1 Die Webplattform

In diesem Kapitel spreche ich die Webtechnologien an, die Sie in diesem Buch kennenlernen werden. Ich gehe auf das notwendige Hintergrundwissen ein, damit Sie das Maximum aus Ihrer Lektüre herausholen können, und ich beschreibe die Anforderungen und Voraussetzungen für die Multi-Device-Entwicklung. In diesem Kapitel möchte ich sozusagen sicherstellen, dass wir uns auch im übertragenen Sinne auf derselben Seite befinden, bevor wir uns in den nächsten Kapiteln wirklich den technischen Details widmen.

Wenn Sie sofort weiterlernen möchten, würden Sie vielleicht am liebsten gleich zu Kapitel 2 blättern. Ich bitte Sie aber, dies nicht zu tun, denn das vorliegende Kapitel 1 enthält einiges an sehr interessantem und nützlichem Hintergrundwissen. Der wenig technische Stoff erlaubt mir zudem, die besten Beispiele meines wundervollen Humors zu präsentieren.

1.1 Ein kurzer Hinweis zu den Begriffen

Websites

Im Buch spreche ich immer wieder von der Website- oder Site-Programmierung. Diese Begriffe sind zwar eine Vereinfachung, helfen mir aber, Wiederholungen zu vermeiden. Das in diesem Buch Gelernte lässt sich auf Websites, Webanwendungen und hybride Apps für Mobilgeräte anwenden, kurz gesagt: auf alles, was HTML, CSS und JavaScript verwendet. Weil das aber ein richtiger Rattenschwanz ist, spreche ich meist von »Websites« – es sei denn, ich muss es präzisieren.

Browser und User Agents

Ich spreche auch wahlweise von »Browsern« oder »User Agents«, wenn ich eine beliebige Softwareinstanz meine, die Websites oder Applikationen wiedergeben kann. Auch hiermit möchte ich ganz einfach Wiederholungen vermeiden.

1.2 Wer Sie sind und was Sie wissen müssen

Lassen Sie mich vorab klären, welche Annahmen ich über Sie treffe. Außerdem möchte ich Ihnen mitteilen, welches Hintergrundwissen Sie benötigen, um den bestmöglichen Nutzen aus diesem Buch zu ziehen.

HTML, CSS und JavaScript

Sprechen wir zuerst von Ihnen. Egal ob Sie bereits Profi sind, gerne einer werden wollen oder ob Sie einfach nur Spaß daran haben, mit dem Web herumzuspielen: Sie besitzen bereits brauchbare HTML-, CSS- und JavaScript-Kenntnisse. Diese müssen nicht allzu tief gehen oder besonders detailliert sein, aber doch so gut, dass ich Ihnen nicht beibringen muss, was HTML, CSS und JavaScript sind und wie diese Sprachen geschrieben werden.

Website-Programmierung

Vielleicht haben Sie die Website-Programmierung schon vor einiger Zeit erlernt und müssen Ihre Fähigkeiten auf den neuesten Stand bringen; vielleicht lernen Sie Webentwicklung in der Schule und wünschen sich noch einige Extrastunden. Oder Sie arbeiten bereits als Entwickler, haben aber keine Gelegenheit, die neuesten Trends in der Webprogrammierung zu verfolgen. Wenn eine dieser Beschreibungen auf Sie zutrifft, dann möchten Sie sicher gerne lernen, wie Websites auf moderne Weise programmiert werden, sodass sie auch auf unterschiedlichen Geräten laufen und sich den Fähigkeiten und Abmessungen dieser Geräte anpassen – zweifellos haben Sie deshalb zu diesem Titel gegriffen.

Dieses Buch baut auf Ihren Kenntnissen als Webentwickler auf. Es ist kein Leitfaden für Anfänger, aber auch kein Fortgeschrittenenbuch. Es ist vielmehr eine Momentaufnahme der aktuellen, neuen und in naher Zukunft relevanten Funktionen von HTML, CSS, JavaScript und der mit ihnen verwandten Technologien. Ein Schwerpunkt liegt dabei auf den Techniken, die am besten zur Programmierung von Sites für die vielfältige Gerätewelt geeignet sind.

