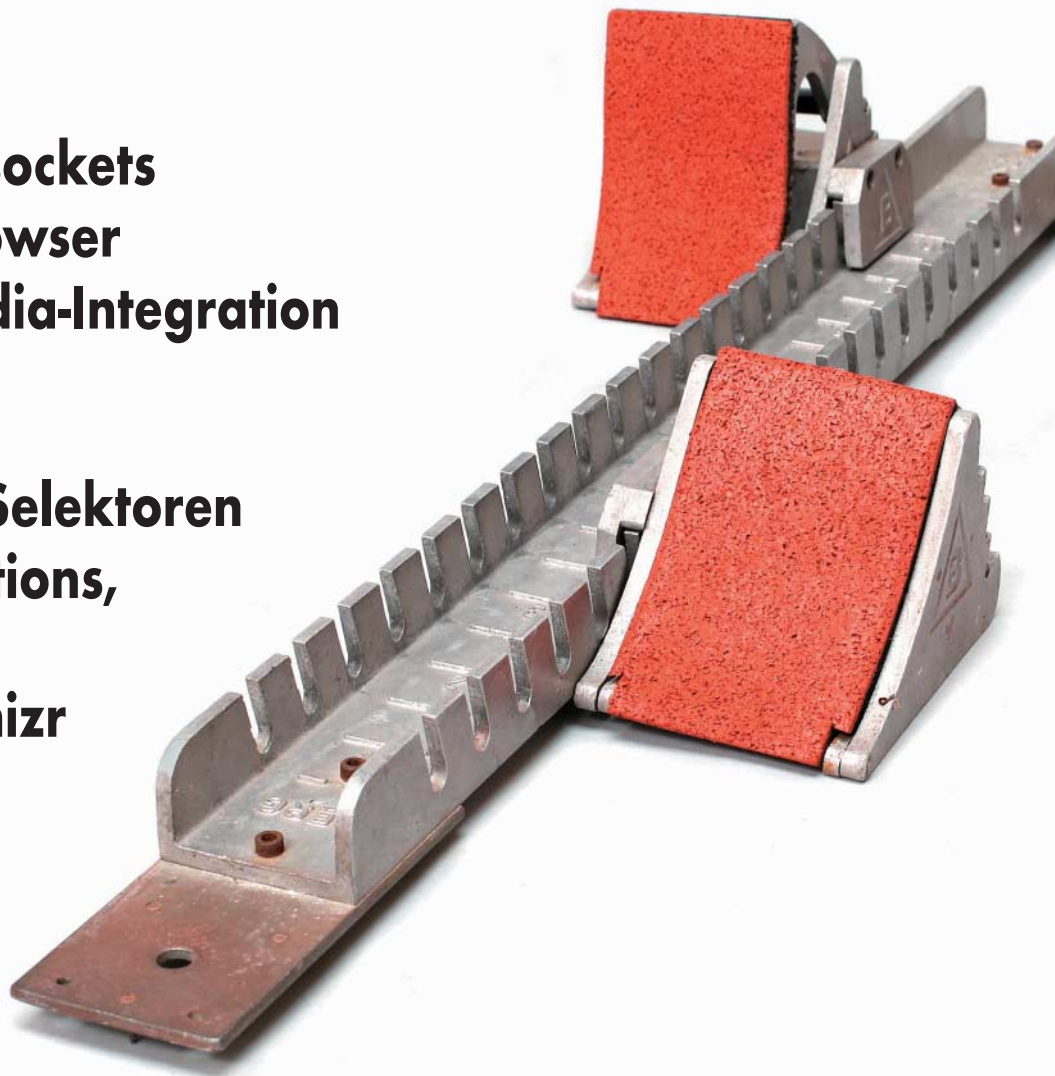
jQuery Mobile

Webapps-Tutorial

Dritte Dimension im Web:

WebGL – Tools und Tutorial

X3DOM und XML3D



15% sparen mit
Rabattcode:
„HTML5“



Powering the Voice of Information Technology



Java Magazin
www.java-magazin.de/abo



dot.Net Magazin
www.dotnet-magazin.de/abo



Entwickler Magazin
www.entwickler-magazin.de/abo



PHP Magazin
www.php-magazin.de/abo



Eclipse Magazin
www.eclipse-magazin.de/abo



Business Technology
www.bt-magazin.de/abo



Mobile Technology
www.mobiletechmag.de/abo



Android360
www.android360.de/abo



SharePoint Magazin
www.sharepoint-magazin.de/abo

Sichern Sie sich unsere Magazine im Premiumabonnement mit exklusivem **Sonderrabatt von 15%**. Geben Sie einfach bei Ihrer Bestellung in den jeweiligen Formularen den Rabattcode „HTML5“ an.

Das bewegte Web

In den ungefähr 20 Jahren, die es das World Wide Web gibt, hat es sich im Grunde immer in Bewegung befunden. Kaum hatten die ersten Browser das Licht des Web erblickt und zeigten reinen Text mit ein paar Links, fügten die Entwickler Bilder und Formulare hinzu, wenig später Tabellen. Unsäglichkeiten wie die HTML-Elemente *blink* und *marquee* seien hier verschwiegen.

Schon früh kamen die Mitglieder des World Wide Web Consortium (W3C) auf die Idee, den Inhalt der Dokumente von Anweisungen zu ihrer Präsentation zu trennen. Kurz vor Weihnachten 1996 verabschiedete das Konsortium die erste Fassung der Cascading Stylesheets. Gut gemeint, aber was die Browserhersteller daraus machten, verhiess nichts Gutes. Webdesigner mussten viel Arbeit investieren, um alle Viewer zu „bedienen“.

Viel später nahmen ein paar der Hersteller das Heft in die Hand, indem sie sich in der Web Hypertext Application Technology Working Group (WHATWG) zusammenschlossen, um die nächste Generation der Hypertext Markup Language (HTML) ohne den Segen des W3C zu forcieren. Und schließlich lenkte das Konsortium ein: Es beendete seine Arbeit an einem XHTML-Nachfolger und übernahm das von der WHATWG Erreichte als Entwurf für HTML5. Mittlerweile liegen viele Working Drafts für einzelne Aspekte dieser Auszeichnungssprache vor. Sie decken so Unterschiedliches ab wie neue Elemente (von *nav* bis *canvas*), erweiterte Formulare, Multimedia-Integration, Datenbanken im Browser und Threads in JavaScript.

Gleichzeitig befinden sich etliche Entwürfe für Module der CSS3 in Arbeit, wenigstens kann als abgeschlossen gelten. Aber vieles können Webdesigner schon ausprobieren, weil die Browserhersteller vieles früh implementiert haben. Und in anderen Fällen ist absehbar, dass dieser Status bald erreicht sein dürfte. Es könnten tatsächlich demnächst Zeiten beginnen, in denen Webdesigner nicht mehr ständig überlegen müssen, wie sie welchen Browser berücksichtigen.

Stattdessen sinnieren sie vielleicht, wie sie die dritte Dimension auf Smartphones hinbekommen können oder Webseiten für solche Geräte so aufbereiten, dass sie dort gut aussehen.

Offenkundig gibt es für Webautoren und -designer nach wie vor einiges zu tun, und sie müssen wie immer auf dem Laufenden bleiben. Dieses Sonderheft der *iX* soll dabei helfen, neuere Webtechniken nachvollziehen zu können, früh zu sehen, was man demnächst zumindest ausprobieren und auf den diversen Viewern prüfen sollte. Kurz: nicht hinter dem State of the Art zurückzubleiben.

HENNING BEHME



HTML5

Fast 15 Jahre lang galt HTML 4 als Standard. Doch das Web hat sich weiterentwickelt, ist bunter und lauter geworden. Neue Multimedia-Elemente sollen dem Rechnung tragen. Aber auch „ernsthafte“ Webanwendungen, die nach Formularen oder clientseitigen Datenbanken verlangen, berücksichtigt HTML5. Was Webentwickler über den neuen Standard wissen sollten

ab Seite 7



CSS3

Eine Webseite sollte auf dem Desktop anders aussehen als auf dem Smartphone. Kein Problem, wenn man die CSS3 Media Queries nutzt. Neben HTML5 hat das W3C die neue Version der Cascading Stylesheets vorgestellt. Zwar liegen diverse Module von CSS3 bisher nur als Entwurf vor, aber die Browserhersteller haben schon viel davon implementiert. Was man wie nutzen kann

ab Seite 73



HTML5

Webentwicklung	
Aussichten für HTML5 und CSS3 in der Praxis	8
Hypertext Markup Language	
Mehr Elemente, weniger Einigkeit	12
Webformulare	
Neue Eingabeelemente	22
Multimedia	
Audio und Video in der Markup Language	32
Accessibility	
Barrierefreiheit mit HTML5	37
Datenbanken	
Im Browser Daten lokal verwalten	41
Web Workers	
Parallelverarbeitung: Threads mit JavaScript	50
Interaktion im Web	
Zusammenarbeit in Echtzeit	56
Webclients	
Wie HTML5 Rich Internet Applications verändert	60
Kommunikation	
Bidirektionale Verbindung per WebSocket	65
Animation	
Mehr Bewegung mit Tools auf der Basis von Webstandards	68

CSS3

Stil & Form	
Wohin die Reise der Cascading Stylesheets geht	74
Media Queries	
Geräteabhängige Stilvorgaben	78
Webtypografie	
WOFF: Mehr Schriften ins Web per CSS	83
Selektoren	
HTML-Elemente gezielt ansprechen	86
Less	
JavaScript-Bibliothek fürs Erstellen von Cascading Stylesheets	89
Modernizr	
Browser per JavaScript steuern	94
Animation	
Übergänge, Umwandlungen und Animationen mit CSS3	98
Shader	
Kommende Filtereffekte	107
Grafik	
Historie	
3D-Darstellungen im Webbrowser	110
X3DOM & XML3D	
Deklaratives 3D in HTML5	112
Transformationen und Interaktion	120

