# 2 Struktur und Semantik

*Kennen Sie das Gleichnis von dem Mann, der sein Haus auf Sand baute? Oder das von den Schweinen, die ihre Häuser aus Stroh und Stöcken errichteten? Das sind Verlierertypen. Sie haben verloren, weil sie zu wenig Wert auf die Wichtigkeit der Struktur legten.*

## 2.1 Grundsätzliche Anmerkungen

Gute Sites benötigen eine gute Struktur. Im Web ist die Grundlage dafür HTML.

Durch einen entsprechenden Seitencode erhalten Sie nicht nur jetzt, sondern auch in Zukunft eine tragfähige Struktur. Ob Sie nun eine stark interaktive Webapplikation entwickeln, eine Hybrid-App für Mobilgeräte oder eine digitale Broschüre mit einer einzigen Seite – immer hat es oberste Priorität, dass Sie eine solide Struktur erzeugen. Dadurch wird Ihre Site besser zugänglich; Erstellung und Pflege werden vereinfacht. Browser und andere User Agents können den Sinngehalt Ihrer Seiten leichter erfassen. Ein gut strukturiertes DOM kann auch die Performance Ihrer Site stark verbessern, weil der Browser sie leichter parsen kann und der Speicherbedarf sinkt.

Semantischer Gehalt geht über die bloße Struktur hinaus. Wenn Sie dem Inhalt Ihrer Seiten einen solchen zusätzlichen Sinngehalt geben, erzielen Sie einen unmittelbaren Vorteil: Die Suchmaschinen können Ihre Datei leichter durchsuchen und verstehen. Und außerdem könnten sich längerfristig Vorteile auftun, die bisher noch nicht ersichtlich sind.

All dies wird durch HTML5 und verwandte Technologien vereinfacht. Wenn Sie vorhandene und gut implementierte Methoden verwenden, können Sie stabile, aussagekräftige, leistungsstarke und datenintensive Seiten erstellen.

## 2.2 Neue HTML5-Elemente

Eine der wichtigsten neuen HTML5-Funktionen sind mehrere semantische Elemente, durch die HTML weit über seinen ursprünglichen Zweck hinaus – die Auszeichnung wissenschaftlicher Dokumente mit Überschriften, Listen und Absätzen – ausgebaut wird. Die meisten dieser neuen Elemente sollen die Seitenstruktur verbessern. Der Entwickler soll damit die Möglichkeit zur Auszeichnung inhaltlicher Elemente erhalten, statt einfach ein `div`-Element in Verbindung einer `id` oder mit Klassen zu verwenden.

Hier ein Beispiel. Bisher haben Sie etwa den folgenden Code verwendet:

```
<div class="article">...</div>
```

In HTML5 können Sie stattdessen die folgende Zeile notieren:

```
<article>...</article>
```

*Structural Elements*

Die HTML5-Spezifikation des W3C listet neun strukturierende Elemente (*Structural Elements*) auf. Drei davon gibt es bereits in HTML4: `body`, `h1–h6` (wenn wir ein bisschen schummeln und die Überschriftelemente als eine Einheit zählen) sowie `address`.

*Sectioning Content*

Von den sechs neuen Elementen gehören vier in die Kategorie *Sectioning Content* der HTML5-Spezifikation. Ich erkläre Ihnen gleich, was es damit auf sich hat. Zunächst liste ich diese Sektionselemente aber einfach auf:

- `article` – ein unabhängiger Teil eines Dokuments oder einer Site, zum Beispiel ein Foren- oder Blog-Beitrag oder vom Benutzer übermittelter Inhalt
- `aside` – ein lose mit dem umliegenden Inhalt verbundener Seitenbereich, der aber als getrennte Einheit betrachtet werden sollte, etwa eine seitliche Kolumne in einem Magazinartikel
- `nav` – der Navigationsbereich eines Dokuments, ein Bereich, der Links zu anderen Dokumenten oder zu anderen Bereichen desselben Dokuments enthält
- `section` – eine thematisch zusammenhörige Gruppe von Inhalten, beispielsweise ein Buchkapitel, ein Register in einem durch Tabs unterteilten Dialogfeld oder die Einführung auf der Startseite einer Website

Die anderen zwei strukturierenden Elemente definieren Bereiche innerhalb des Seiteninhalts[1]:

- `footer` – die Fußzeile eines Dokuments oder ein Dokumentbereich mit Metadaten über den Abschnitt, in dem er sich befindet – zum Beispiel Details über den Autor
- `header` – meist die Kopfzeile eines Dokuments; es könnte sich aber auch um den Kopfbereich eines Dokumentteils handeln, im Allgemeinen mit h1- bis h6-Elementen für die Auszeichnung von Überschriften

HTML5 enthält noch weitere neue Elemente, die nichts an der grundlegenden Seitenstruktur ändern; ich bespreche sie bei Bedarf in den übrigen Kapiteln dieses Buchs. Im Moment beschäftigen wir uns damit, warum diese neuen Elemente eigentlich definiert wurden.

### 2.2.1 Was bringen die neuen Elemente überhaupt?

Das definierte Ziel dieser neuen Elemente besteht darin, eine klare Dokumentstruktur zu schaffen, damit die Dokumente besser vom Browser und anderen Geräten, etwa Screenreadern, geparst werden können. Stellen Sie sich diese Struktur wie eine Landkarte des Dokuments vor, die die Hierarchie der darin enthaltenen Inhalte darstellen, aufzeigen, welche Überschriften am wichtigsten sind, die Abhängigkeiten der Inhaltsbereiche verdeutlichen usw.

In HTML 4 wurden dazu in erster Linie die Überschriftelemente `h1` bis `h6` verwendet. Die Überschrift `h1` stellte die einzige oder die Hauptüberschrift auf der Seite dar; `h2`-Elemente standen normalerweise in direkter Abhängigkeit von `h1` usw. Typisch war etwa die folgende Struktur:

```
<h1>Menschenaffen</h1>
  <h2>Gorilla</h2>
    <h3>Östlicher Flachlandgorilla</h3>
    ...
    <h3>Westlicher Flachlandgorilla</h3>
    ...
  <h2>Orang-Utan</h2>
  ...
```

1. Das ursprünglich dazugehörige Element hgroup sollte zur Gruppierung mehrerer Überschriftebenen dienen, beispielsweise einer Unterüberschrift oder eines Slogans. Es wurde aber im April 2013 wieder gestrichen (http://lists.w3.org/Archives/Public/public-html-admin/2013Apr/0003.html).

Diese Verschachtelung der Überschriften erzeugte die folgende Dokumentstruktur:

1. Menschenaffen

1.1 Gorilla

1.1.1 Östlicher Flachlandgorilla

1.1.2 Westlicher Flachlandgorilla

1.2 Orang-Utan

**Hinweis**
Menschenaffen-Fans ist bestimmt aufgefallen, dass ich den Bonobo und den Schimpansen weggelassen habe. Dies geschah nicht aus Voreingenommenheit, sondern aus Platzgründen und zur Verdeutlichung.