Entwicklertools

Neben Ihrem Grundwissen müssen Sie sich auch noch mit den Entwicklertools Ihres Browsers auskennen; Sie müssen hier allerdings kein Power-User sein. In einigen JavaScript-Beispielen gebe ich die Ergebnisse in der Entwicklerkonsole eines Tools aus. Dies ist eine Standardmethode, bei der es keine Rolle spielt, ob Sie die integrierten Funktionen von Chrome, Firefox, IE9+, Opera oder Safari oder Drittanbieterlösungen wie Firebug nutzen. Eine meiner Codezeilen könnte etwa so aussehen:

```
console.log('Hello World');
```

Und das Ergebnis wird in der Konsole ausgegeben; Abb. 1–1 zeigt diese Ausgabe in Firebug. Ich verwende die Konsole oder Entwicklertools, wie gesagt, recht selten. Wenn Sie aber nicht damit vertraut sind, sollten

Sie sich nun wirklich etwas Zeit nehmen, um den Umgang mit der Konsole zu lernen.

```
console.log('Hello World!');
Hello World!
```

Abb. 1–1
Eine Hello-World-Nachricht als Konsolenausgabe in Firebug

Wenn Sie immer noch weiterlesen, dann haben Sie entweder die nötigen Kenntnisse oder Sie versuchen, sich durchzumogeln. Wie dem auch sei – nun sprechen wir über die Technologie.

1.3 Begriffsklärung

HTML5

Es herrscht einige Unklarheit darüber, was HTML5 überhaupt genau ist. Da ist einerseits das, was die breite Öffentlichkeit (und ein Großteil unserer Kunden) darunter versteht, andererseits das wirkliche Wesen von HTML5. Es handelt sich um keine brandneue Plattform zur Programmierung von Websites; es ist keine Rich-Media-Umgebung; es ist keine Funktion, die Sie einschalten können, damit Ihre Websites auf unterschiedlichen Geräten laufen. HTML5 ist im Wesentlichen ein Versuch, das Web so weiterzuentwickeln, dass es unseren heutigen Bedürfnissen entspricht. Denn diese Anforderungen haben sich seit der ersten Implementierung von HTML als simples Netzwerk aus verknüpften Dokumenten dramatisch geändert.

Die »Webplattform«

Für die breite Öffentlichkeit ist HTML5 zu einem Oberbegriff für eine Reihe miteinander verwandter und ineinandergreifender Technologien wie etwa CSS3, SVG, JavaScript-APIs und mehr geworden. Auch wenn einige Entwickler sich gerne diese weiter gefasste Bedeutung zu eigen machen, gefällt mir persönlich die Verschmelzung all dieser Technologien nicht besonders gut. Ich nenne HTML5 daher lieber die »Webplattform«. Im Grunde genommen bevorzuge ich sogar den von Bruce Lawson vorgeschlagenen Begriff »New Exciting Web Technologies (NEWT)«. NEWT ist eine coole Abkürzung, und es gibt außerdem ein niedliches Logo dazu. Allerdings hat er sich zugegebenermaßen leider nie durchgesetzt – also sprechen wir von der Webplattform.

Die Webplattform ist riesig. Um sich ihren Umfang zu verdeutlichen, werfen Sie einen Blick auf *http://platform.html5.org*. Hier werden alle Technologien aufgeführt, die zur Plattform gezählt werden. Die Liste ist wirklich eindrucksvoll lang – sie umfasst viel mehr, als ich je in diesem Buch abzuhandeln wage.

Stattdessen konzentriere ich mich auf den Kern, auf diejenigen Technologien, die mir zur Entwicklung von Websites für unterschiedli-

che Geräte ausreichend und hilfreich erscheinen: HTML5, CSS3, SVG, Canvas und einige Geräte-APIs. Ich erkläre sie im Verlauf des Buchs jeweils im Einzelnen. Zunächst möchte ich aber genauer darauf eingehen, was mit HTML5 und CSS3 gemeint ist.

1.4 Das echte HTML5

HTML5 ist eine Weiterentwicklung von HTML 4.01 mit einigen neuen Funktionen, ein paar als veraltet gekennzeichneten oder entfernten Funktionen und einigen bestehenden Merkmalen, deren Funktionsweise geändert wurde. Das Ziel war, die von den Entwicklern jahrelang eingesetzten Hacks und Design-Muster zu standardisieren, um den Ansprüchen des modernen Web gerecht zu werden. Und dabei geht es (mindestens) ebenso sehr um Applikationen wie um Dokumente. Der ursprüngliche Vorschlag für das heutige HTML5 hieß daher auch Web Application 1.0.