WebGL

Interaktive 3D-Szenen stehen schon seit den Kindertagen des Web auf der Wunschliste der Anwender. Mittlerweile ist selbst das kleinste Smartphone in der Lage, anspruchsvolle Grafiken ansprechend darzustellen. Der neue Standard WebGL bietet Webentwicklern jetzt leistungsfähige Alternativen zu den proprietären Plug-ins. Eine Einführung in WebGL sowie Alternativen, um weniger hardwarenah zu programmieren

ab Seite 109



JavaScript-Frameworks

Bibliotheken für WebGL 125

Grafikprogrammierung

2D-Grafik mit canvas und JavaScript 132

Spieleentwicklung

Pingpong mit JavaScript und HTML5 canvas 135

WebGL-Tutorial

Grundlagen 140

Texturen, Animation und ModelView-Matrix 144

Beleuchtung, Transparenz, Funktionsplotter 149

Mobile

JavaScript-Bibliothek

Einsatz von HTML5 mit jQuery Mobile 156

Tutorial: Vom Web zur App

Webseitenoptimierung für das iPhone 161

iPhone-Apps mit Webtechniken 166

Native iPhone-Apps mit Webtechniken 172

Sonstiges

Editorial 3

Inserentenverzeichnis 6

Impressum 6

Die Software zum Heft

Trial-Versionen

BBEdit: Web- und Text-Editor für Mac OS X

Dreamweaver: Adobes Webdesign-Tool für Mac OS X und Windows

Dynamic HTML Editor: WYSIWYG-Editor zur Website-Gestaltung

Expression Studio 4 Web Professional: Expression Web, Encoder und Design – Microsofts Webdesign-Suite

FDT 5: Development-Toolkit für Windows, Mac OS X und Linux – spezielle 30-Tage-Lizenz für iX-Leser – Details auf der DVD

HTMLPad: HTML-, CSS-, JavaScript- und XHTML-Editor

phase 5: HTML-Editor, für Privatanwender und Schulen kostenlos

PHPedit: deutsche und englische Version mit Erweiterungen für FTP, CVS und Subversion, PHPUnit, eZ Publish, Prado und Symfony

RapidWeaver: Webdesign-Tool für Mac OS X

UltraCompare: Vergleichstool für Webseiten

UltraEdit: Web-Editor für Windows, Mac OS X und diverse Linuxe

Web Architect 9: Web-Editor (integrierter HTML-Editor und CSS-Editor) kombiniert mit einem Desktop-CMS für Windows

Freie Tools

Boilerplate: Tipps und Tricks zur Webgestaltung

JoApp: App-Framework für HTML5

Maqetta: In-Browser-Editor für HTML5

SproutCore: Entwicklungs-Framework für Mac OS X

Libraries

GLGE: JavaScript-Bibliothek zur einfachen WebGL-Nutzung

PhiloGL: WebGL-Framework für Datenvisualisierung und Spieleentwicklung

SceneJS: 3D Scene Graph Engine für WebGL

Dokumente und Spezifikationen

RFCs zu CSS und HTTP

Spezifikationen zu CSS3, HTML5, WebGL, X3DOM, X3D, XML3D

Hinweis für Käufer der digitalen Ausgaben dieses Sonderhefts sowie für Käufer in der Schweiz und in Österreich: Sie finden das Image der DVD zum Download unter <ftp://ftp.heise.de/pub/ix/html5/dvd/dvd.iso>



para//el 2012

Eine Veranstaltung von iX, heise developer und dpunkt

Die neue Softwarekonferenz für Parallel Programming, Concurrency und Multi-Core-Systeme.

**Frühbucherrabatt
bis 31.03.2012!
(750 Euro anstatt 890 Euro)**

FÜR:

- // Softwarearchitekten
- // Softwareentwickler
- // Projektleiter
- // IT-Strategen
- // Forscher

TEILNEHMER

- // erfahren mehr über wichtige Grundlagen und wesentliche Aspekte paralleler und nebenläufiger Programmierung.
- // erhalten aus erster Hand wertvolle Ratschläge über den Einsatz von Produkten, Techniken und Mechanismen zum Ausschöpfen des Potenzials parallelisierter Softwarearchitekturen.
- // lernen ausgewiesene Experten der Multi-Core-Programierung kennen.
- // haben die Möglichkeit zum Networking und zum Erfahrungsaustausch.

Jetzt anmelden!

Veranstalter:



heise
Developer

dpunkt.verlag

www.parallel2012.de



iX kompakt Webdesign

Postfach 61 04 07, 30604 Hannover;
Karl-Wiechert-Allee 10, 30625 Hannover

Redaktion

Telefon: 05 11/53 52-387, Fax: 05 11/53 52-361, E-Mail: post@ix.de

Chefredakteur: Jürgen Seeger (-386)

Konzeption und redaktionelle Leitung: Kersten Auel (-367, ka@ix.de),
Henning Behme (-374, hb@ix.de)

Autoren dieser Ausgabe: Henning Behme, Johannes Behr, Helge Grimm, Tobias Günther, Stephan Heller, Michael Jendryschik, Yvonne Jung, Nikolaos Kaintantzis, Kai König, David Linner, Stefan Mintert, Helmut Moritz, Thomas Mühlichen, Stefan Neufeind, Philipp Slusallek, Kristian Sons, Peter Strohm, Gerhard Völkl

Redaktionsassistentz: Carmen Lehmann (-387), Michael Mentzel (-153)

Korrektorat: Wiebke Preuß

DTP-Produktion: Enrico Eisert, Wiebke Preuß, Matthias Timm,
Hinstorff Verlag, Rostock; Anja Fischer, Heise Zeitschriften Verlag

Titelidee: iX

Fotografie: Martin Klauss Fotografie, Despetal / Barfelde

Verlag

Heise Zeitschriften Verlag GmbH & Co. KG, Postfach 61 04 07, 30604 Hannover; Karl-Wiechert-Allee 10, 30625 Hannover; Telefon: 05 11/53 52-0, Telefax: 05 11/53 52-129

Geschäftsführer: Ansgar Heise, Dr. Alfons Schröder

Mitglied der Geschäftsleitung: Beate Gerold

Verlagsleiter: Dr. Alfons Schröder

Anzeigenleitung: Michael Hanke (-167), E-Mail: michael.hanke@heise.de

Leiter Vertrieb und Marketing: André Lux (-299)

Teamleitung Herstellung: Bianca Nagel

Druck: Dierichs Druck + Media GmbH & Co. KG, Kassel

DVD-Herstellungsleitung: Klaus Ditze

Verantwortlich: Textteil: Jürgen Seeger;
Anzeigenteil: Michael Hanke

iX kompakt Webdesign 2/2012: Einzelpreis € 12,90,
Österreich € 9,90, Schweiz CHF 18,00, Luxemburg: € 10,45

Eine Haftung für die Richtigkeit der Veröffentlichungen kann trotz sorgfältiger Prüfung durch die Redaktion vom Herausgeber nicht übernommen werden. Kein Teil dieser Publikation darf ohne ausdrückliche schriftliche Genehmigung des Verlages verbreitet werden; das schließt ausdrücklich auch die Veröffentlichung auf Websites ein.

Printed in Germany

© Copyright by Heise Zeitschriften Verlag GmbH & Co. KG

Die Inserenten

Comet	www.comet.de	93
dpunkt	www.dpunkt.de	49
Galileo Press	www.galileo-press.de	77
Goneo	www.goneo.de	35
IfaD	www.ifad.de	53
Infoware	www.infoware.de	39
Ipoque	www.ipoque.com	97
Hella Gutmann	www.hella-gutmann.com	47
Marketing Factory	www.marketing-factory.de	27
Mirabyte	www.mirabyte.com	35
PEAK-System Technik	www.peak-system.com	47
Pearson Education	www.addison-wesley.de	55
plentySystems	www.plentymarkets.eu	85
Software & Support Verlag	www.software-support.biz	2
Studiosus Reisen München	www.studiosus.com	29
Thomas Krenn	www.thomas-krenn.com	179
Trivago	www.trivago.de	25
Widas	www.widas.de	45
Wiethe	www.wiethe.com	27, 29

Die hier abgedruckten Seitenzahlen sind nicht verbindlich. Redaktionelle Gründe können Änderungen erforderlich machen.



HTML5: Die Gegenwart der Zukunft

HTML5 ist „da“. Es dauert aber noch zehn Jahre, bis das World Wide Web Consortium die Sprache vollendet hat. Oder sie bleibt für immer unfertig, was unproblematisch wäre, wenn die Browserhersteller weiterhin implementieren, was halbwegs sicher Bestandteil des Standards sein dürfte. Ob neue Elemente, Formulare oder Multimedia-Integration – vieles, das sich noch im Prozess der Standardisierung befindet, können Webdesigner schon jetzt in ihren Seiten verwenden. Oder in allernächster Zukunft.

Aussichten für HTML5 und CSS3 in der Praxis	8
Neue Elemente, weniger Einigkeit	12
Neue Eingabeelemente in HTML5	22
Audio und Video in HTML5	32
Barrierefreiheit mit HTML5	37
Im Browser Daten lokal verwalten	41
Parallelverarbeitung: Threads mit JavaScript	50
Zusammenarbeit in Echtzeit mit HTML5	56
Wie HTML5 Rich Internet Applications verändert	60
Bidirektionale Verbindung per WebSocket	65
Mehr Bewegung mit Tools auf der Basis von Webstandards	68

Aussichten für HTML5 und CSS3 in der Praxis

Schon gestartet

Stephan Heller



In den vergangenen zehn Jahren hat das Web sich dramatisch verändert – sogar in Richtung des Guten. Das liegt zum großen Teil an der besseren Struktur durch XHTML und an der Trennung des Inhalts von der Präsentation. Und die Aussichten sind weiterhin vielversprechend.

Vor 15 Jahren war es noch etwas Besonderes, wenn jemand einen Internetzugang hatte und sich mit einem 14k-Modem sowie merkwürdigen Geräuschen in diese virtuelle Welt einwählte. Dass es ein Leben ohne Internet gibt, ist für viele heute unvorstellbar. Eine Kinderfrage wie „Mutti, wenn ihr früher keine Computer hattet, wie seid ihr da ins Internet gekommen?“ lässt einen schmunzeln und zeigt zugleich das Selbstverständnis heutigen Kommunikations- und Informationsverhaltens.