*Implicit Sectioning*

Die von mir erzeugte Struktur ist visuell sinnvoll, und Sie erhalten auf diese Weise eine Dokumentstruktur, die als »implizite Gliederung« (*Implicit Sectioning*) bezeichnet wird.

*Explicit Sectioning*

In HTML5 erzeugen Sie mit den weiter vorne besprochenen Sektionselementen die Abschnitte der Gliederung – nicht durch die Überschriften innerhalb dieser Abschnitte. Dies ist eine »explizite Gliederung« (*Explicit Sectioning*). Um also in HTML5 dieselbe Struktur für unsere Menschenaffen zu erzielen, könnten wir Folgendes notieren:

```
<h1>Menschenaffen</h1>
<section>
  <h1>Gorilla</h1>
  <article>
    <h1>Östlicher Flachlandgorilla</h1>
    ...
  </article>
  <article>
    <h1>Westlicher Flachlandgorilla</h1>
    ...
  </article>
</section>
<section>
    <h1>Orang-Utan</h1>
    ...
</section>
```

*Sectioning Content*

Sie würden dasselbe Ergebnis wie im HTML4-Beispiel erhalten, weil jedes `section`- oder `article`-Element einen neuen Abschnitt in der Gliederung erzeugt. Dies sind die Sektionselemente, mit denen man Inhalte

in Abschnitte unterteilt (*Sectioning Content*) und die ich weiter vorne zusammen mit den Elementen `aside` und `nav` erwähnt habe.

Jeder Abschnitt sollte eine Überschrift haben – eine beliebige Überschriftebene ist in Ordnung. In meinem Beispiel habe ich überall `h1`-Überschriften verwendet. In Wirklichkeit ist es aber völlig gleichgültig, welche Überschriftebene ich wähle, weil ich neue Abschnitte per Sectioning Content erzeuge. Genauso gut hätte ich die Überschriftebenen nach dem Zufallsprinzip vergeben können.

**Hinweis**

Das ist etwas leicht dahingesagt – Sie können (und sollten) trotzdem die Überschriftebenen `h1` bis `h6` auf hierarchische Weise einsetzen, das garantiert eine gute Abwärtskompatibilität und vereinfacht die Gestaltung.

*Sectioning Roots*

Genau wie eine Überschrift kann jeder Abschnitt einen eigenen `header` und `footer` enthalten, darüber hinaus weitere Abschnitte und sogenannte *Sectioning Roots*. Solche Sectioning Roots wie etwa `blockquote` und `figure` können mit einer eigenen Gliederung versehen sein. Sie ändern jedoch nichts an der Gesamtgliederung des Dokuments.

Wenn Ihnen all dies etwas unklar erscheint, sind Sie in guter Gesellschaft. Die Verwirrung darüber, wozu die einzelnen Sektionselemente gedacht sind, hat so weit geführt, dass der gute HTML5 Doctor ein Schaubild veröffentlicht hat (Abb. 2–1). Dieses Bild soll Ihnen helfen, das jeweils richtige Element für eine bestimmte Aufgabe zu finden. Ein Schaubild! Damit Sie ein Element auswählen können! Wenn Sie etwas zwischen den Zeilen lesen können, ist Ihnen inzwischen wahrscheinlich ganz richtig aufgefallen, dass ich kein Fan der neuen strukturierenden Elemente in HTML5 bin.

### 2.2.2 Der Nachteil der Sektionselemente in HTML5

*article oder section?*

Wie Sie an meinem zweifellos frustrierten Unterton im vorigen Abschnitt erkannt haben, kann es recht anspruchsvoll sein, mit manchen dieser neuen Elemente zurechtzukommen. Dies gilt besonders für den Unterschied zwischen `article` und `section`. Zur Wiederholung: Ein `section`-Element kann Artikel und Abschnitte enthalten, und ein `article`-Element kann Abschnitte und Artikel enthalten – und beide erzeugen Abschnitte in der Gliederung. Es gibt zwar einen Unterschied zwischen den beiden Elementen, aber niemand – noch nicht einmal der Autor der Spezifikation – hat bisher eine Definition liefern können, die so klar und deutlich wäre, dass die Entwickler sie sich leicht merken können.

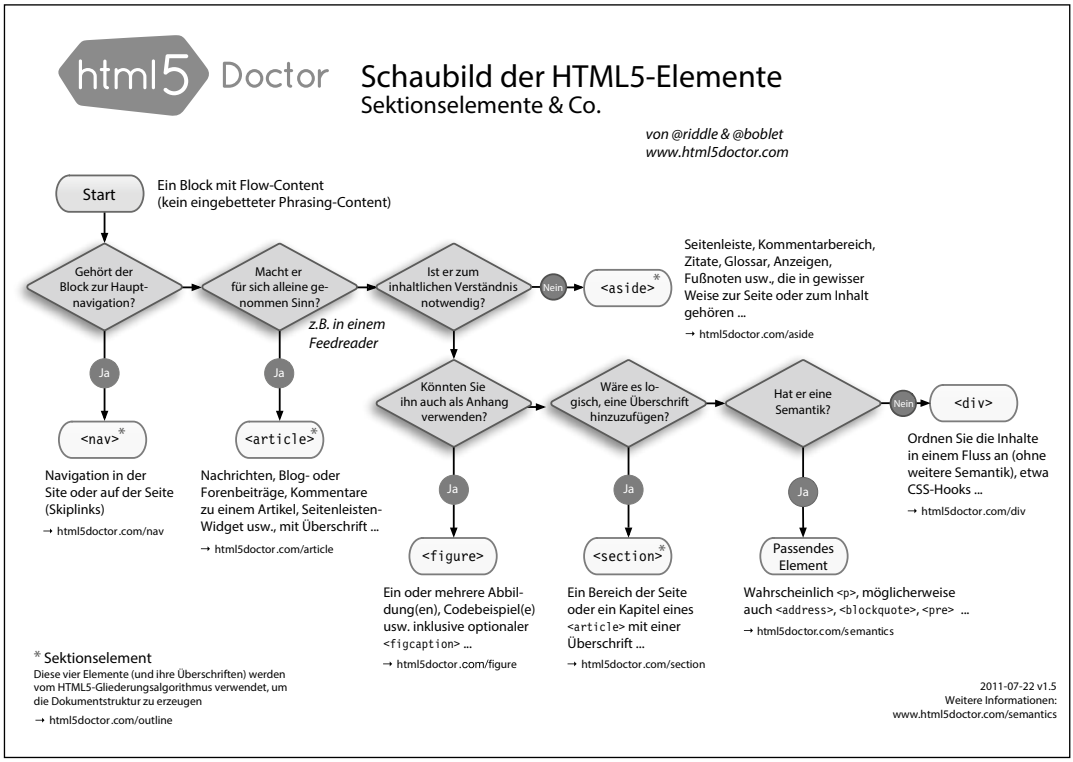


**Abb. 2–1**
*Wenn Sie nicht wissen, welches Sektionselement das richtige ist, liefert der HTML5 Doctor Ihnen die Antwort.*