Neue Möglichkeiten

Zu den neuen HTML5-Funktionen gehören Möglichkeiten zur Strukturierung von Dokumenten, sodass diese sinnvoller und besser zugänglich werden. Damit beschäftige ich mich in Kapitel 2. HTML5 umfasst zudem eine ganze Palette neuer Möglichkeiten für Formulare und Bedienelemente, mit denen sich Anwendungen leichter entwickeln lassen – diese sehen wir uns in Kapitel 8 an. Und HTML5 beinhaltet auch die Funktion, die viele in erster Linie damit assoziieren: native Videowiedergabe (ohne Plugin). Dies besprechen wir in Kapitel 9.

WHATWG und W3C

Zwei große Gruppen arbeiten an HTML5, und ihre Rollen und Verantwortungsbereiche teilen sich – grob betrachtet – folgendermaßen auf: Das WHATWG (Sie brauchen die Bedeutung dieser Abkürzung nicht zu kennen), ein Konsortium aus Browserherstellern und »interessierten Parteien« erstellt durch den Federführer Ian Hickson eine »Live-Spezifikation« von HTML. Im Grunde ist das eine versionslose Spezifikation, in der ständig neue Funktionen integriert und bestehende aktualisiert werden. Das W3C (World Wide Web Consortium), das Gremium zur Standardisierung des Web, fertigt Momentaufnahmen dieser Spezifikation an und versieht sie mit Versionsnummern. Dabei wird auch die Kompatibilität der Implementierung durch die Browserhersteller sichergestellt.

In Wirklichkeit ist die Situation etwas komplizierter; es gibt diverse politische Verwicklungen. Dies ist jedoch nur für Standardisierungsspezialisten wichtig und hat für Sie keine praktische Bedeutung.

Das W3C hat vorgeschlagen – zum Zeitpunkt, als dieses Buch entstanden ist, jedoch noch nicht abgesegnet –, dass HTML5 (die Momentaufnahme des W3C) als offizielle Empfehlung im Jahr 2014 veröffentlicht werden, also »fertig« sein soll. HTML5.1 soll 2016 folgen. Außerdem soll HTML5 nicht aus einer einzigen monolithischen Spezifikation, sondern aus verschiedenen Einzelmodulen bestehen. Dadurch kann die Arbeit an Einzelaspekten fortgeführt werden, ohne dass sich dadurch alles verzögern würde. Diese Jahresangaben sind für Sie jedoch überhaupt nicht von Bedeutung, Sie müssen nur wissen, wann HTML5 von den Browsern interpretiert und damit von Ihnen genutzt werden kann.

1.4.1 Das HTML5-Gerüst

Wenn Sie elementare HTML-Kenntnisse haben, sind Sie mit den Grundlagen der Seitenauszeichnung vertraut. In HTML5 gibt es jedoch in dieser Hinsicht einige Änderungen – nicht viele, aber sie sollten dennoch erwähnt werden. Der folgende Codeblock zeigt das grundlegende Gerüst, das ich für alle Beispiele in diesem Buch verwende (Sie finden es auch in der Beispieldatei *template.html*):

```
<!DOCTYPE html>
<html lang="de">
<head>
  <meta charset="utf-8">
<title></title>
</head>
<body></body>
</html>
```

Doctype

Dieser Code sollte Ihnen größtenteils vertraut erscheinen. Ich möchte jedoch auf zwei interessante Punkte hinweisen. Der erste ist der *Doctype*. Dieser ist ein Überbleibsel aus der Zeit, als Sie dem Browser mitteilen mussten, welche Art Dokument Sie erstellten – HTML Strict, HTML Transitional, XHTML1.1 und so weiter. In HTML5 ist das nicht mehr notwendig – es gibt nur eine einzige HTML-Ausprägung. Deshalb ist die Doctype-Deklaration überflüssig geworden – jedenfalls in der Theorie. Moderne Browser haben üblicherweise drei Darstellungsmodi: Der *Quirks-Modus* simuliert die nicht-standardgemäße Darstellung des Internet Explorers 5. Dies ist für die Kompatibilität mit veralteten Webseiten notwendig. Der *Standardmodus* ist für moderne, standardkonforme Webseiten gedacht, und der »Beinahe-Standardmodus« (*Almost Standards Mode)* entspricht dem Standardmodus mit ein paar Verhaltensweisen des Quirks-Modus.