Zur Jahrtausendwende und der Zeit der – vermeintlich – geplatzten Dotcom-Blase bestand das Handwerkzeug des Webdesigners aus HTML 4.01 und den Cascading Style sheets 2.1. Die HTML-Version hatte das World Wide Web Consortium (W3C) am 24. Dezember 1999 zur Recommendation erklärt und damit als verbindlichen Standard festgelegt (allerdings damals nicht „Standard“ genannt). Webdesigner haben ihre Seiten in dieser Zeit danach aufgebaut. Allerdings war vom semantischen Web noch keine Rede. Wie auch. Es war die Zeit, in der Internet Explorer 5.5 und Netscape 4.79 die Arbeit des Webdesigners bestimmten und damit klare Grenzen setzten. Ziel und Aufgabe des Webdesigners war es, irgendwie das Layout hinzubekommen. Wie, spielte zu der Zeit keine Rolle, Hauptsache hübsch.

Apropos Webdesigner. Bis heute gibt es keine zuverlässige und verbindliche Berufsbezeichnung für Menschen, die Webseiten generieren. Zwar haben sich Bezeichnungen wie Frontend-Entwickler, Webworker, Web Developer oder Site Developer etabliert, trotzdem entscheidet selbst in großen IT-Unternehmen jeder noch selber, mit welchem Berufstitel er sich wohlfühlt. Und nach wie vor lässt jede der Bezeichnun-

gen nicht zwangsläufig darauf schließen, was derjenige wirklich kann, ob und wo die Grenze zum Grafikerdesigner und zum Programmierer zu finden ist.

Um die Jahrtausendwende bestimmten HTML-Attribute das Layout, legten Farben, Schriftgrößen, Textausrichtungen fest. Editoren wie Dreamweaver speicherten derlei damals im HTML-Dokument – schreckliche Quelltexte. Mit Zähneknirschen erinnert man sich an Microsofts HTML-Editor Frontpage, dessen Erzeugnisse mit heutiger Qualität von Quelltexten nichts zu tun hatte. Auch die Photoshop-Funktion „als Webseite exportieren“ zerschnitt Grafiken in viele kleine und lieferte ebenfalls nicht wirklich schönes HTML.

Layout Tausender Seiten durch ein Stylesheet

CSS2 stand seit 1998 zur Verfügung, seit 2002 arbeitete das W3C an der Version 2.1, was die Webentwicklung ein kleines Stück revolutionierte. Mit der Entwicklung des Standards begannen die Browserhersteller, CSS-Unterstützung zu implementieren. Die Vorzüge lagen auf der Hand. Eine einzige CSS-Datei konnte das Layout von Hunderten, Tausenden von HTML-Seiten steuern. Das bot theoretisch die Möglichkeit, mit dem Austauschen der CSS-Datei das komplette Layout einer Homepage zu verändern.

Theoretisch deswegen, weil ein solcher Austausch in der reinen Form sicherlich höchst selten, wenn überhaupt vorgekommen ist. Große Unternehmen nehmen spätestens nach drei oder vier Jahren einen Relaunch ihres Website-

Designs vor. Und der beschränkt sich nicht auf CSS. Der HTML-Code wird immer mit überarbeitet und an den aktuellen Stand der Technik angepasst. In der Regel kommen bei einem Relaunch außerdem neue Funktionen hinzu, Datenbanken werden angepasst und derlei mehr. Selbst wenn nur ein sogenannter „Rebrush“ einer Homepage ansteht, eine Art Neuanstrich, bleibt das HTML davon nicht unberührt.

Am 26. Januar 2000 veröffentlichte das W3C den XHTML 1.0 Standard, die überarbeitete Version am 1. August 2002. Auf HTML 4.01 fußend hatten die Herausgeber die Sprache überarbeitet und vor allem bereinigt. XHTML lehnt sich einerseits an das XML-Format an und verlangt von daher saubere Quelltexte. Vor allem aber steht XHTML für den Versuch, alle Sünden aus HTML 4.01 zu tilgen. Im Rückblick ist XHTML 1.0 die Wende zu einem qualitativ hochwertigen Web. Die Aufgabe des Webdesigners war es nicht mehr, Webseiten irgendwie zu realisieren, sondern sie gut zu gestalten. Eins der wenigen Qualitätskriterien einer Webseite ist ein korrekter Quelltext, den Validatoren vom W3C oder Validome überprüfen können. Durch ihn stellen Webentwickler ihre Kompetenz unter Beweis.

Webdesign wurde aber auch bezüglich der dahinterstehenden Konzepte immer professioneller. War HTML 4.01 wie ein Kessel Buntes mit wild vermischem Layout und Inhalt, brachte XHTML immer mehr Ordnung in die Webgestaltung. Mit der Konsequenz: Das neue HTML diente ausschließlich der Struktur und lieferte die Inhalte; CSS, komplett vom HTML entkoppelt, lieferte das Layout. JavaScript als letzter Teil der Dreifaltigkeit war für die Verbesserung der Bedienung, sprich: der Usability zuständig. In den ersten fünf Jahren des Jahrtausends haben Autoren Webseiten immer mehr nach dem XHTML-1.0-Standard umgesetzt. Barrierefreiheit wurde langsam ein Thema, was letztendlich mit dem semantischen Web einhergeht, denn ohne irgendeine Struktur haben Screenreader- oder Braillezeilennutzer kaum eine Chance, eine Webseite effektiv zu besuchen.

Bei Suchmaschinen wollen alle reüssieren

Letztendlich ist aber die HTML-Semantik essentiell für die größten blinden Anwender der Welt: Suchmaschinen. Wo man sich am Anfang des Jahrtausends über die *meta*-Tags wie *keywords* und *description* Gedanken machte, um die Seiten angemessen zu platzieren, kamen Google und Co schnell dahinter, dass der Inhalt der *meta*-Tags nicht zwingend etwas mit dem Inhalt der Webseite zu tun haben muss. So waren Porno oder Sex beliebte Schlüsselwörter, selbst wenn die Seite Himbeer-tee verkaufen wollte – nur um Suchmaschinen anzulocken.

Es ist heute kein großes Geheimnis mehr, dass ein gut gewählter *title* und eine aussagekräftige *h1* wesentlich mehr Einfluss auf das Ranking als irgendein *meta*-Tag haben. So war es auch nicht unbedingt die Liebe zu Quelltexten, die die Qualität der Webseiten verbesserte, sondern die Anforderung an den Webdesigner, bei Suchmaschinen auf Seite eins landen zu müssen.

Was die Standards anbelangt, so schien die Weiterentwicklung in der Mitte des ersten Jahrzehnts in den Winterschlaf zu gehen. Der XHTML-Standard war solide und bot alles, was man brauchte. CSS 2.1 war ebenfalls recht stabil, obwohl das W3C diesen Standard erst 2011 zur offiziellen Empfehlung erklärte.

Einfache, farbige
Version des W3C-HTML5-
Logos (Abb. 1)



Das Konsortium beschäftigte sich längst mit der Entwicklung von XHTML 2.0, um HTML noch weiter zu verbessern. Allerdings gab es seit 2004 Browserentwickler, die mit der eher starren Herangehensweise des W3C nicht einverstanden waren. Vertreter von Opera, Apple und Mozilla gründeten die WHATWG (Web Hypertext Application Technology Working Group) und begannen mit der Entwicklung eigener Standards: Web Forms 2.0 und Web Apps 1.0. Beide sollten Erweiterungen für HTML beschreiben. Schließlich haben sie beide Entwürfe in HTML5 zusammengefasst.

Es kann manchmal nur einen Standard geben

Aber auch das W3C begann irgendwann, neben der Entwicklung von XHTML 2.0 mit einem HTML 5-Standard (zur Abgrenzung zur WHATWG zunächst mit einem Leerzeichen vor der 5). Während Webdesigner draußen in der Welt beständig mit XHTML 1.0 arbeiten konnten, war es unklar, wohin die Reise geht, schien eine neuer Standard in weiter Ferne zu sein. Irgendwann haben jedoch ein paar kluge Köpfe realisiert, dass es keine zwei konkurrierenden Standards geben kann. 2007 änderte das W3C seine Haltung, da man einsah, dass es nie zu einem XHTML-2.0-Standard kommen dürfte. Das Konsortium hat den Entwurf der WHATWG als Grundlage eines neuen HTML5 akzeptiert, und seitdem arbeiten beide Gruppen gemeinsam am neuen Standard (beide Organisationen fungieren nach wie vor als Herausgeber von HTML5). Schließlich ist es für den Webentwickler am Arbeitsplatz gleichgültig, wer den Standard entwickelt. Hauptsache, es gibt irgendwann etwas Verbindliches.

Von XHTML 1.0 unterscheidet HTML5, dass Letzteres kein HTML-Element als böse abstrahlt, sondern im Grunde alles zugelassen ist, was in HTML 4.01 oder XHTML 1.0 erlaubt war. Das mag den einen oder anderen Quelltextfetschisten in Rage bringen. Pragmatisch betrachtet: Auch wenn in HTML5 beispielsweise das Element *acronym* nicht mehr auftaucht und mit *abbr* unter dessen Namen vereinigt wird, ist es letztendlich ziemlich gleichgültig, wenn jemand ein *acronym* einsetzt, da alle Browser dieses Element unterstützen. Und kein Browserhersteller „explementiert“ ein funktionierendes Feature, nur weil ein Standard es nicht mehr enthält.

Damit definiert sich HTML5 als letzter Standard, eine Version 6 kann es nicht geben, da alle Elemente, die es in der Vergangenheit gab und die in Zukunft hinzukommen, automatisch Teil von HTML5 sind, das damit die Menge aller verfügbaren Elemente darstellt.

Dies entspricht ohnehin der Realität. Es ist nicht so, dass das Konsortium oder die WHATWG den Standard entwickeln, diesen verabschieden und danach die Browserhersteller ihn implementieren. Im Gegenteil, Browserhersteller dürften das, was sie eingebaut haben, weiterhin unterstützen, selbst wenn dies laut Standard nicht zulässig wäre. Sie integrieren neue Eigenschaften, und wenn sie gut sind, gehen sie in die Standardentwicklung ein.