In seinem im Jahr 2012 bei Create Space erschienenen Buch *The Truth About HTML5* schreibt Luke Stevens über die vage Beschreibung des `article`-Elements:

> *Spezifikationen versagen, wenn sie bei Ihnen Fragen offen lassen. Der Sinn einer Spezifikation besteht darin, genau zu spezifizieren, was Sie tun sollen. In diesem Fall lässt sie aber Interpretationsspielraum, bietet keinen klaren Vorteil und wiederholt vorhandene Funktionen ...*

Ich muss ihm beipflichten. Ich sage jetzt schon voraus, dass viele Entwickler mit diesen Elementen üblen Missbrauch treiben werden, bis es klarere Definitionen gibt.

Aus technischen Gründen empfehle ich, die neuen Elemente im öffentlichen Internet nicht zu nutzen: Zum einen werden sie in älteren Versionen des Internet Explorers (bis zur Version 8) einfach nicht unterstützt. Damit diese Browser die neuen Elemente erkennen, müssen Sie sie in JavaScript erzeugen. Dies geht recht problemlos; fügen Sie einfach in einem bedingten Kommentar das bekannte JavaScript-Skript *HTML5-Shiv* ein:

```
<!--[if lt IE 9]>
<script src="html5shiv.js"></script>
<![endif]-->
```

Allerdings machen Sie Ihre Besucher dadurch von JavaScript abhängig: Alle Nutzer, die einen älteren Internet Explorer mit abgeschaltetem JavaScript nutzen, sehen zwar den Inhalt der neuen Elemente, allerdings ist er unformatiert. Sicherlich dürfte es sich dabei nur um einen geringen Nutzeranteil handeln. Barrierefreiheit sollte aber bedeuten, dass *jeder* Ihre Inhalte sehen kann.

Hinzu kommt, dass keiner der aktuellen Browser den neuen Outline-Algorithmus unterstützt (nur der Screenreader JAWS unterstützt ihn, jedoch mit Bugs), sodass all Ihre harte Arbeit in Wirklichkeit kaum Sinn hat – in der Zukunft wird sich das aber aller Voraussicht nach ändern.

*Alternativen suchen*

Im Endeffekt liegt die Entscheidung natürlich bei Ihnen, ob Sie die neuen strukturierenden Elemente verwenden oder nicht. Sie sind nicht zwingend vorgeschrieben – nach wie vor können Sie mit `div`-Elementen arbeiten. Es fällt mir aber schwer, die neuen Elemente zu empfehlen – es sei denn, Sie sind bereit, die HTML5-Spezifikation zu lesen und genau zu verstehen, wie die strukturierenden Elemente auf die Dokumentstruktur wirken, und vorausgesetzt, Sie arbeiten in einer Umgebung, in der veraltete Browser kein Thema sind. So, wie die Dinge momentan (während ich dieses Buch schreibe) liegen, würde ich nach einer alternativen Möglichkeit zur Darstellung der Seitenstruktur suchen. Glücklicherweise gibt es diese Möglichkeit in Form von WAI-ARIA.

## 2.3 WAI-ARIA

*Barrierefreiheit*

Die *Accessible Rich Internet Applications*-Suite *(WAI-ARIA)* wurde als Antwort auf die mangelnde Barrierefreiheit des Web entwickelt. WAI-ARIA entstand, als im Web nicht mehr nur einfache Dokumente, sondern immer mehr interaktive Anwendungen veröffentlicht wurden. Dazu wurde eine Anzahl HTML-Erweiterungen entwickelt (genauer gesagt Erweiterungen für beliebige DOM-basierte Sprachen wie SVG und XML), sodass die Entwickler Browser und andere Geräte, etwa Screenreader, auf interaktive Inhalte aufmerksam machen konnten.

Wenn Sie beispielsweise einen Link haben, der beim Anklicken JavaScript einsetzt, um einen Dialog zu öffnen, dann haben Sie keine Möglichkeit, den Browser darauf aufmerksam zu machen. Der Code sieht einfach aus wie ein ganz normaler Link:

```
<a href="http://example.com">Launch popup</a>
```

Weil das Event dem Link per Skript zugeordnet ist, hat ein Screenreader keine Informationen darüber, was hier passiert, sodass dem Nutzer diese Funktionalität verborgen bleibt. WAI-ARIA führt genau für diese Situation das neue Attribut `aria-haspopup` ein, sodass der Benutzer weiß, was vor sich geht:

```
<a href="http://example.com" aria-haspopup="true">Launch popup</a>
```

*Landmark Roles*

Neben zahlreichen anderen Attributen sind sogenannte *Landmark Roles* verfügbar. Diese Attribute informieren Screenreader und andere assistive Navigationsgeräte über die Struktur Ihrer Seite, sodass sich der Benutzer leicht in Ihrem Dokument zurechtfinden kann. So erfüllt Ihre Seite die strukturellen Aufgaben, für die eigentlich die neuen HTML5-Elemente gedacht sind.

**Hinweis**

Wie gesagt: Während ich dieses Buch schreibe, parsen manche User Agents und assistiven Geräte die neue HTML5-Dokumentgliederung nicht korrekt. Deshalb könnte es für die Abwärtskompatibilität sinnvoll sein, Landmark Roles zu verwenden.

Landmark Roles werden mit dem `role`-Attribut und einer Reihe vordefinierter Werte zugewiesen. Diese Werte entsprechen nicht direkt den strukturierenden Elementen in HTML5; meist lassen sie sich sehr gut mit diesen vergleichen. Zur Definition des Dokumentbereichs mit den allgemeinen Informationen über die Site, beispielsweise Logo und Slogan, verwenden Sie beispielsweise die `banner`-Role:

```
<div role="banner">...</div>
```

Diese Role entspricht weitgehend dem `header`-Element und wird in den meisten Fällen an dessen Stelle verwendet.

Möchten Sie dem Screenreader mitteilen, wo sich der Hauptinhalt befindet, nehmen Sie die `main`-Role:

```
<div role="main">...</div>
```

Lange Zeit gab es hierfür keine HTML5-Entsprechung, aber im Frühsommer 2013 wurde hierfür das `main`-Element eingeführt.