Der Browser findet anhand des Doctype heraus, welchen Modus er verwenden soll. Sie sollten stets den Standardmodus verwenden, und der Doctype ist in HTML5 die schnellste Möglichkeit, den Standardmodus auszulösen:

```
<!DOCTYPE html>
```

Das meta-Element

Der zweite wichtige Punkt und die einzige andere Änderung am Standard-HTML5-Gerüst ist das `meta`-Element. Es definiert, welcher Unicode-Zeichenbereich zur Darstellung des Textes auf der Seite verwendet werden soll – im Web ist UTF-8 der Standard. Also sollten Sie ihn in den meisten Fällen verwenden. Das `meta`-Element verwendet das Attribut `charset`:

```
<meta charset="utf-8">
```

Das war es schon. Wenn ein Kunde Sie bittet, seine Website in HTML5 umzuwandeln, können Sie diese beiden Elemente aktualisieren und ihm eine dicke Rechnung stellen. (Bitte tun Sie das nicht – dies war nur ein Scherz.)

Ich könnte viele weitere Optionen nennen. Im Sinne der Verständlichkeit und Einfachheit habe ich sie hier weggelassen. Die bekannte Website *HTML5 Boilerplate* bietet ein umfassendes HTML-Gerüst, das in der Dokumentation ausführlich erläutert wird – denken Sie aber bitte daran, dass es als Ausgangspunkt und nicht als unumstößliches Gesetz betrachtet werden sollte.

1.4.2 Neue Standardverfahren

Außer den Änderungen am Grundgerüst gibt es in HTML5 ein oder zwei neue Standardverfahren, die Sie sich möglicherweise zu eigen machen sollten. Da HTML5 den vielen unterschiedlichen Programmierstilen der Entwickler explizit Rechnung trägt, sollten Sie auch diese Verfahren nicht als unumstößliche Regeln betrachten. Meiner Ansicht nach haben Sie es dadurch jedoch leichter, Ihren Code zu schreiben und zu pflegen.

Das type-Attribut

Zunächst sollten Sie sich daran gewöhnen, dass Sie das `type`-Attribut für die meisten externen Quellen nicht mehr brauchen. In HTML 4.01 oder XHTML mussten Sie für jedes `link`-, `script`- oder `style`-Element einen `type` angeben:

```
<link href="foo.css" rel="stylesheet" type="text/css">
<script src="foo.js" type="text/javascript"></script>
```

Im Web sind CSS und JavaScript jedoch die mit diesem Element verwendeten Quasi-Standard-Ressourcen. Deshalb ist es ein wenig redundant, jedes Mal den `type` anzugeben. Aus diesem Grund können Sie ihn nun weglassen. Ihr Code wird dadurch etwas kürzer, kann von den Browsern aber nach wie vor bestens interpretiert werden:

```
<link href="foo.css" rel="stylesheet">
<script src="foo.js"></script>
```

Nur wenn Sie kein Standard-CSS oder -JavaScript verwenden, müssen Sie den `type` dieser Elemente angeben. Beispielsweise enthalten einige Firefox-Versionen experimentelle Implementationen der neuesten JavaScript-Versionen. Aus Sicherheitsgründen ist es in diesem Fall erforderlich, dass Sie das `type`-Attribut mit einem Flag versehen:

```
<script src="foo.js" type="application/javascript;version=1.8">
</script>
```

Nachsichtig bei der Syntax

Auch was die Syntax angeht, ist HTML5 äußerst nachsichtig. Ob Sie nun gerne alles ganz in Kleinbuchstaben schreiben, Ihre Attribute mit Anführungszeichen versehen oder leere Elemente schließen – HTML5 wird Ihren Code problemlos parsen und verstehen. Die folgenden Codezeilen sind daher gleichwertig:

```
<img src=foo.png>

<img src=foo.png />

<IMG SRC="foo.png"/>
```

Hinweis

Attributwerte mit mehreren Werten (etwa einer Folge von Klassennamen) oder mit bestimmten Sonderzeichen benötigen Anführungszeichen.

Manche Attribute, sogenannte *boolesche Attribute*, können nur wahr oder falsch, genauer gesagt: *true* oder *false* sein. Wenn Sie nichts anderes vorgeben, werden sie als *true* betrachtet. Deshalb müssen Sie keinen Wert angeben – es sei denn, Sie verwenden eine XML-ähnliche Syntax, in der Werte vorgeschrieben sind. In diesem Fall nehmen Sie den Namen des Attributs selbst. Das heißt, dass die beiden folgenden Codezeilen gleichwertig sind:

```
<input type="checkbox" checked>

<input type="checkbox" checked="checked">
```

Ich selbst schreibe gerne alles in Kleinbuchstaben und alles mit Anführungszeichen, und leere Elemente schließe ich nicht:

```
<img src="foo.png">
```

Diesen Stil verwende ich im gesamten Buch, weil ich ihn eleganter finde und weil ich leichter damit arbeiten kann. Mein Texteditor verfügt über eine Syntaxhervorhebung; deshalb kann ich den Code leicht prüfen. Verwenden Sie das von Ihnen bevorzugte System, dieses jedoch konsistent. So lässt sich Ihr Code leichter pflegen.