Bedeutung für die heutige Webentwicklung

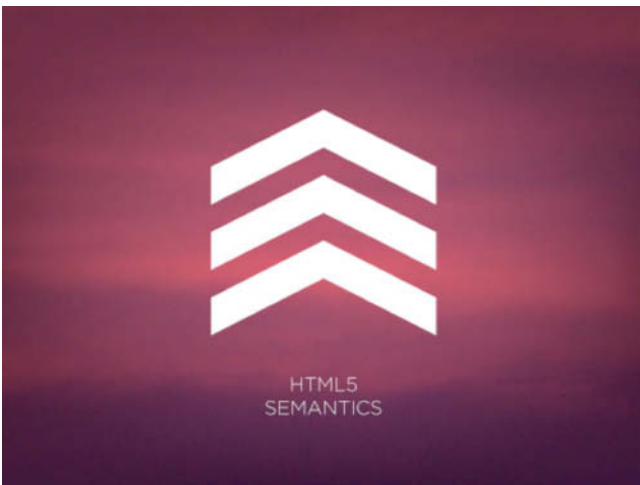
Im Alltag schaut kein Entwickler im Standard nach, was er bietet, und entwickelt anschließend ein Frontend. Sondern er nimmt ein neues Feature und prüft, welcher Browser es unterstützt. Hat nur ein kleiner Teil der Browser ein neues Attribut implementiert, kommt es für den realen Einsatz erst mal nicht infrage. Unterstützt aber ein großer Teil ein neues Feature, muss der Webautor prüfen, wie aufwendig ein Fallback für die restlichen Browser ist, oder ob man keinen Funktionsverlust hinnehmen muss, wenn einer für ein bestimmtes Verhalten keinen Support bietet.

Eine revolutionäre Neuerung von HTML5 sind die *audio* und *video*-Elemente (siehe S. 32), mit dem Webautoren Audios beziehungsweise Videos ohne externe (Flash-)Player ins

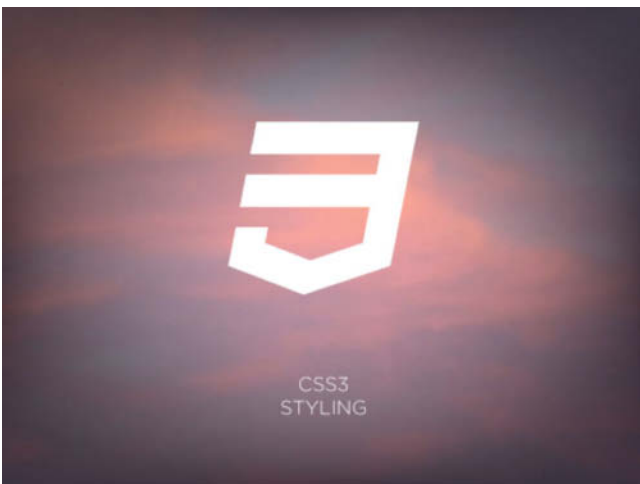
HTML einbinden können. Der Support hängt neben dem Element selbst vom Audio- beziehungsweise Video-Format ab. Eine zuverlässige Einbindung erreicht man nur, wenn verschiedene diesbezügliche Formate hinterlegt sind und zusätzlich für ältere Browser ein Flash-Player als Fallback eingebunden ist. Mit anderen Worten: Heute sollten Webautoren einen funktionierenden Player noch nicht aus Webseiten rausnehmen, wenn das HTML5-Element keine Verbesserung, sondern eher Schwierigkeiten bringt.

Mit HTML5 beginnt außerdem ein neues Zeitalter für Formulare beziehungsweise Eingabefelder (siehe S. 22). So ergibt `<input type="date"/>` ein Eingabefeld, das einen Datums-picker zur Auswahl des Tages anbietet und das aufwendige Einbinden eines Kalenders über JavaScript erspart. Da allerdings nur Opera und Safari dieses neue *date*-Attribut des *input*-Elements unterstützen, ist wiederum ein Fallback für die anderen Browser notwendig, einschließlich einer sinnvollen Validierung, wenn Anwender das Datum per Hand eingeben müssen. Davon abgesehen ist das *date*-Attribut kein Freibrief dafür, ein Datum serverseitig nicht mehr zu prüfen.

Kleine, aber feine neue Attribute für die *input*-Felder sind *autofocus* und *placeholder*. Sie ersparen die Arbeit mit JavaScript, um Platzhalter in das Feld zu schreiben, den Wechsel von Platzhalter und Benutzereingabe zu kontrollieren oder den Fokus auf ein Feld zu legen. Da es hier lediglich um die Verbesserung der Bedienung geht, eine Seite aber funktional nicht darunter leidet, wenn ein Browser die Attribute nicht kennt, können Webautoren sie getrost einsetzen. Allerdings kann man hier wieder den Standpunkt vertreten, dass man diese Usability-Features nach wie vor durch vorhandenes JavaScript steuern sollte, bis alle wichtigen Browser die HTML5-Attribute flächendeckend beherrschen.



Quelle: W3C [c]



Analog zu einem der HTML5-Logos haben W3C-Designer ein gesondertes CSS3-Emblem kreiert (Abb. 2).

Was Webautoren guten Gewissens einsetzen können

Dies sind drei Beispiele für grandiose Features von HTML5, die Browser aber noch so mäßig implementiert haben, dass man ihren Einsatz immer kritisch prüfen muss. Was aber die Tätigkeit des Webentwicklers verändern dürfte, denn wo er früher gegen die Unzulänglichkeiten des Internet Explorer 6 kämpfte, muss er in Zukunft herausfinden, welches HTML5-Element er in der Praxis mit gutem Gewissen einsetzen kann. Werkzeuge wie Modernizr und LessCSS helfen dabei (siehe S. 94 und 89).

Für CSS3 gilt Vergleichbares, aber nicht in dieser kritischen Dimension wie für HTML5. Beim W3C seit 2000 in der Entwicklung, bietet CSS3 im Vergleich zu CSS 2.1 mehr als doppelt so viele Eigenschaften. Vor allem Transitions, Animations und Transforms (siehe S. 98) bringen Bewegung in die Weboberfläche. Mouseover-Effekte müssen nicht mehr Entwickler über aufwendiges JavaScript implementieren, sondern sie sind jetzt Bestandteil der Stylesheets. Hier ist die mangelnde Unterstützung älterer Browser kein Drama, da die einen Effekt nicht anzeigen, was die Funktion aber nicht beeinträchtigen muss.

Was revolutionäres Potenzial hat: Komplette Diashows oder Akkordeons lassen sich durch Transitions oder Animations realisieren – womit Stylesheets mit jQuery konkurrieren. Gerade, da Ladezeiten wegen Smart-Phones und potenziell geringer Bandbreite wieder eine Rolle spielen, muss sich jeder Frontend-Entwickler die Frage stellen, ob er eine fette jQuery-Bibliothek einbinden will, wenn er denselben

Effekt mit ein paar Zeilen CSS erreichen kann. Zumal man damit kein fallback mehr anbieten muss, da diese Features nicht mehr von JavaScript abhängig sind.

Prognosen, wie viel jQuery durch CSS3 wieder verschwindet, sind allerdings Kaffeesatzleserei. Es steht zu befürchten, dass sich jQuery schon so weit etabliert hat, dass viele eine Umstellung auf CSS weder als notwendig noch als gleichwertig einstufen. Allerdings kann man gespannt darauf sein, welche CSS3-Unterstützung der Internet Explorer 10 bieten wird. Hier sehen die Aussichten gut aus, da sich Microsoft mit der nächsten IE-Version einiges vorgenommen hat. Das wiederum könnte die Tätigkeit eines Webdesigners und den Einsatz von CSS3 wesentlich verändern.

Was einer nicht kann, fällt unter den Tisch

Im Praxisalltag scheinen von daher die Karten neu gemischt zu sein und man darf gespannt sein, wohin die Reise geht. Man kann nicht zwingend erwarten, dass in die tägliche Arbeit alles einfließt, was die neuen Standards bieten. Dafür, wie die Web-Welt in fünf Jahren aussieht, spielen andere Kriterien eine Rolle. An der CSS-Eigenschaft *position* und deren Wert *fixed* kann man aus der Erfahrung vielleicht eine Prognose für die Zukunft stellen. Allein die Tatsache, dass der Internet Explorer 6 diesen Wert nicht unterstützte, hat dazu geführt, dass er im Alltag kaum auftaucht. Was früher über Frames realisierbar war, könnte man heute über *position: fixed* realisieren. Macht aber keiner, da der Wert in vielen Köpfen als nicht unterstützt gespeichert ist. Dadurch kann man mutmaßen, dass Top-Features, sowohl aus HTML5 oder CSS3, ebenfalls nicht zum Einsatz kommen dürften, weil es einen (mehr oder wichtigen) Browser gibt, der es nicht umsetzen kann.

Gerade was CSS3-Eigenschaften anbelangt, befindet sich der Standard noch in der Entwicklung, sodass man nur sukzessive einzelne Eigenschaften verwenden kann. Es gibt Eigenschaften wie *overflow-x* und *overflow-y*, die schon der Inter-

Mohl Design hat das einfache HTML5-Logo als Vorlage für ein CSS3-Emblem genommen (Abb. 3)



net Explorer 6 unterstützte, und solche wie *grid-columns* oder *grid-rows*, die noch so experimentell sind, dass die Gefahr besteht, dass sie wieder aus dem Standard verschwinden. Die Eigenschaft *text-outline* ist ein Beispiel dafür und nicht die einzige. Legt man die Entwicklungszeit von CSS 2.1 bis zur W3C-Empfehlung zugrunde, dürfte CSS3 noch Jahre auf sich warten lassen, bis der Standard final ist.

Ein ganz anderer Blick geht in Richtung der Dreieinigkeit von HTML für Struktur, CSS für Layout und JavaScript für die Verbesserung der Bedienbarkeit. HTML5 und CSS3 übernehmen in Zukunft Funktionen, die bislang nur über JavaScript zu realisieren waren. Das oben genannte *autofocus*-Attribut oder eine Transition zum sanften Aufklappen einer Box verschieben das Verhalten einer Webseite wieder nach HTML beziehungsweise CSS. Eine deutliche Trennung von Struktur, Verhalten und Layout verwischt sich, was sich auch im Alltag des Entwicklers niederschlagen und die Anzahl der Best-Practice-Wege wieder erhöhen dürfte.

Dennoch bewirken HTML5 und CSS3 etwas Wesentliches: Sie orientieren sich an der Nutzbarkeit einer Seite und optimieren Effekte, die den Besuch einer Webseite erlebnisreicher machen können. Es geht nicht mehr um hübsche Quelltexte, sondern darum, Seiten zu entwickeln, die den Ansprüchen der Surfer gerecht werden. Letztlich kann nur das Ziel sein, die Bedienung einer Seite so komfortabel zu gestalten, dass Einkäufe intuitiv und einfach abzuwickeln sind oder es einfach Freude bereitet, auf einer Seite zu verweilen.