In der WAI-ARIA-Spezifikation sind acht Landmark Roles definiert:

- `application` definiert einen Seitenbereich, der kein Dokument, sondern eine interaktive Applikation ist

- `banner` definiert, wie erwähnt, allgemeinen Site-Inhalt, wahrscheinlich im Seitenkopf; entspricht in diesem speziellen Zusammenhang dem `header`-Element
- `complementary` definiert Inhalte, die sich zwar auf den Hauptinhalt beziehen, in diesem aber nicht enthalten sind, etwa eine Seitenleiste; entspricht dem `aside`-Element
- `contentinfo` liefert Informationen über das Dokument, zum Beispiel rechtliche Hinweise; häufig in der Fußzeile angesiedelt und deshalb in diesem Zusammenhang dem `footer`-Element entsprechend
- `form` definiert Formulare (außer Suchformulare), beispielsweise ein Kontaktformular
- `main` definiert den Hauptinhalt eines Dokuments
- `navigation` definiert Link-Gruppen für die Navigation im aktuellen Dokument oder in zugehörigen Dokumenten, entspricht dem `nav`-Element
- `search` definiert Formulare, die speziell für die Suche in dieser oder anderen Sites gedacht sind

*CSS-Formatierung*

Die Landmark Roles sind nicht nur für Navigation und semantische Informationen wichtig, sondern eignen sich auch gut als Ankerpunkte für die Formatierung per CSS. Mithilfe des Attributselektors, der auf einen bestimmten Wert prüft, können Sie beispielsweise dem Seitenkopf problemlos Regeln zuweisen:

```
div[role='contentinfo'] { background-color: blue; }
```

Dieser Selektor ist in so ziemlich jedem (mir bekannten) Browser, der in den letzten zehn Jahren entwickelt wurde, implementiert. Wenn Sie also nicht gerade einen sehr alten oder sehr primitiven Browser verwenden, ist diese Technik sinnvoll – denken Sie jedoch daran, dass komplexe Selektoren einen negativen Effekt auf die Seitenladezeit haben können.

## 2.4 Die Bedeutung semantischen Markups

Bevor wir uns mit weiteren Möglichkeiten befassen, Ihren Seiten mehr Sinngehalt zu verleihen, legen wir eine Pause ein und stellen uns die Frage: Warum sollten wir uns überhaupt um die Semantik kümmern? Spricht grundsätzlich etwas dagegen, eine Seite auf die folgende Weise hauptsächlich mit div-Elementen auszuzeichnen?

```
<div class="first">This is the heading.</div>
<div class="main"><b>This is the first sentence.</b><br> This is the
second sentence.</div>
```

Divya Manian ging darauf in ihrem polemischen Artikel »Our Pointless Pursuit of Semantic Value« ein. Sie stellt darin die These auf, dass es für die meisten Menschen eine Zeitverschwendung ist, zu viel Gewicht auf semantisches Markup zu legen:

> *Das Markup strukturiert den Inhalt; aber welche Elemente wir auswählen, ist längst nicht so wichtig, wie wir gelernt haben ...*

*Wartungsfreundlichkeit*

Ich finde jedoch, dass es zwei gute Gründe für die Verwendung semantisch korrekter Elemente gibt. Die erste und nüchternste ist, dass Sie auf einen De-facto-Standard mit guter Wartungsfreundlichkeit hinarbeiten. Wenn Sie semantische Elemente verwenden, sind Sie sicher, dass Ihre Kollegen oder eventuellen Nachfolger an Ihrem Code arbeiten können, ohne dass sie die von Ihnen verwendeten Namenskonventionen erlernen müssten. Und auch das Gegenteil trifft zu. Wenn Sie den standardgemäß notierten Code eines anderen Entwicklers übernehmen, wissen Sie genau, was darin vor sich geht.

*Aboutness*

Ein eher metaphysischer Grund ist, dass die Verwendung semantischer Elemente Ihrem Inhalt verstärkte »Aboutness« verleiht. Einfach ausgedrückt, ist Aboutness ein Maß für die Qualität des Sinngehalts; das Wesen einer Sache wird durch seine Aboutness beschrieben.

Als einfaches Beispiel für dieses Prinzip stellen Sie sich eine Webseite mit W. H. Audens Gedicht »Funeral Blues« vor:

> *He was my North, my South, my East and West,*
> *My working week and my Sunday rest,*
> *My noon, my midnight, my talk, my song;*
> *I thought that love would last forever: I was wrong.*

Zwar wissen *wir*, dass das Gedicht vom Tod handelt; aber das Wort »Tod« taucht nicht in dem Gedicht auf. Wie könnte eine Suchmaschine, die die Seite indiziert, wissen, wovon sie handelt, und diesen Sinngehalt in den Suchergebnissen für das Thema »Tod« zurückgeben? Die Suchmaschine betrachtet den Text der Links zu dieser Seite. Also bringt ein Link mit dem Text »Weiterlesen« in diesem Zusammenhang nicht viel, während ein Link mit dem Text »W.H. Audens Gedicht über den Tod« Aboutness bietet.

Die Verwendung korrekter semantischer Elemente hat denselben Vorteil. Wenn alle Inhalte auf Ihrer Seite mit `div`-Elementen ausgezeichnet sind, bietet der Inhalt keinen Sinnzusammenhang; wenn Sie Ihre Seite semantisch auszeichnen, erhält der Inhalt eine Bedeutung:

```
<h1>This is the heading.</h1>
<p>This is the first sentence.</p>
<p>This is the second sentence.</p>
```

Nun wissen Sie genau, welche Überschrift wichtig ist und was der Hauptinhalt ist. Sie haben den Inhalt mit Aboutness versehen.

Neben der korrekten Verwendung semantischer Elemente zur Auszeichnung Ihres Inhalts können Sie die Aussagekraft Ihrer Dokumente nicht nur für menschliche Benutzer, sondern auch auf unterschiedliche Weise für Maschinen verbessern (man spricht dann von »strukturierten Daten«). Sie können vorhandene Attribute und Elemente mit vordefinierten Mustern (*Mikroformaten*) oder HTML mit neuen Attributen (*RDFa* und *Mikrodaten*) erweitern. Diese Möglichkeiten stelle ich jetzt kurz vor.

## 2.5 Mikroformate

*Keine HTML5-Erweiterung*

Mikroformate wurden von Entwicklern »von unten«, d.h. ohne Mitwirkung der bekannten Normungsgremien, definiert. Sie versehen Inhalte durch standardisierte Markup-Muster für vorhandene Attribute mit zusätzlicher Bedeutung. Ihr Hauptvorteil ist, dass sie auf aktuellen Entwicklungsmethoden basieren; sie sind keine HTML-Erweiterung, sondern ein Design-Prinzip bzw. ein Satz von standardisierten Nutzungsmustern.