1.5 CSS3 und mehr

CSS3 verhält sich zu CSS 2.1 wie HTML5 zu HTML 4.01: Es ist ein Evolutionsprozess, der bestimmte vorhandene, aber in den Browsern bisher unterschiedlich implementierte Funktionen standardisiert und außerdem einen komplett neuen Satz Funktionen einbringt. Dadurch ist CSS für eine Welt gerüstet, in der Webbrowser überall eingesetzt werden können.

Präsentation

Die ersten CSS3-Funktionen, die es in die Browser schafften, dienten größtenteils der Präsentation der Seiten und basierten auf Hacks, die die Entwickler schon seit Jahren verwendeten: die Nutzung von Schriften aus jeder beliebigen Quelle, abgerundete Ecken und Schlagschatten für Texte und Rahmen. Es folgten zahlreiche neue Selektoren, um die Auswahl von Elementen sowie dynamische Effekte wie etwa zwei- und dreidimensionale Übergänge und Übergangsanimationen zu vereinfachen. Mehr darüber erfahren Sie bei Bedarf in meinem Buch *The Book of CSS3* (erschienen bei No Starch Press).

Media Queries

Die wirkliche CSS3-Revolution waren jedoch nicht die ganzen glitzernden Effekte, sondern die *Media Queries* – eine Syntax, mit der Sie den Browsern je nach deren Abmessungen und Möglichkeiten unterschiedliche Stildefinitionen bieten können. Dies war der erste Schritt in Richtung einer echten Multi-Device-Gestaltung. Media Queries sowie verschiedene andere CSS-Eigenschaften, die für die Entwicklung responsiver und adaptiver Websites geeignet sind, behandle ich in Kapitel 3.

Layout

Die nächste große Herausforderung für CSS ist das Layoutproblem – die Gestaltung von Layouts, die für den jeweiligen Browser bzw. User Agent wirklich geeignet sind. Dazu gehören Eigenschaften für dynamische Benutzeroberflächen und CSS-gesteuerte Rastersysteme, über die Sie in Kapitel 4 mehr erfahren werden.

CSS3 ist keine einzelne Spezifikation wie CSS 2.1, das in einem einzigen Dokument vollständig beschrieben werden konnte. Dazu ist CSS3

viel zu umfangreich und komplex. CSS3 ist vielmehr modular – eine Reihe kürzerer, spezifischer Spezifikationen, die von den Browsern auf modulare Weise implementiert werden können. Wie bei HTML5 wäre es ziemlich dumm, zu warten, bis CSS3 »fertig« ist, denn manche Module werden lange vor den übrigen fertig und implementiert sein.

CSS3 steht für »alles, was neuer ist als CSS 2.1«

Die CSS-Module werden durchnummeriert, um anzuzeigen, wie viele Entwicklungsschritte sie schon hinter sich haben. Manche sind bereits bei Version bzw. Level 4 – diese sollten gegenüber solchen mit Level 3 möglichst bevorzugt werden. Das heißt jedoch nicht, dass es irgendwann ein CSS4 geben wird – so wird es nicht sein. CSS3 ist das Kürzel für alles, was »neuer ist als CSS 2.1«. Eines Tages wird es diese Unterscheidung nicht mehr geben, sondern einfach nur noch CSS.

1.5.1 Herstellerspezifische Präfixe

Experimentelle Implementierungen

Wenn die Browserhersteller bestimmte Funktionen auf experimentelle Art oder vorläufig implementieren, verwenden sie herstellerspezifische Präfixe, um Kompatibilitätsprobleme mit standardisierten Eigenschaftennamen zu vermeiden. Stellen Sie sich beispielsweise vor, dass ein Menschenaffen-CSS-Modul eine neue Eigenschaft namens `gorilla` einführt und dass sowohl Firefox als auch WebKit diese experimentell, aber leicht unterschiedlich implementieren.