Wie Webdesign im Jahre 2020 oder 2015 aussehen kann, ist schwer zu steuern und nicht zuverlässig zu prognostizieren. Flash gehört der Vergangenheit an. Ob jQuery seine jetzige Rolle behält, ist fraglich. Und letztendlich kennt noch niemand die Techniken, die in den nächsten Jahren hinzukommen. Es bleibt deshalb ein spannendes Webzeitalter. Was nicht verwunderlich ist, denn die Branche ist noch jung, und die Entwicklung der letzten zehn Jahre hätte sicherlich niemand so vorausgesehen. (hb)

Onlinequellen

W3C

- [a] HTML5 www.w3.org/TR/html5/
- [b] Aktueller Stand der HTML5-Spezifikation <http://dev.w3.org/html5/spec/>
- [c] HTML5-Logos www.w3.org/html/logo/
- [d] CSS-Homepage www.w3.org/Style/CSS/
- [e] CSS3-Spezifikationen www.w3.org/Style/CSS/specs

WHATWG

- [f] HTML Living Standard www.whatwg.org/specs/web-apps/current-work/multipage/
- [g] HTML5 für Webentwickler <http://developers.whatwg.org/>

andere

- [h] CSS3 Gets a New Logo www.css3.info/css3-gets-a-new-logo/
- [i] CSS3: opacité des images et couleur du texte sélectionné <http://blog.nicolasgut.com/2011/05/03/css3-opacite-des-images-et-couleur-du-texte-selectionne/>




STEPHAN HELLER

ist seit 2001 als Webentwickler, seit 2004 als freischaffender Webdesigner tätig. Momentan arbeitet er an einer umfassenden CSS-Referenz (www.css-wiki.com) sowie an einem HTML5-CSS3-Workshop, der 2012 bei dpunkt erscheint.

Alle Links: www.ix.de/ix1214008





Neue Elemente,
weniger Einigkeit

Freie Bahn

Stefan Mintert

Seit 1997 gibt die heute noch gültige Version 4 der Websprache HTML vor, wie Webseiten auszuzeichnen sind. Seit 2004 arbeiten Browserhersteller an der nächsten Version 5. Vielleicht gibt es die nie – oder ständig ein bisschen mehr, denn vieles kann man heute schon nutzen.

Auf den ersten Blick ist der Begriff HTML5 leicht einzuordnen. Es sieht nun einmal aus wie die neue Version von HTML. Jeder, der sich ein wenig mit dem Thema beschäftigt hat, weiß, dass die vorherige Version die Nummer 4 trägt. Tatsächlich ist bei HTML5 aber vieles anders.

Der erste Unterschied ist die Herkunft. Nachdem sich 1994 das World Wide Web Consortium (W3C) gegründet hatte, galt dieser Firmenverbund als Hüter der HTML-Spezifikation. Was das W3C vorgab, galt als das „richtige“ HTML. Das Konsortium selbst sprach lange nicht von Standards oder einer Norm, immer von einer „Empfehlung“ (Recommendation). Aber de facto war es das längst.

Mehr oder weniger hielten sich die Browserhersteller an das, was in der Spezifikation stand. Ob jemals ein Browser die HTML-4-Spezifikation vollständig umsetzte, darf man allerdings bezweifeln. 1998, nach dem Erscheinen von XML auf der Bildfläche, hat das W3C seine Metasprache konsequent zur Formulierung neuer HTML-Dialekte verwendet. Die größte Bekanntheit hat zweifellos XHTML 1.0 erlangt (X wie üblich für extensible/erweiterbar), inhaltlich gegenüber HTML 4 unverändert, nur folgt die Syntax nun den

schärferen XML-Regeln, nicht mehr denen der Vorgängerin SGML.

Seit XHTML 1.0 ist nichts Wesentliches passiert, der Versuch einer Definition von XHTML 2.0 ist gescheitert. Nach achtjähriger Arbeit hat das W3C die Arbeitsgruppe Ende 2010 aufgelöst. Diesen um den Jahrtausendwechsel begonnenen Stillstand hat eine Gruppe von Personen 2004 mit der Gründung der WHATWG beantwortet.

Besorgte Einzelpersonen

Auf ihrer Website beschreibt sich die Gruppe selbst: „Die Web Hypertext Application Technology Working Group (WHATWG) ist eine wachsende Gemeinschaft von Personen, die an der Entwicklung des Web interessiert ist. Sie konzentriert sich auf die Entwicklung von HTML und von APIs, die für Webanwendungen gebraucht werden. Die WHATWG wurde von Einzelpersonen von Apple, der Mozilla-Stiftung und Opera Software im Jahre 2004 nach einem W3C-Workshop gegründet. Apple, Mozilla und Opera waren zuneh-

mend besorgt über die vom W3C eingeschlagene Richtung von XHTML, mangelndes Interesse an HTML und die scheinbare Missachtung der Bedürfnisse von Autoren. Als Erwiderung nahm sich diese Organisationen vor, die Sache in die Hand zu nehmen, und so war die Web Hypertext Application Technology Working Group geboren.“ Alle Verweise sind über den iX-Link zu finden.

Konkurrenz, die das Geschäft belebt

Es ist offensichtlich, dass die WHATWG eine Konkurrenz zum W3C darstellt. Das Konsortium hat auch mal als Gemeinschaft von HTML-Aktivisten angefangen. Deshalb darf man gespannt sein, wie sich beide Organisationen in Zukunft entwickeln. Letztlich aber entscheiden die Browser, wie der Markup-Standard aussieht. Bemerkenswert ist zumindest, dass die HTML5-Spezifikation bei beiden Organisationen parallel veröffentlicht wird. Bei der WHATWG trägt sie den Titel „HTML Living Standard“, das W3C-Dokument heißt „HTML5 A vocabulary and associated APIs for HTML and XHTML“, zumindest im Moment noch (Januar 2012). Seitens der WHATWG ist der Name kein Marketing-Trick, sondern Dogma. Nach eigenen Aussagen arbeitet sie nicht mehr an HTML5, sondern an HTML – keine Versionsnummern mehr, keine Meilensteine. In diesem Zustand befand sich die Hypertext Markup Language schon einmal – zuletzt circa 1993 (ein paar kritische Anmerkungen zur HTML-Entwicklung enthält der Kasten „HTML5: Fortschritt oder Rückschritt?“). Ob es zu einer Konsolidierung der HTML-Welt führt, wenn man klar kommunizierbare Versionen abschafft, sei dahingestellt. Eins ist klar: Vor zwanzig Jahren hat es nicht funktioniert.

Neben dieser geschichtlichen und organisatorischen „Revolution“ passiert aber auch inhaltlich viel. Während das W3C verschiedene Webtechniken in Spezifikationen standardisiert hat, führt die WHATWG-getriebene HTML5-Entwicklung mehrere Dinge in einem Text zusammen. So finden sich in HTML5 sowohl die Beschreibungen des HTML-Markup (Elemente und Attribute) als auch Details von JavaScript-APIs, beispielsweise des DOM, und von CSS. Darüber hinaus gibt es eine Reihe von Co-Standards, sodass man HTML5 als Standardfamilie betrachten kann.

Neuigkeiten von der Auszeichnungssprache

Wenn man in Zukunft von HTML spricht, muss man dank Version 5 präzisieren, was gemeint ist. Markup und JavaScript-API sind nicht mehr strikt voneinander getrennt. In der Wahrnehmung der Öffentlichkeit scheint CSS3 ebenfalls dazugehören. Wenngleich das formal nicht richtig ist, nimmt CSS3 im Zuge der aktuellen Browserentwicklung Fahrt auf. Insofern kann man unter dem Schlagwort HTML5 eine breite Erneuerung der clientseitigen Webtechniken verstehen. Diese Sichtweise liegt der weiteren Betrachtung zugrunde. Es dreht sich um verschiedene Techniken.

Beim HTML-Markup geht es darum, welche neuen Elemente und Attribute, welche veränderten Bedeutungen der neue Standard bringt. Herausragende Neuerungen sind

- Elementtypen zur Strukturierung der Webseite, wenngleich zurzeit nicht in jedem Fall absehbar ist, was der Browser daraus machen soll,



Eine typische HTML5-Struktur für ein Blog: **header** und **footer** für Kopf- und Fußbereich, **article** für einzelne Beiträge und **aside** für die Seitenleiste (Abb. 1)

- Markup für Audio und Video (S. 32), in Verbindung mit (hoffentlich in Zukunft) flächendeckend unterstützten Codecs,
 - umfassende Erneuerung im Bereich Grafiken in Webseiten, angefangen bei neuen Elementtypen für Abbildungen, über Vektorgrafiken mit eingebettetem SVG, bis zu skriptbaren 2D- und 3D-Grafiken mit dem *canvas*-Element (S. 132 und 135),
 - Modernisierung der Formulare (S. 22),
 - eine Reihe weiterer Elemente und Attribute.
- Hinsichtlich JavaScript-APIs kann dieser Artikel nicht auf alle Änderungen ins Detail gehen. Einige Themen hat iX schon behandelt, andere sollen folgen. Dazu gehören:
- Webworkers (S. 50), eine Technik für Parallelverarbeitung in JavaScript,
 - clientseitige Datenbanken (S. 41), die Speichermöglichkeiten bieten, die weit über Cookies hinausgehen (eine

Ausprägung davon war schon im Rahmen des „Apps mit Webtechniken“-Tutorials ein Thema, S. 166),

- dokumentübergreifende Nachrichten (cross-document messaging)
- Websockets (S. 65), ein Kommunikationskanal zwischen Webseite und -server,
- Geolocation, eine Möglichkeit, per JavaScript die geografische Position des Benutzers herauszufinden, was mit mobilen Geräten gut und mit stationären Geräten nicht schlecht funktioniert (S. 166).

In der öffentlichen Wahrnehmung gehört CSS3 zu der gerade stattfindenden Runderneuerung der Webtechniken. Neben vielen kleinen Innovationen gibt es zwei, die auffallen: Webfonts und HTML5 auf mobilen Geräten. Dabei spielen Cascading Style Sheets eine zentrale Rolle.