*Rel-Tag*

Mikroformate können ziemlich komplex oder auch ganz einfach sein. Hier ein Beispiel für das wahrscheinlich einfachste Mikroformat:

```
<a href="http://flickr.com/photos/tags/gorilla/"
   rel="tag">Gorillas</a>
```

Dies ist das Mikroformat *Rel-Tag*. Das Schlüsselwort `tag` im Attribut `rel` informiert Maschinen darüber, dass die im `a`-Element referenzierte URL eine Seite ist, die durch ein Element beschrieben wird, dessen Name die letzte Pfadkomponente der URL ist – in diesem Fall `gorilla`. An diesem Punkt sollte ich wahrscheinlich darauf hinweisen – falls es Ihnen noch nicht selbst aufgefallen ist –, dass ich Gorillas sehr mag.

*hCard*

Ein komplexeres, aber genauso geradliniges und nützliches Beispiel ist das Mikroformat *hCard*. Es zeichnet häufig verwendete Personeninformationen mit einer standardisierten »Visitenkarten-Syntax« aus. Mit den folgenden Zeilen könnte ich beispielsweise online zu meinen Kontaktdaten verlinken:

```
<div class="details">
<p><a href="http://about.me/petergasston">Peter Gasston</a>
writes for <a href="http://broken-links.com">Broken Links</a>.</p>
</div>
```

Der Code ist in Ordnung, er hat jedoch nicht viel Aboutness. Der Leser würde verstehen, dass es sich um meinen Namen und um meinen Blog handelt; aber ein Suchmaschinen-Spider wäre wahrscheinlich zu diesem kognitiven Sprung nicht imstande. Mit dem *hCard*-Muster kann ich den Code mit Attributwerten versehen, die seinen semantischen Gehalt vermehren:

```
<div class="vcard">
<p><a href="http://about.me/petergasston" class="url fn">
Peter Gasston</a>
writes for <a href="http://broken-links.com" class="url org">
Broken Links</a>.</p>
</div>
```

Diese paar standardisierten Klassennamen machen diesen Inhalt für Suchmaschinen sehr viel sinnvoller; ein Web-Crawler, der das Muster *hcard* erkennt, sieht die Klasse `vcard` und weiß, dass deren Inhalt die gesuchten Informationen enthält: Er weiß, dass der erste Link den vollständigen Namen (`fn` = full name) des Kontakts, der zweite Link die Organisation (`org`) enthält, für die der Kontakt arbeitet.

Es gäbe noch mehr über Mikroformate zu sagen; es gibt Mikroformate für Veranstaltungen, Besprechungen, geografische Koordinaten, sogar für Rezepte und noch vieles andere. Sie werden von Suchmaschinen wie Google verwendet, um die Suchergebnisse zu verbessern. Also ist die Verwendung von Mikroformaten in Ihren Seiten durchaus nicht (nur) eine Übung in Entwickler-Esoterik.

## 2.6 RDFa

*Eine HTML5-Erweiterung*

Das *Resource Description Format in Attributes (RDFa)* ist eine HTML-Erweiterung, die Inhalte unter Verwendung eines ganzen neuen Satzes maßgeschneiderter Attribute für Maschinen sinnvoll und verständlich macht.

Die Hauptsyntax ist als *RDFa Core* bekannt. Weiterhin ist die einfachere Untermenge *RDFa Lite* verfügbar. Beide beschreiben Inhalte mithilfe vordefinierter Schemata (Datenbeschreibungen).

Statt zu erläutern, was damit gemeint ist, zeige ich es Ihnen (die Jahre an der Schule für Drehbuchautoren waren nicht umsonst!). Ein typisches Beispiel für Daten im Web, vor allem in Blogs und News-Sites, ist das Datum. Recht häufig wird ein Datum etwa folgendermaßen ausgezeichnet:

```
<p class="date">2013-04-01</p>
```

Dieser Code ist funktional und unkompliziert, aber der einzige semantische Sinngehalt wird durch den Klassennamen vermittelt. Mit RDFa Lite können Sie seinen Kontext verbessern, sodass Maschinen erkennen können, dass es sich um ein Datum handelt. Dazu verwenden Sie das neue Attribut `property`:

*property*

```
<p property="http://purl.org/dc/elements/1.1/date">2013-04-01</p>
```

Der Wert des Attributs ist die URL der Beschreibung des Begriffs `date` in einem Schema, das zum standardisierten Vokabular *Dublin Core* gehört.

Sie haben wahrscheinlich bemerkt, dass das von mir verwendete Datum kein besonders leserfreundliches Format hat. Dies ist der Nachteil von RDFa Lite: Alle Inhalte müssen auf eine streng maschinenlesbare Weise formatiert werden. Damit der Inhalt für menschliche Betrachter besser geeignet ist, müssen sie RDFa Core verwenden. Mit RDFa Core kann ich mithilfe des Attributs `content` einen Satz Informationen für Maschinen und einen anderen für Menschen liefern:

```
<p property="http://purl.org/dc/elements/1.1/date"
   content="2013-04-01">April 1</p>
```

Die Leser sehen den Inhalt des Elements; die Maschine liest den Wert des Attributs. Dazu ist Zusatzcode notwendig, aber alle sind zufrieden. Wie bei den Mikroformaten suchen manche Suchmaschinen nach RDFa-Mustern, um ihre Suchergebnisse zu verbessern. Auf Seite 45 im Abschnitt »Rich Snippets« gehe ich ins Detail.

## 2.7 Mikrodaten

HTML5 begegnet dem Semantikproblem mit einer einfachen Syntax, die *Mikrodaten* (*Microdata*) genannt wird. Prinzipiell handelt es sich dabei um eine Reihe von Name-Wert-Paaren, die sinnvolle maschinenlesbare Daten liefern. Auch hier versuche ich nicht, es zu erklären, sondern zeige Ihnen lieber, wie es funktioniert:

*Microdata*

```
<p itemscope>I live in <span itemprop="city">London</span></p>
```

Dieser Code erzeugt ein einzelnes Element *(Item)*. Das Attribut `itemscope` wird für das Container-Element verwendet, um den Geltungsbereich (*Scope*) dieses speziellen Items zu definieren.