Würden beide denselben Eigenschaftsnamen verwenden, hätte dies in jedem Browser unterschiedliche Auswirkung. Deshalb vermeiden Browserhersteller diesen potenziellen Konflikt durch herstellerspezifische Präfixe, die meist als Vendor-Präfixe bezeichnet werden:

```
-moz-gorilla: foo;
-webkit-gorilla: foo;
```

Vom Prinzip her ist dies ein hervorragendes System; in der Realität ist es jedoch etwas verwirrend. Unter anderem wurden manche mit Präfix versehene Eigenschaften von den Entwicklern so stark genutzt, dass einige Browserhersteller das Bedürfnis hatten, die Vendor-Präfixe ihrer Konkurrenten ebenfalls zu implementieren. Das ist zwar verständlich, macht die ganze Sache jedoch sinnlos.

Die Browserhersteller versuchen inzwischen, die Verwendung der herstellerspezifischen Präfixe einzuschränken. Manchmal wird es aber nahezu unvermeidlich sein, Eigenschaften mit Vendor-Präfix zu verwenden. Mit ein paar Ausnahmen enthalten meine Codebeispiele nur Eigenschaften ohne Präfix. In Anhang A finden Sie einen Hinweis, an welchen Stellen Vendor-Präfixe notwendig sind.

1.5.2 CSS-Frameworks und Präprozessoren

Heutzutage sind Hilfsmittel bei der CSS-Entwicklung geradezu unerlässlich, besonders bei der Arbeit in großen Entwicklerteams und/oder an umfangreichen Projekten. Normalerweise treten diese Helfer in Form von Frameworks oder Präprozessoren oder oft einer Kombination daraus in Erscheinung.

Framework

Ein Framework ist ein Satz vordefinierter CSS-Regeln, die Sie für die schnelle Entwicklung verwenden können; normalerweise beinhalten sie typografische Elemente, Formulare und recht häufig auch Layoutkomponenten. Eines der besseren Frameworks, das in vielen bekannten Websites verwendet wird, ist *Blueprint.css*. Das bekannteste aktuelle Framework ist jedoch *Bootstrap* von Twitter. Es bietet viele vorformatierte Layout-, Typografie- und Formularoptionen, zahlreiche wiederverwendbare Komponenten und auch eine Erweiterbarkeit durch JavaScript.

Präprozessoren

Präprozessoren sind meist serverseitig ausgeführte Programme, die Erweiterungen und Syntaxkürzel in einer CSS-artigen Sprache bieten. Diese werden zur Laufzeit in korrekt formatierte Stylesheets transformiert. Die Erweiterungen enthalten zeitsparende Features wie Variablen und verschachtelte Regeln sowie benutzerdefinierte Funktionen, die dem Anwender fantastische Möglichkeiten bieten. Die Hauptkonkurrenten im Präprozessor-Bereich sind *LESS* und *Sass*. Sass ist der bekanntere der beiden Präprozessoren.

Zwar haben sowohl Frameworks als auch Präprozessoren einen festen Platz in der modernen Webentwicklung, ich beschäftige mich in diesem Buch jedoch nicht damit, da ich Ihnen eher die sprachlichen Grundlagen – auf denen auch die soeben erläuterten Hilfsmittel basieren – nahebringen möchte.

1.6 Browserunterstützung

Unterschiedliche Implementierungen

Nun ist Ihnen sicherlich klar, dass das Multi-Device-Web unglaublich umfang- und variantenreich ist, dass eine immense Vielzahl Browser auf den zahlreichen verschiedenen Geräten läuft und dass es sogar innerhalb dieser Browser eine Fülle von verschiedenen Versionen und Implementierungen gibt (ich hoffe es zumindest, denn im größten Teil der Einleitung habe ich versucht, genau dies zu vermitteln). Aus diesem Grund kann es sehr wohl sein, dass manche in diesem Buch besprochenen Funktionen nicht oder etwas anders implementiert sind.

Statt mich im Text mit den Stufen der Implementierung zu beschäftigen, behandle ich alle neuen Funktionen so, als wären sie vollständig implementiert, und weise in Anhang A auf Ungereimtheiten und Kuriositäten in der realen Welt hin.

Außerdem sind bei brandneuen, innovativen Vorschlägen Änderungen stets vorbehalten – selbst wenn mit manchen Browsern bereits experimentelle Implementierungen ausgeliefert wurden (das in Kapitel 4 beschriebene CSS3-Rasterlayout wurde beispielsweise aktualisiert, während ich an diesem Buch arbeitete). Wenn Sie diese Zeilen lesen, könnte also ein Teil der Syntax, die ich hier verwende, bereits überholt sein. Dieses Risiko muss man eben eingehen, wenn man ein auf Papier gedrucktes Buch über Entwicklungsstandards veröffentlicht. Ich versuche jedoch, es zu minimieren, indem ich auf Funktionen hinweise, die möglicherweise noch geändert werden, und indem ich in der Website zum Buch *http://modernwebbook.com* eine Liste mit Errata und Aktualisierungen pflege.[1]

Nutzen Sie Online-Ressourcen!