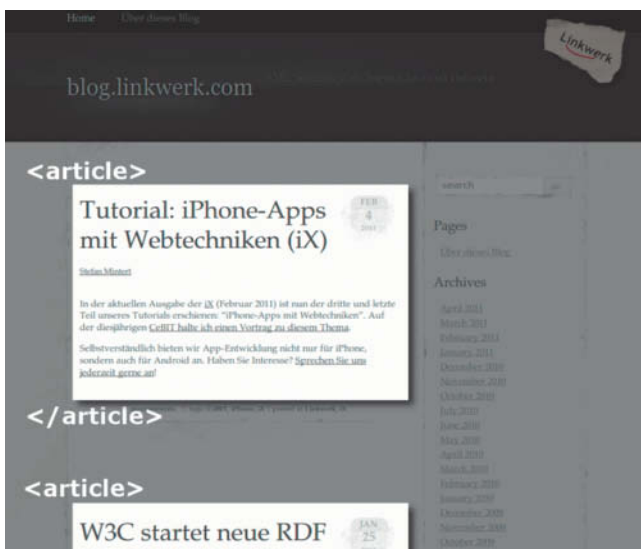
Webfonts lassen Hoffnung aufkommen, dass man endlich beliebige Schriften auf einfache und browserübergreifende Weise im Web benutzen kann (S. 83), und Medienselektoren erlauben zwischen Desktop- und mobilen Browsern zu unterscheiden (S. 78).

Neu strukturierte Webseiten

Die grundlegende Struktur einer Webseite hat sich nicht geändert. Geblieben sind die Elementtypen *html*, *head*, *body*. Zur Strukturierung des Inhalts gibt es ebenfalls weiterhin die bekannten Überschriften *h1* bis *h6*.

Listing 1: Strukturierung mit *article*

```
<body>
<article>
  <h1>Tutorial: iPhone-Apps mit Webtechniken (iX)</h1>
  <p class="author">Stefan Mintert</p>
  <p>In der aktuellen Ausgabe...</p>
  <p>Selbstverständlich bieten...</p>
</article>
<article>
  <h1>W3C startet neue RDF Working Group</h1>
  ....
</article>
</body>
```



Ein Beitrag in einem Blog bietet sich für die Auszeichnung mit dem Element *article* an (weiß hervorgehoben). Am unteren Rand folgt der nächste *article*. Die sichtbaren Tags symbolisieren den zugrunde liegenden Quellcode (Abb. 2).

Neu sind Gliederungselemente, die beim Blick auf ihre Bezeichnungen vertraut wirken. Wer sich die Details ansieht, muss aber mitunter starke Nerven mitbringen. Doch so schlimm muss es ja nicht sofort werden. Zur Orientierung genügt die Vogelperspektive. Die auffälligsten Elemente sind *header*, *footer*, *nav*, *section*, *article* und *aside*. In allen Fällen spricht der Spezifikationsentwurf von semantischen Elementtypen. Die Namen der Elemente sind sicher unmittelbar verständlich, allerdings lässt der angehende Standard erstaunlich viel Interpretationsspielraum, verzichtet weitgehend auf Erklärungen, was der Browser oder ein anderer „User Agent“ daraus macht, und hält last, but not least Überraschungen bereit. Oder wer hätte gedacht, dass der *footer* zweimal vorkommen oder zu Beginn der Seite auftreten kann?

Blogs motivieren Markup: Seit Ende der 90er-Jahre ist zwar inhaltlich bei HTML nichts passiert, im Web jedoch viel. Ein seit dieser Zeit aufgetretenes Phänomen sind Blogs. Es hat den Anschein, dass die beim Entwurf einiger der genannten Elemente Pate standen. Blogsites ähneln sich oft in ihrem Seitenlayout. Abbildung 1 zeigt ein typisches Blog, hier auf Grundlage von WordPress und einem üblichen „Theme“.

Eine zweite verbreitete Struktur sind Texte aus ineinander verschachtelten Gliederungseinheiten. Bücher sind ein Paradebeispiel, obwohl ein Buch normalerweise auf einem altmodischen Device namens „Papier“ ausgegeben wird (allerdings fußt die EPUB-Spezifikation 3 auf HTML5) und daher mit HTML meist wenig zu tun hat. Eine Einteilung in Kapitel, Abschnitte und Unterabschnitte kennt trotzdem jeder.

Wer nun auf die Idee kommt, dass ein Blog-Artikel aus Abschnitten bestehen kann und diese eigenständige Artikel erhalten können: WHATWG/W3C sind genau dieser Meinung. In der Tat ist eine Verschachtelung von *section* und *article* zulässig. Für den Überblick sollte das genügen, Zeit für die Details.

Gut für Blogs: Eigenständige Texte auf einer Webseite

Das Element *article* enthält einen Teil der Webseite, der alleinstehend sinnvoll ist. Blogs sind ein gutes Einsatzgebiet für dieses Element, weil Blog-Beiträge unabhängig von anderen Beiträgen sind.

Diese Unabhängigkeit bedeutet, dass man einen Beitrag aus dem Kontext des Blogs entfernen kann, ohne dass der Text seinen Sinn verliert. Bei der Syndizierung von Nachrichten ist das eine wesentliche Eigenschaft. Blog-Aggregatoren führen eine solche Aufgabe aus. Sie beobachten Blogs, kopieren die Inhalte und verbreiten sie unabhängig vom Ursprungs-Blog. Das *article*-Element erlaubt ihnen, zuverlässig zu erkennen, wo ein Beitrag beginnt und endet.

Abbildung 2 zeigt eine Blog-Übersichtsseite. Der hervorgehobene Beitrag ist auf der Übersichtsseite vollständig zu lesen. Alle anderen Texte, die darüber, daneben oder darunter zu sehen sind, haben mit dem Beitrag nichts zu tun. Listing 1 zeigt den HTML5-Code.

In einem Blog ist es üblich, Beiträge auf der Übersichtsseite nur anzureißen und mit „Mehr...“ oder ähnlichen Links auf den Volltext zu verweisen. Dort ergibt das *article*-Element wieder Sinn.

Exkurs: Das Element *time* ist kein Strukturelement, gehört deshalb eigentlich nicht hierher. Es hat aber seine Berechtigung im Zusammenhang mit dem vorgestellten *article* und zeichnet die Angabe eines Datums oder Zeitpunkts aus. Mit dem optionalen Attribut *pubdate* kann man es in einem *article* nutzen, um dessen Veröffentlichungsdatum anzugeben:

```
<time pubdate="pubdate"
datetime="2011-02-04T14:28:00"></time>
```

Gliederungsabschnitte und ihre Überschriften

Mit *div* steht schon lange ein Element zur Verfügung, das nahezu beliebige Teile einer Webseite als zusammengehörig kennzeichnen kann, aber *div* hat keine weitere Bedeutung. HTML5 führt deswegen das Element *section* ein, das etwas wie ein Kapitel in einem Buch markiert, und die Überschrift der *section* steht im Inhaltsverzeichnis.

Wer einen Blick auf die aktuelle Wikipediaseite zu HTML5 wirft und sich die Abschnitte genauer ansieht, dürfte feststellen: Auf Code-Ebene könnte die Seite in Zukunft aussehen wie Listing 2.

Das Beispiel zeigt, dass man *section*-Elemente ineinander verschachteln kann. Des Weiteren sind einige der bekannten Überschriften *h1* bis *h6* zu sehen. Dazu sind noch erklärende Worte notwendig.

Die aus früheren Versionen von HTML bekannten Überschriften sind auch in Version 5 enthalten. Für die weiteren Erörterungen ist die Nummer im Elementnamen wichtig. Sie bezeichnet den Rang der Überschrift. Bei *h1* spricht man vom höchsten Rang, bei *h6* vom niedrigsten.

Aus zwei Überschriften mach eine

Man kann die Überschriften wie bisher als direkte Kind-elemente von *body* verwenden. Allerdings erzeugen sie im Browser implizit korrespondierende Abschnitte. Was es damit genau auf sich hat, folgt unten.

Manchmal bilden mehrere Elemente eine Überschrift. Bei Wikipedia lautet sie zum Beispiel „Wikipedia – Die freie Enzyklopädie“. Was hier durch einen Gedankenstrich gekennzeichnet ist, könnte in HTML wie folgt aussehen.

```
<h1>Wikipedia</h1>
<h2>Die freie Enzyklopädie</h2>
```

Dass es sich bei diesen beiden Überschriftselementen von der Bedeutung her um nur eine Überschrift – Titel und Untertitel – handelt, ist dem Markup nicht zu entnehmen. Mit Version 5 gehört das der Vergangenheit an, denn die Zusammengehörigkeit drückt das neue Element *hgroup* aus. Damit wandelt sich der obige Code zu

```
<hgroup>
<h1>Wikipedia</h1>
<h2>Die freie Enzyklopädie</h2>
</hgroup>
```

Die Überschrift besteht nun aus dem Inhalt von *hgroup*. Die HTML5-Spezifikation verlangt, dass in einem solchen Fall in einem Inhaltsverzeichnis der Inhalt des ranghöchsten *h*-Elements erscheint. Hier demnach „Wikipedia“.

Das Zusammenspiel von Überschriften und Abschnitten in HTML5 ist nicht leicht zu verstehen. Wie oben erwähnt, kommt den Überschriften nun eine Bedeutung zu. Sie markieren den Anfang von Abschnitten; deren Ende soll der Browser implizit erkennen. Man kann einwenden: „Was interessiert mich die Bedeutung der Elemente, das hat noch nie eine Rolle gespielt.“ Das ist bestenfalls solange richtig, wie sich die Browser an die Vorgaben der Spezifikation halten. Und die hat es in sich. So zeigt der aktuelle Entwurf eine Struktur wie in Listing 3.