*Item*

Innerhalb des Containers definieren Sie das Name-Wert-Paar, das P*roperty* genannt wird. Der Wert des Attributs `itemprop` ist der Name – in diesem Beispiel `city` –, und der Inhalt des Elements ist der Wert – in

diesem Beispiel `London`. Das Ergebnis ist ein Element mit einer einzelnen Eigenschaft:

```
city: 'London'
```

Sie sind jedoch nicht auf eine einzelne Eigenschaft beschränkt; Sie können so viele Eigenschaften definieren, wie Sie möchten:

```
<p itemscope>Hello, my name is <span itemprop="given-name">Peter
</span> and I'm <span itemprop="role">a developer</span> from
<span itemprop="city">London</span>.</p>
```

In diesem Fall sieht die Eigenschaftenliste des Elements folgendermaßen aus:

```
given-name: 'Peter'
role: 'a developer'
city: 'London'
```

Wie Sie erkennen können, ist dieser Code durchaus RDFa-ähnlich, und genau wie bei diesem Format können Sie Maschinen und Menschen unterschiedliche Werte liefern. Betrachten Sie das folgende Beispiel, in dem ich das Attribut `datetime` verwende:

```
<p itemscope>My birthday this year is on <span itemprop="birthday"
datetime="2013-12-14">December 14</span>.</p>
```

Und Sie können – wie bei RDFa – Inhalte mit vordefinierten Schemata beschreiben, indem Sie sie mit dem Attribut `itemtype` verknüpfen:

```
<p itemscope itemtype="http://example.org/birthday">My birthday
this year is on <span itemprop="birthday" datetime="2013-12-
14">December 14</span>.</p>
```

Dabei können Sie ein Schema wie den zuvor erwähnten Dublin Core verwenden – oder sogar ein von Ihnen selbst erdachtes Schema, wie der vorige Codeblock gezeigt hat.

### 2.7.1 Die API für Mikrodaten

Mikrodaten haben eine zugehörige DOM-API, die zur Extraktion von Daten aus der Seite geeignet und bereits recht umfassend in den modernen Browsern implementiert ist. Der Schlüssel ist die Methode `getItems()`, die eine `NodeList` mit allen Elementen auf der Seite zurückgibt:

```
var items = document.getItems();
```

Von hier aus können Sie ein einzelnes Element auswählen und beispielsweise mit dem `properties`-Objekt prüfen, wie viele Eigenschaften enthalten sind:

```
var firstItemLen = items[0].properties.length;
```

Oder Sie können den Wert einer dieser Eigenschaften herausfinden:

```
var itemVal = items[0].properties['name'][0].itemValue;
```

Ich demonstriere Ihnen dies in der Beispieldatei *microdata-api.html*[2]. Die Ergebnisse habe ich an die Konsole ausgegeben; öffnen Sie also Ihren bevorzugten Browser und sehen Sie nach. Experimentieren Sie ruhig selbst damit. Falls Sie gerade keinen Browser zur Hand haben, zeigt Ihnen Abb. 2-2, wie die Ergebnisse in Firebug aussehen.


**Abb. 2–2**
*Das Ergebnis einiger einfacher Experimente mit der Mikrodaten-API in der Konsole*

### 2.7.2 Mikrodaten, Mikroformate und RDFa

Wenn Sie finden, dass maschinenlesbare semantische Daten für Ihre Seiten das Richtige sind, stellt sich die Frage: Welches Format sollen Sie wählen? Die Antwort lautet wie immer: Es kommt darauf an. Prüfen Sie Ihre Inhalte, informieren Sie sich über die Stärken und Schwächen der einzelnen Datentypen und entscheiden Sie, welcher für Sie am besten geeignet ist.

*Eine Mischung*

Meiner persönlichen Meinung nach wird es in Zukunft meistens eine Mischung von Mikrodaten und einfachen Mikroformaten geben. An Mikrodaten ist besonders interessant, dass sie beide ihrer Zeitgenossen in ihre eigene, flexible Syntax aufnehmen können. `hCard` wird beispielsweise mithilfe von Mikrodaten folgendermaßen notiert:

```
<div itemscope itemtype="http://microformats.org/profile/hcard">
<p><a href="http://about.me/petergasston" itemprop="url fn">
Peter Gasston</a> writes for <a href="http://broken-links.com"
itemprop="url org">Broken Links</a>.</p>
</div>
```

Ebenso können Sie problemlos das RDFa-Datenschema verwenden:

```
<p itemscope itemtype="http://purl.org/dc/elements/1.1/date"
datetime="2013-04-01">April 1</p>
```

*Mikrodaten*

Meiner Meinung nach wird die Flexibilität von Mikrodaten dazu führen, dass sie immer häufiger eingesetzt werden. Dennoch sind Mikrodaten keine perfekte Lösung für alles; manche Mikroformate – etwa *RelTag* – sind so einprägsam und leicht anwendbar, dass es kaum sinnvoll wäre, sie zu ersetzen.

2. Alle Beispieldateien finden Sie über die dpunkt-Webseite zum Buch (www.dpunkt.de/gasston) oder direkt auf der Website der Originalausgabe (modernwebbook.com).

### 2.7.3 Schema.org

Durch Mikrodaten wird Ihr Inhalt von Suchmaschinen und Portalen erkannt und besser herausgestellt. Auch dies ist ein Grund, warum Mikrodaten meiner Ansicht nach die Mikroformate und RDFa ausstechen werden.

*Google, Microsoft, Yahoo! und Yandex*

Im Jahr 2011 brachten vier große Web-Riesen – Google, Microsoft, Yahoo! und Yandex – die Website Schema.org heraus. Mit dieser führten sie gemeinsame Vokabulare ein, um gängige Muster mit Mikrodaten auszuzeichnen. Dazu gehören häufig im Web verwendete Elemente wie *Reviews, Events, Places, Items* und *Objects*.

*Beispiel Buchrezension*

Zur Verdeutlichung nehmen wir an, dass Sie eine Buchrezension für Ihre Website schreiben (ich habe wahllos ein Buch herausgegriffen und eine ganz wertfreie Rezension geschrieben):

```
<div class="review">
   <h1>The Book of CSS3 by Peter Gasston</h1>
   <p>What an amazing book! 5 stars!</p>
</div>
```

Diese Rezension enthält zwei Elemente: die Buchdetails und die Rezension. Schema.org enthält zwei Vokabulare, die Sie zur Auszeichnung dieser Rezension verwenden können: *Book* und *Review*. Ein Besuch der entsprechenden Seiten auf Schema.org zeigt mir, welche Mikrodatenmuster ich verwenden soll. Anschließend kann ich meinen Code folgendermaßen optimieren:

```
<div class="review" itemscope itemtype="http://schema.org/Review">
   <h1><span itemprop="itemReviewed">The Book of CSS3</span>,
   by <span itemprop="creator">Peter Gasston</span></h1>
   <p><span itemprop="reviewBody">What an amazing book!</span>
   <span itemprop="reviewRating">5</span> stars!</p>
</div>
```

Der Code ist jetzt zwar komplexer geworden, aber auch sinnvoller. Jedes der von mir verwendeten Vokabulare wird im Attribut `itemtype` durch einen Link zum relevanten Schema definiert, und die Elemente sind mit vordefinierten `itemprop`-Werten ausgezeichnet.

Interessant an Schema.org ist die Tatsache, dass bestimmten Schemata die Eigenschaften von übergeordneten Schemata vererbt werden. *Book* beispielsweise besitzt Eigenschaften aus seinem eigenen Schema, dem umfassenderen Vokabular *CreativeWork*, und zudem aus dem Vokabular der höchsten Ebene *Thing* (toller Name!) mit seinen allgemeingültigen Eigenschaften.