Sie sollten auf jeden Fall einige Online-Ressourcen nutzen, um sich über den Fortschritt der Implementierung auf dem Laufenden zu halten. Die meisten davon konzentrieren sich allerdings auf Desktop- und Mobilgerätebrowser. *Can I Use...* zeigt die Unterstützung für zahlreiche aktuelle Technologien und kommende Versionen häufig verwendeter Browser, während sich *HTML5 Please* damit beschäftigt, wie sicher die Nutzung innovativer Funktionen ganz allgemein ist, vor allem von CSS3 und JavaScript (aus diesem Grund finde ich den Namen der Site etwas unpassend).

Die Site *The HTML5 Test* informiert Sie darüber, welche Funktionen der HTML5-Spezifikation Ihr Browser unterstützt, führt aber auch sehr sinnvoll den Umsetzungsgrad in vielen unterschiedlichen Browsern und Geräten auf, einschließlich den Browsern von TV-Geräten und Spielekonsolen. Außerdem ermöglicht sie einen direkten Vergleich von bis zu drei verschiedenen Browsern. Allerdings ist die Site nur auf HTML5-Unterstützung beschränkt.

1.7 Testen, testen und nochmals testen

40-50 Prozent der Entwicklungszeit einplanen

Wenn Sie in der vielfältigen Gerätelandschaft sicherstellen möchten, dass die von Ihnen entwickelte Site überall funktioniert, sind Tests die einzige sichere Möglichkeit. Testen Sie zu Projektbeginn, testen Sie am

1. Anmerkung des Verlags: Bei der vorliegenden Übersetzung haben wir uns zudem bemüht, Änderungen und Neuerungen, die seit dem Druck des englischen Buchs hinzukamen, zu erkennen und in der deutschen Fassung zu berücksichtigen.

Ende, testen Sie bei jeder Gelegenheit und während der gesamten Projektdauer. Wenn Sie ein Projekt für mehrere Geräte planen, sollten den Tests 40 bis 50 Prozent der Entwicklungszeit eingeräumt werden. Wirklich.

Open Device Labs

Tests auf echten Geräten sind durch nichts zu ersetzen. Bauen Sie sich also ein Arsenal mit so vielen Geräten auf wie möglich. Wenn sich in Ihrer Nachbarschaft weitere Agenturen befinden, sollten Sie darüber nachdenken, die Ressourcen zusammenzulegen. In vielen Städten entstehen Open Device Labs – Geräteparks mit zahlreichen, von lokalen Entwicklern und Firmen zur Verfügung gestellten Geräten, die für jedermann genutzt werden können. Suchen Sie online nach dem nächstgelegenen Open Device Lab oder bauen Sie vielleicht selbst eines in Ihrer Firma oder an Ihrem Arbeitsplatz auf. Und beschränken Sie sich dabei nicht auf mobile und Tablet-Geräte; denken Sie vielmehr auch an Spielekonsolen-Browser, wenn Ihre Sites für ein jüngeres Publikum gedacht sind (Untersuchungen haben ergeben, dass in den Vereinigten Staaten etwa jeder vierte Jugendliche mit einem Spielekonsolen-Browser ins Internet geht) oder an TV-Geräte, wenn Ihre Sites für den Freizeitmarkt gedacht sind.

Emulatoren und Simulatoren

Wenn Sie keinen Zugang zu aktuellen Geräten haben, gibt es auch spezielle Tools, und die meisten (oder alle) Benutzeroberflächenentwickler und/oder Gerätehersteller bieten kostenfrei herunterladbare Software-Development-Kits (SDKs) mit Geräteemulatoren.

Im mobilen und Tablet-Bereich gibt es SDKs für Android, Windows Phone und Blackberry und viele andere mehr. Das über den App Store erhältliche Xcode von Apple enthält einen iOS-Simulator, in dem Sie zu Testzwecken zwischen Geräte- und OS-Versionen umschalten können.