Die Gliederung, die der Browser daraus ermitteln soll, nimmt die folgende Gestalt an (die Nummerierung dient ausschließlich der Veranschaulichung):

1. Ein Versuch in HTML
- 1.1 Über implizite Abschnitte
- 1.2 Abschnitte an und für sich
- 1.3 Explizite Abschnitte

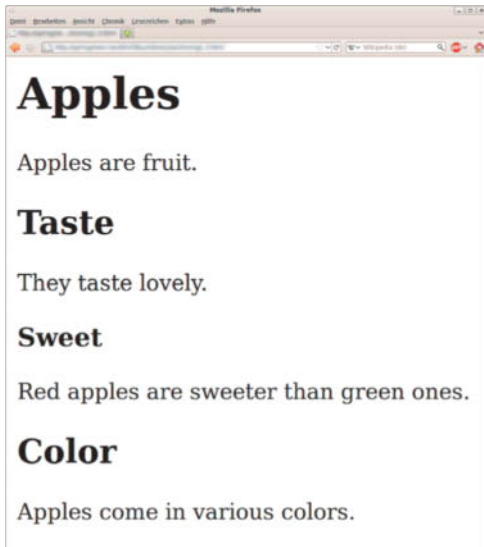
Leser, die das intuitiv finden, dürfen gerne einige Zeilen überspringen. Allen anderen sei die überraschende Denkweise nahegebracht, die sich dahinter verbirgt. Zunächst kennt HTML5 sogenannten gliedernden Inhalt („sectioning content“). Es handelt sich um vier Elementtypen, von denen zwei schon vorgestellt wurden: *article*, *aside*, *nav* und *section*. Ihnen ist gemein, dass sie jeweils einen Geltungsbereich für Überschriften definieren. Um das zu erklären, sei aus dem obigen Code ein Ausschnitt gewählt.

Listing 2: Verschachtelte *section*-Elemente

```
<body>
<h1>HTML5</h1>
<p>HTML5 ist eine textbasierte Auszeichnungssprache zur Strukturierung und ...</p>
<section>
  <h2>Entstehung</h2>
  <p>Die von mehreren Browser-Herstellern unterstützte WHATWG veröffentlichte...</p>
</section>
  <h3>Verschiedene Arbeitsmodelle von W3C und WHATWG</h3>
  <p>Die WHATWG verfolgt ein versionsloses Modell der Entwicklung.</p>
</section>
  <h3>Verhältnis der Spezifikationen des W3C und der WHATWG</h3>
  <p>Der Verfasser (engl. editor) der Spezifikation ist...</p>
</section>
  <h3>W3C-Veröffentlichungen</h3>
  <p>Im Folgenden die Veröffentlichungen der HTML5-Entwürfe durch das W3C...</p>
</section>
  <h3>Fortschritt in der Entwicklung</h3>
  <p>In der Spezifikation der WHATWG wird darauf hingewiesen,...</p>
</section>
  <h2>Ziele von HTML5</h2>
  <p>Die ersten wichtigen Ziele für HTML5 wurden von Tim Berners-Lee...</p>
</section>
...
</body>
```

Listing 3: Struktur und *h*-Elemente

```
<body>
<h1>Ein Versuch in HTML</h1>
<h2>Über implizite Abschnitte</h2>
<blockquote>
  <h3>Keep it simple</h3>
</blockquote>
<p>"Implizit" ist das Gegenteil von "Übersichtlich" ;-></p>
<h2>Abschnitte an und für sich</h2>
<section>
  <h3>Explizite Abschnitte</h3>
</section>
<p>Nur nicht den Überblick verlieren.</p>
</body>
```



Alle Überschriften sind vom Typ *h1*. Unterschiedliche Schriftgrößen sind die Folge der *section*-Verschachtelung (Abb. 3).

```
<section>
<h3>Explizite Abschnitte</h3>
</section>
```

Das *h3*-Element stellt die erste Überschrift in der *section* dar. Damit ist sie die Überschrift des Abschnitts. Sie ist gleichwertig zu einer *h1*-Überschrift an derselben Stelle. Folgen weitere *h*-Elemente, erzeugen sie implizite Abschnitte oder Unterabschnitte, je nach ihrem Rang. Das vollständige Listing der Reihe nach:

- Das Element mit dem Text „Ein Versuch in HTML“ ist die Überschrift der expliziten Body-Section.
- Es folgt eine Überschrift mit niedrigerem Rang und dem Inhalt „Über implizite Abschnitte“. Aufgrund des niedrigeren Rangs bildet sie einen Unterabschnitt, ein *section*-Element innerhalb des Body.
- *blockquote* gilt als sogenannte Gliederungswurzel („sectioning root“). Das bedeutet, dass ihre Abschnitte und deren Überschriften als Teil einer eigenen Gliederungshierarchie anzusehen sind. Der Inhalt von *blockquote* fällt deshalb aus der weiteren Betrachtung heraus.
- Der Absatz mit dem Text „,Implizit‘ ist das Gegenteil von ‚übersichtlich‘ ;-“ gehört noch zum selben Abschnitt wie die *blockquote*.
- Direkt danach beginnt mit „Abschnitte an und für sich“ eine neue implizite *section*. Es folgt unmittelbar darauf die letzte, diesmal explizite.

Zwei weitere Dinge sind hier bemerkenswert:

- Der Absatz „Nur nicht den Überblick verlieren“ ist Teil des umgebenden Abschnitts mit der Überschrift „Ein Versuch in HTML“.
- Die Überschriften „Abschnitte an und für sich“ und „Explizite Abschnitte“ befinden sich auf der gleichen Ebene. Ihr unterschiedlicher Rang spielt keine Rolle, weil sie sich in verschiedenen (impliziten und expliziten) Abschnitten befinden.

Abschnitte inkonsistent und dennoch korrekt

Diese durchaus verwirrende Betrachtung mag außerdem theoretisch wirken. Ob Letzteres zutrifft, wird von den Browsern abhängen. Da schon erste Implementierungen

existieren, wird das nächste Listing zeigen, dass man sich mit dem Thema auseinandersetzen muss. Es handelt sich um ein weiteres Zitat aus dem aktuellen Entwurf von HTML5 und besteht aus drei Listings, die für einen Browser dieselbe Bedeutung haben sollen. Den Anfang macht Code, in dem Abschnitte inkonsistent aber korrekt zum Einsatz kommen.

```
<h4>Apples</h4>
<p>Apples are fruit.</p>
<section>
<h2>Taste</h2>
<p>They taste lovely.</p>
<h6>Sweet</h6>
<p>Red apples are sweeter than green ones.</p>
<h1>Color</h1>
<p>Apples come in various colors.</p>
</section>
```

Wer es übersichtlicher haben möchte, schreibt:

```
<h1>Apples</h1>
<p>Apples are fruit.</p>
<section>
<h2>Taste</h2>
<p>They taste lovely.</p>
<section>
<h3>Sweet</h3>
<p>Red apples are sweeter than green ones.</p>
</section>
<section>
<h2>Color</h2>
<p>Apples come in various colors.</p>
</section>
```

Auf den Rang der Überschrift kommt es nicht an

In jeder *section* ist nun genau eine Überschrift enthalten. Wie oben beschrieben, kommt es in so einem Fall nicht mehr auf den Rang einer Überschrift an. Er spielt nur eine Rolle beim „Aufspüren“ von impliziten Abschnitten. Deshalb kann man das Ganze noch vereinfachen.

```
<h1>Apples</h1>
<p>Apples are fruit.</p>
<section>
<h1>Taste</h1>
<p>They taste lovely.</p>
<section>
<h1>Sweet</h1>
<p>Red apples are sweeter than green ones.</p>
</section>
<section>
<h1>Color</h1>
<p>Apples come in various colors.</p>
</section>
```

Listing 4: header und implizite section

```
<body>
<header>
<h1>Homepage von Donald Duck</h1>
<h2>Mein Leben in Bildern</h2>
</header>
<p>Willkommen auf meiner Seite...</p>
<p>...</p>
<h2>Wie alles anfing</h2>
<p>Fauntleroy hätte ich heißen sollen, wenn sich mein Vater durchgesetzt hätte. ...</p>
</body>
```


Dieses Listing erscheint im Firefox ab 4.01 wie in Abbildung 3. Alle Überschriften sind vom Typ *h1*. Unterschiedliche Schriftgrößen sind die Folge der *section*-Verschachtelung.

Man beachte die unterschiedlichen Schriftgrößen, obwohl alle Überschriften mit *h1* ausgezeichnet sind. Aktuelle Versionen von Safari und Chrome unter Windows zeigen die Überschriften alle gleich groß an. Man darf gespannt sein, ob sich die Browser ändern oder die Spezifikation.

Der eine oder andere Leser könnte hier auf dieselbe Idee kommen wie der Autor: Weshalb ersetzt man nicht die sechs verschiedenen Überschriften durch eine einzige und motiviert damit den sauberen Aufbau mit verschachtelten Abschnitten? Eine Antwort können nur die Urheber des kommenden Standards geben. Die Initiatoren der WHATWG nennen in ihren FAQ als Grund für die Initiative ausdrücklich die mangelnde Berücksichtigung der Bedürfnisse von Autoren. Mit dem Überschrifts-/Abschnittsdurcheinander haben sie Schreibern von HTML keinen Dienst erwiesen.

Ein *header* stellt keinen gliedernden Inhalt dar

Nach diesem langen Exkurs zu Abschnitten und Überschriften sollen die anderen Strukturierungselemente nur noch kurz zu Wort kommen. Sie scheinen ja schon ganz vertraut.

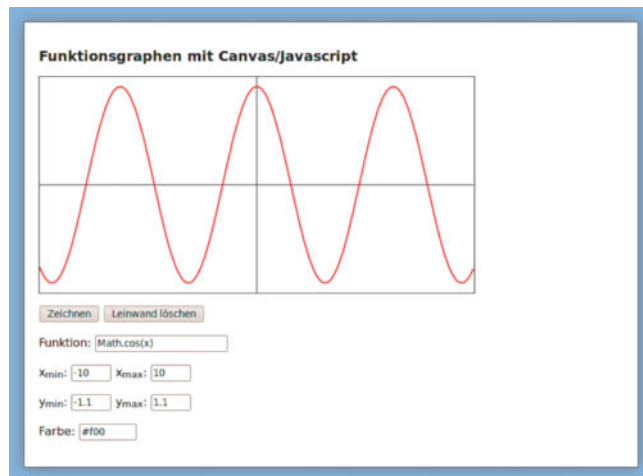
header und *footer* sollen den Kopf- und Fußbereich einer Webseite auszeichnen, ohne dass eine bestimmte Darstellung damit verbunden wäre. Bots, die eine Seite lesen, haben es damit unter Umständen leichter, wichtigen von unwichtigem Inhalt zu unterscheiden. *header* stellt im Gegensatz zu *section* keinen gliedernden Inhalt dar. Was das bedeutet, lässt sich am besten am Quellcode diskutieren (siehe Listing 4).

h1 ist die Überschrift des *body*. Die darauffolgende *h2* erzeugt implizit eine *section*. Da der Header nun keine gliedernde Wirkung hat, befinden sich die beiden Absätze, die hinter dem Kopfbereich stehen, in der erzeugten *section*. Versucht man, diesen Sachverhalt durch Markup hervorzuheben, führt das zu Code wie Listing 5.