Wenn ich meinen Inhalt mit Schema.org-Mustern auszeichne, erkennen alle Crawler auf meiner Seite den Autor und Titel des Buchs, die Tatsache, dass ich es rezensiere und dass ich ihm eine Fünf-Sterne-Bewertung gegeben habe. Wenn jemand nach dem Buch sucht, könnte meine Rezension in den Suchergebnissen erscheinen oder mit anderen Rezensionen gemeinsam präsentiert werden, sodass der Leser eine sinnvolle Übersicht erhält.

### 2.7.4 Rich Snippets

*Google*

Google bezeichnet die Technik, den Suchmaschinen zusätzliche Informationen zu liefern, als *Rich Snippets*. Rich Snippets ordnen einer Suchabfrage einen Kontext zu. Dadurch kann der Nutzer die Relevanz des Ergebnisses besser beurteilen, ohne dass er sich durch die Seite klicken muss. Ein Beispiel für ein Rich Snippet sehen Sie in Abb. 2–3.

**The Book of CSS3**: A Developer's Guide to the Future ... - **Goodreads**
www.goodreads.com › Computer Science › Programming
★★★★☆ Rating: 4.0 - 37 votes
May 13, 2011 – **The Book of CSS3** has 37 ratings and 11 reviews. Tami said: CSS has such potential. Even in its infancy, it's absolutely revolutionized the way ...

**Abb. 2–3**
*Ein Beispiel für ein Rich Snippet, das auf der Google-Suchergebnisseite Zusatzinformationen liefert*

Rich Snippets funktionieren mit Mikroformaten und RDFa; die bevorzugte Syntax sind jedoch Mikrodaten. Entwickler finden auf den Google-Webmaster-Seiten zahlreiche Informationen und Dokumentationen. Hier steht auch ein nützliches Tool zur Verfügung, mit dem Sie die korrekte Formatierung Ihrer Mikrodaten prüfen können. Abb. 2–4 zeigt die Daten, die dieses Tool aus der Buchrezension im vorigen Abschnitt extrahiert hat.

Item

| | |
|---|---|
| **type:** | http://schema.org/review |
| **property:** | |
| itemreviewed: | The Book of CSS3 |
| creator: | Peter Gasston |
| reviewbody: | What an amazing book! |
| reviewrating: | 5 |

**Abb. 2–4**
*Mit dem Rich-Snippet-Testwerkzeug aus der ausgezeichneten Buchrezension extrahierte Daten*

## 2.8 Datenattribute

*HTML5*

Eine weitere HTML5-Möglichkeit zur Erweiterung des Sinngehalts von Elementen sind Datenattribute. Es handelt sich dabei um benutzerdefinierte Eigenschaften, deren Werte Informationen zu einem Element liefern sollen, ohne dass sich aber dessen Sinngehalt für Maschinen oder Menschen erhöht. Dies möchte ich etwas detaillierter erläutern.

Nehmen wir an, Sie möchten einen Satz Daten ausgeben. Jedes Datenelement hat zwei Werte – einen Namen und einen Zahlenwert (beispielsweise eine eindeutige Datenbank-ID). Der Name soll im Dokument angezeigt werden; Sie möchten aber auch den Zahlenwert für Skripte verfügbar machen. Momentan ist kein relevantes Attribut verfügbar; Sie müssten wahrscheinlich eine Klasse verwenden:

```
<p class="id-123">Peter</p>
```

*data-...*

Genau aus diesem Grund wurden Datenattribute geschaffen: Sie sollen Daten zuordnen. Sie können damit solche Zusatzinformationen speichern, ohne ihnen irgendeine zusätzliche Bedeutung zu verleihen, wie es bei der Verwendung von Klassen der Fall wäre. Jedes Datenattribut beginnt mit dem Wort `data`. Darauf folgt ein benutzerdefinierter, eindeutiger Schlüssel. In unserem Beispiel könnte das folgendermaßen aussehen:

```
<p data-id="123">Peter</p>
```

Das Datenattribut *id* ist nun dem Wert *Peter* zugeordnet. Obwohl das Element dadurch keine zusätzliche semantische Bedeutung erhält, ist das Attribut vorhanden und kann von anderen Prozessen genutzt werden: Möglicherweise befinden sich die Informationen über diese Daten in einer zugehörigen JSON-Datei, sodass Sie mithilfe von JavaScript darauf zugreifen können.

### 2.8.1 Die API für Datenattribute

*dataset*

Damit Skripte leichter auf diese Daten zugreifen können, ist eine einfache DOM-API verfügbar, die die `dataset`-Eigenschaft nutzt. Um den Wert eines Datenattributs zu ermitteln, verwenden Sie diese Eigenschaften mit dem Schlüssel des Attributs, das Sie abfragen:

```
var el = document.querySelector('p');
var id = el.dataset['id'];
```

Hier würde das Ergebnis *123* zurückgegeben. Sie können Datenattributwerte mit `dataset` auch aktualisieren:

```
el.dataset['id'] = 100;
```

Hier ein praktisches Beispiel:

```
var el = document.querySelector('p');
console.log('The ID is',el.dataset['id']);
el.dataset['id'] = 100;
console.log('Now the ID is',el.dataset['id']);
```

In diesem Beispiel führe ich drei Operationen aus: Zuerst ermittle ich die *id*-Daten, dann setze ich sie auf 100, dann ermittle ich sie erneut, und jedes Mal gebe ich die Ergebnisse an die Konsole aus. Das Ergebnis zeigt Ihnen Abb. 2–5.


**Abb. 2–5**
*Das Ergebnis der Manipulation des Datenattributs mit der API in der Konsole*

### 2.8.2 jQuery und Datenattribute

Wenn Sie jQuery nutzen, ist der Umgang mit Datenattributen noch einfacher (wenn Sie nicht wissen, was jQuery ist, erfahren Sie es in Kapitel 5; Sie können auf diesen Abschnitt zurückkommen, nachdem Sie das Kapitel 5 gelesen haben).

*data()*

Nutzen Sie die Methode `data()`, um Datenwerte zu ermitteln und zu setzen:

```
var id = $(el).data('id');
```

Dieser Code entspricht dem im vorigen Abschnitt und gibt denselben Wert zurück: *123*.

Es gibt jedoch einen großen Vorteil gegenüber `dataset`, bei dem alle Ergebnisse als Zeichenkette zurückgegeben werden: Die Methode `data()` parst auch den Wert des Attributs und konvertiert ihn in den korrekten Typ. Im vorigen Beispiel wäre der Typ ein Zahlenwert. Würden Sie den Code jedoch folgendermaßen ändern ...

```
<p data-name="Peter">123</p>
```

... und wieder die Methode `data()` verwenden, ...

```
var name = $(el).data('name');
```

... wäre der Wert der Variablen *name* nun *Peter* und der Typ ein String.