Wenn sie einmal eingerichtet sind, geben viele dieser SDKs Ihnen auch die Möglichkeit, physische Geräte per USB anzuschließen, um über den Browser zu debuggen; eine einfachere Möglichkeit ist jedoch der Mobile Emulator von Opera. Sobald er geöffnet und mit einer Desktopversion von Opera verbunden ist, können Sie die Entwickler-Tools auf dem Desktop verwenden, um die Seite auf dem Mobilgerät zu debuggen. Wenn Sie WebKit verwenden müssen – und das kann sehr gut sein, weil es die dominierende Multi-Device-Engine ist –, können Sie über die Software *weinre* eine Version von Chrome oder Safari auf dem Desktop mit Android-, iOS- oder Blackberry-Emulatoren/-Simulatoren verbinden.

Adobe bietet eine App namens Edge Inspect, die den Browser Chrome mit jedem Gerät synchronisiert, auf dem Edge Inspect läuft (momentan auf iOS und Android). Dadurch können Sie Ihre Site auf vielen verschiedenen Geräten gleichzeitig betrachten und den Web-Inspector für das Remote-Debugging nutzen.

1.8 Zusammenfassung

Dieses Kapitel liefert Ihnen alle Informationen, die Sie für den Einstieg in die moderne Webentwicklung benötigen. Ich habe die allgemein übliche Definition von HTML5 enger gesteckt und Sie mit der Webplattform vertraut gemacht. Sie haben gelernt, wozu HTML5 dient und wie Sie es schreiben. Außerdem haben Sie eine kurze Einführung in CSS3 erhalten.

Die Schlüsselaussagen des Kapitels finden Sie den späteren Kapiteln wieder. Erstens: Halten Sie sich stets über den Implementierungsgrad von Webplattform-Funktionen in den verschiedenen Browsern auf dem Laufenden. Zweitens: Testen Sie Ihre Sites, testen Sie sie noch einmal, testen Sie weiter, und wenn Sie denken, dass Sie genug getestet haben, testen Sie erneut. Und dann noch einmal. Nach all diesen Erläuterungen krempeln wir die Ärmel hoch und machen uns an die Arbeit.

1.9 Literaturempfehlungen

Falls Sie es überlesen haben: Die *Liste der Technologien, die die Webplattform ausmachen*, finden Sie auf http://platform.html5.org/. Bruce Lawson schlug den Begriff NEWT in seinem Blog vor: http://www.brucelawson.co.uk/2010/meet-newt-new-exciting-web-technologies/.

Die *HTML5-Spezifikation des W3C* finden Sie auf http://www.w3.org/TR/html5/ und die *Live-Spezifikation des WHATWG* auf http://whatwg.org/html. Es gibt dort auch eine nützliche *Edition for Web Developers*, in der gewisse obskure Sprachelemente weggelassen werden und die deshalb besser lesbar ist: http://developers.whatwg.org/.

Die *vollständige Formulierung des HTML5-Standards* gibt es auf http://html5boilerplate.com/. Denken Sie daran: Verwenden Sie nur die von Ihnen benötigten Teile; Sie müssen nicht alles wörtlich umsetzen.

Mehr über den Implementierungsgrad verschiedener Funktionen bieten Alexis Deverias Site *Can I Use* auf http://caniuse.com/, die Community-Site *HTML5 Please* auf http://html5please.com/ sowie *The HTML5 Test* auf http://html5test.com/.

Die Website *LabUp!* ist eine Ressource für Open Device Labs: http://lab-up.org/. Der Testleiter der BBC, David Blooman, beschäftigte sich in dem langen und ausführlichen Artikel *Testing for Dummies* damit, wie ein großes Unternehmen Multi-Device-Tests durchführt: http://mobiletestingfordummies.tumblr.com/post/20056227958/testing.

Patrick Meenans Präsentationsfolien für seinen Vortrag *Taming the Mobile Beast* enthalten eine Fülle von Links und Informationen über Tests auf mobilen Geräten: http://www.slideshare.net/patrickmeenan/velocity-2012-taming-the-mobile-beast/. Und Anna Debenhams Artikel für A List Apart *Testing Websites in Game Console Browsers* handelt von ... na ja, der Titel ist ziemlich selbsterklärend: http://www.alistapart.com/articles/testing-websites-in-game-console-browsers/.

Opera bietet eine *detaillierte Anleitung zum Remote-Debugging*: http://www.opera.com/dragonfly/documentation/remote/. *weinre* können Sie sich von http://people.apache.org/~pmuellr/weinre/docs/latest/ herunterladen. Weitere *Informationen über Adobe Edge Inspect* erhalten Sie auf http://html.adobe.com/edge/inspect/.