Zu diesem überraschenden Ergebnis führt die wörtliche Auslegung des aktuellen Standard-Entwurfs. Dass es sich dabei keinesfalls um eine korrekte Auszeichnung handelt, sollte klar sein. Selbst nach den ungewöhnlichen Ideen der HTML5-Macher ist diese Verschachtelung unzulässig. Da es sich um implizites Markup handelt, wird niemand gezwungen, die Tags wirklich in die Seite zu schreiben. Spannend bleibt aber die Frage, was im Browser passiert und wie das Document Object Model aussieht. Die Spezifikation gibt da-

Listing 5: Gedachte schlechte Elementstruktur

```
<body>
<header>
<h1>Homepage von Donald Duck</h1>
<section>
<h2>Mein Leben in Bildern</h2>
</header>
<p>Willkommen auf meiner Seite...</p>
<p>...</p>
</section>
<section>
<h2>Wie alles anfing</h2>
<p>Fauntleroy hätte ich heißen sollen, wenn sich mein
Vater durchgesetzt hätte. ...</p>
</section>
</body>
```



Ein von JavaScript auf einen Canvas gezeichneter Funktionsgraph (Abb. 4)

rauf keine Antwort und den Browsern sieht man die Antwort nicht an.

Die beiden letzten neuen Elemente zur Seitenstruktur heißen *aside* und *nav*. Ersteres dient der Auszeichnung von „marginalem“ Inhalt, etwa einer Sidebar. Das Element *nav* soll der Container für die Hauptnavigation einer Seite sein.

Um Grafiken soll es nun gehen, jedoch ist das erste vorzustellende Element viel allgemeiner. Es heißt *figure* und nimmt nicht nur Grafiken, Zeichnungen, Fotos und derartiges auf, sondern mehr. *figure* enthält Dinge, die eine logische Einheit innerhalb des Fließtextes bilden und aus diesem heraus referenziert werden. Im Text heißt es dann häufig „siehe Abbildung 5“.

Die bisherige Möglichkeit, Bilder in Webseiten einzubauen, ist recht ausdruckschwach. Man nehme ein *img*-Element, reiche es noch um ein *alt*-Attribut an und fertig. Wer dem Bild noch eine Überschrift mitgeben wollte, ummantelte es mit einem Absatz (*p*). Mit dem neuen Element und dem zugehörigen *figcaption* sieht das strukturierter aus:

```
<figure>
<br/>
<figcaption>Blauer Himmel, toller Strand -- Urlaub in
Lummerland</figcaption>
</figure>
```

Zeichnungen in HTML: Canvas in Aktion

Das *canvas*-Element ist eines der Highlights der neuen HTML-Entwicklung. Es ist schon seit einigen Jahren bekannt (siehe S. 132 und 135), bekommt aber erst jetzt die große Aufmerksamkeit. Die Gründe sind klar: Berücksichtigung im angehenden Standard und breite Unterstützung durch die Browser. „Canvas“ bedeutet „Leinwand“ und auf ihr kann man mit JavaScript zeichnen. HTML-seitig ist die Angelegenheit recht einfach. Man erzeugt einen Canvas und gibt ihm die gewünschte Größe. Für den einfachen Zugriff per JavaScript empfiehlt sich eine ID.

```
<canvas id="plotter" width="600"
height="300"></canvas>
```

Wer Rücksicht auf Browser nehmen möchte, die *canvas* nicht kennen, kann einen Ersatztext in das Element schreiben. Ohne Angabe der Maße besitzt ein Canvas eine Breite von 300 und eine Höhe von 150 Pixeln. Für das Styling steht wie üblich CSS zur Verfügung. Folgende Anweisungen haben nur bei obigem Canvas dieselbe Wirkung. Die erste wirkt auf alle gleichnamigen Elemente.

```
canvas { border: 1px solid #000; }
#plotter { border: 1px solid #000; }
```

So weit, so gut, aber was kann man mit dieser Leinwand anfangen? Der Nutzen entsteht erst durch JavaScript. Zur Veranschaulichung soll ein in eine Webseite integrierter Funktionsplotter dienen. Um den zu programmieren, genügen wenige Kenntnisse über die Canvas-API. Die erste bemerkenswerte Eigenschaft ist die Tatsache, dass es nicht nur eine API gibt, sondern momentan zwei: eine für 2D-Grafiken und eine für 3D. Der Standard spricht hier von einem sogenannten Kontext. Eine Übersicht über die etablierten Kontexte für einen HTML-Canvas findet sich bei der WHATWG. Sie tragen derzeit die Bezeichner „2d“ und „webgl“. Für Ersteres ist eine Spezifikation bei W3C zu finden. Die Spezifikation für WebGL steht bei der Khronos Group zur Verfügung; siehe dazu das WebGL-Tutorial ab Seite 140.

Für den Funktionsplotter genügt ein 2D-Kontext. Zugriff auf die Leinwand und den Kontext erhält man auf folgende Weise.

```
plotterCanvas = document.getElementById("plotter");
ctx = plotterCanvas.getContext('2d');
```

Eigenschaften des *canvas* fragt man auf übliche Weise ab.

```
w = plotterCanvas.width;
h = plotterCanvas.height;
```

Für das Zeichnen eines Funktionsgraphen sind folgende Methoden und Eigenschaften des Kontextes erforderlich:

```
ctx.strokeStyle = "#000"; // Zeichenfarbe
ctx.lineWidth = 1; // Linienbreite
ctx.beginPath(); // Neue Linie beginnen
ctx.moveTo(x, y); // Zur Position (x,y) springen
ctx.lineTo(x, y); // Linie zu (x,y) ziehen
ctx.stroke(); // Zeichnen auslösen
ctx.closePath(); // Linie abschließen
```

Jeder, der schon einmal grundlegende Zeichenfunktionen in irgendeiner Sprache benutzt hat, dürfte die Methoden verstehen; bemerkenswert ist allenfalls die Methode *stroke()*. Mit diesen Zutaten lässt sich der Funktionsplotter bauen. Er besteht aus zwei Dateien, der HTML- und einer ausgelagerten JavaScript-Datei. Im Browser kommt der Plotter unscheinbar daher, schließlich gibt es fast keine CSS-Anweisungen, die ihn optisch aufwerten.

Die Formularfelder erlauben dem Benutzer die Eingabe der Funktion und der Intervallgrenzen. Außerdem kann er die Zeichenfarbe für den Funktionsgraphen bestimmen. Ein Klick auf „Zeichnen“ erzeugt die Darstellung in Abbildung 4. Wenn man die Leinwand nicht löscht, kann man mehrere Funktionen übereinanderlegen.

Listing 6 enthält die HTML-Datei. Vorsicht: Sie enthält einen „Fehler“, der das Zusammenspiel mit JavaScript betrifft. Die Beschreibung dieses zunächst nicht sichtbaren Fehlers sowie dessen Auflösung folgt unten. Die Webseite ist einfach aufgebaut. Im *head* sorgt eine CSS-Anweisung dafür, dass der Canvas einen Rahmen bekommt, und es folgt die Einbindung des Skripts aus der Datei *plotter.js* (Auszug in Listing 7).

Eingabe einer mathematischen Funktion

Im *body* steht der Canvas mit der ID „plotter“, Breiten- und Höhenangaben und einem Ersatztext. Das sich anschließende Formular erlaubt dem Benutzer die Eingabe einer mathematischen Funktion mit dem formalen Parameter *x*. Der

Listing 6: Plotter (HTML)

```
<!DOCTYPE html>
<html>
  <head>
    <title>Funktionsplotter mit Canvas | HTML5</title>
    <link href="plotter.css" rel="stylesheet" type="text/css"/>
    <script type="text/javascript" src="plotter.js"></script>
    <meta http-equiv="Content-Type" content="text/html; charset=utf-8"/>
  </head>
  <body>
    <section>
      <h1>Funktionsgraphen mit Canvas/JavaScript</h1>
      <form action="" onsubmit="return false;">

        <canvas id="plotter" width="600" height="300">
          Dein Browser unterstützt canvas leider nicht.
        </canvas>

        <p><input type="button" value="Zeichnen"
          onclick="Plotter.draw(this.form.function.value, Number(this.form.xmin.value),
            Number(this.form.xmax.value), Number(this.form.ymin.value),
            Number(this.form.ymax.value))" />
          <input type="button" value="Leinwand löschen" onclick="Plotter.clear()" />
        </p>
        <p>Funktion: <input id="function" type="text" size="20" value="Math.cos(x)" /></p>
        <p>
          x<sub>min</sub></p> <input id="xmin" type="text" size="4" value="-40" />
          x<sub>max</sub></p> <input id="xmax" type="text" size="4" value="40" />
        </p>
        <p>
          y<sub>min</sub></p> <input id="ymin" type="text" size="4" value="-1.5" />
          y<sub>max</sub></p> <input id="ymax" type="text" size="4" value="1.5" />
        </p>
        <p>Farbe: <input id="color" type="text" size="7" value="#00f" /></p>
      </form>
    </section>
    <pre id="log"></pre>
  </body>
</html>
```

Listing 7: plotter.js

```
function log(text) {
  var logElem = document.getElementById("log");
  logElem.innerHTML += text + "\n";
  return true;
}

var Plotter = (function () {
  /* Objekte für Canvas und Context */
  var plotterCanvas = document.getElementById("plotter"),
      ctx = plotterCanvas.getContext('2d'),
      w = plotterCanvas.width,
      h = plotterCanvas.height;

  /* Methode zum Löschen des Canvas */
  var clear = function () {
    plotterCanvas.width = plotterCanvas.width;
  };

  var draw = function (functionString, xMin, xMax, yMin, yMax) {
  };

  return {
    draw: draw,
    clear: clear
  };
})();
```

Listing 8: Eigene Attribute per *data-*einbauen

```
<div class="spaceship" data-ship-id="92432"
  data-weapons="Laser 2" data-shields="50%"
  data-x="30" data-y="10" data-z="90">
  <button class="fire"
    onclick="spaceships[this.parentNode.dataset.shipId].fire()">
    Fire
  </button>
</div>
```