Ein praktisches Beispiel zeigt Ihnen die Datei *data-attributes-jquery.html*. Darin habe ich die beiden unterschiedlichen Datenattribute in demselben Markup kombiniert:

```
<p data-id="123" data-name="Peter">Gasston</p>
```

Per jQuery habe ich den Typ jedes Datenattributs mit dem JavaScript-Operator `typeof` an die Konsole ausgegeben:

```
var el = $('p');
console.log('ID:',typeof el.data('id'));
console.log('Name:',typeof el.data('name'));
```

Die resultierende Ausgabe sehen Sie in Abb. 2–6.


**Abb. 2–6**
*Den Ergebnistyp mithilfe der jQuery-data()-Methode mit Datenattributen ermitteln*

### 2.8.3 Datenattribute in natura

Datenattribute sind so nützlich, dass manche Unternehmen bereits ausgiebig Gebrauch davon machen. Twitter hat sie schnell als Möglichkeit übernommen, Webseiten einen Tweet-Button hinzuzufügen. Bestimmte inhaltliche Parameter werden in einem Satz vordefinierter Attribute gespeichert:

```
<a href="https://twitter.com/share" class="twitter-share-button"
data-url="http://broken-links.com" data-via="stopsatgreen">Tweet</a>
```

Das Twitter-JavaScript wird an beliebiger Stelle der Seite aufgerufen, und dieses Element wird durch einen Tweet-Button mit Verwendung der bereitgestellten Daten ersetzt. Auch viele andere Social-Media-Plattformen, etwa Facebook, Google+ und LinkedIn, nutzen Datenattribute auf dieselbe Weise.

## 2.9 Webkomponenten: Die Zukunft der Auszeichnungssprachen?

*Web Components*

Ein spannender neuer Ansatz zur Erweiterung von HTML befindet sich in der Vorschlagsphase für die sogenannten *Webkomponenten* (*Web Components*), ein Sammelname für eine Gruppe von Technologien, die die Entwicklung von reichhaltigen Benutzeroberflächen für Webanwendungen mit CSS und Markup-Code vereinfachen sollen.

Während ich dieses Buch schreibe, befindet sich die Spezifikation im Entwurfsstadium und wird sehr wahrscheinlich noch geändert werden. Deshalb beschäftige ich mich nicht hier mit den Webkomponenten, sondern gehe in Kapitel 11 ausführlicher darauf ein.

## 2.10 Zusammenfassung

In diesem Kapitel haben wir uns mit einer Kernfunktion einer guten Website bzw. Webanwendung beschäftigt: der zugrunde liegenden Struktur. Wenn Sie den Code korrekt und sinnvoll auszeichnen, legen Sie den Grundstein für alles Weitere, das in diesem Buch noch besprochen wird. Diese Vorgehensweise ist äußerst wichtig, damit Ihre Sites gut zu pflegen und skalierbar sind.

Ich bin ein bisschen harsch mit den strukturierenden Elementen in HTML5 umgegangen – ob Sie sie aber einsetzen oder nicht: Sie sollten auf jeden Fall darüber nachdenken, in Ihrem Projekt WAI-ARIA zu nutzen. Die Zugänglichkeit von Content ist das Fundament des Web – und selbst wenn Sie ausschließlich Roles nutzen und sonst nichts, ziehen Sie und die Nutzer Ihrer Seiten Vorteile daraus.

Sie haben weiterhin einen Blick darauf geworden, wie Sie Ihre Sites mithilfe semantischer und strukturierter Daten sinnvoller gestalten, Sie haben die Wichtigkeit von Aboutness kennengelernt. Welchen Ansatz Sie wählen, hängt natürlich vom jeweiligen Kontext ab. Zum Aufbau einer großen, datenbankgesteuerten Website mit zahlreichen Inhalten für Endverbraucher sind wahrscheinlich tiefergehende Überlegungen zur Semantik nötig, um Suchmaschinen und Crawler bestmöglich zu bedienen. In diesem Fall sollten Sie über den Einsatz von RDFa, Mikroformaten und/oder Mikrodaten nachdenken. Wenn Sie eine Hybrid-App für Mobilgeräte aufbauen, sind diese Überlegungen hingegen viel weniger wichtig.

## 2.11 Literaturempfehlungen

HTML5 Doctor ist die beste Quelle für Informationen über die meisten HTML5-Themen. Sie finden beispielsweise im folgenden *Artikel von Mike Robinson* die meiner Meinung nach beste Definition der neuen Gliederungsalgorithmen: http://html5doctor.com/outlines/. Das *in Abb. 2–1 gezeigte Schaubild* können Sie von http://html5doctor.com/resources/#flowchart/ herunterladen.

Lesen Sie auch *Derek Johnsons Artikel* im Smashing Magazine: http://coding.smashingmagazine.com/2011/08/16/html5-and-the-document-outlining-algorithm/.

Für tiefere Einblick in das Problem der strukturierenden Elemente in HTML5 (Sectioning Elements) empfehle ich Luke Stevens' Buch *The Truth About HTML5*. Sie finden es auf http://www.truthabouthtml5.com/. Wenn Sie die *vollständige HTML5-Spezifikation* studieren möchten, um selbst zu entscheiden, rate ich Ihnen zur Entwicklerversion auf http://developers.whatwg.org/sections.html.

Die *vollständige WAI-ARIA-Spezifikation* finden Sie auf http://www.w3.org/TR/wai-aria/. Für Informationen über die Barrierefreiheit von HTML5 lohnt es sich, den *Paciello Group Blog* zu lesen. Wichtig ist in diesem Zusammenhang der Beitrag http://www.paciellogroup.com/blog/2010/10/using-wai-aria-landmark-roles/.

*Divya Manians Artikel über Semantik* erschien im Smashing Magazine auf http://coding.smashingmagazine.com/2011/11/11/our-pointless-pursuit-of-semantic-value/. Wenn Sie mehr über Aboutness und die Bedeutung der Semantik lesen möchten, empfehle ich unbedingt das Buch *Ambient Findability: What We Find Changes Who We Become* von Peter Morville (O'Reilly, 2005). Die Website http://webdatacommons.org/ liefert *Informationen und Statistiken über Sites, die strukturierte Daten nutzen.*

Lesen Sie *alles über Mikroformate* auf http://microformats.org/. Eine Überarbeitung der Syntax, Microformats 2.0, begann im Jahr 2010 und ist immer noch in Entwicklung begriffen; erfahren Sie mehr darüber auf http://microformats.org/wiki/microformats-2.

Möchten Sie mehr über das RDFa-Format lesen, hat das W3C eine exzellente *Einführung* veröffentlicht: http://www.w3.org/TR/xhtml-rdfa-primer/. Die beste *Ressource über Mikrodaten* stammt ebenfalls vom HTML5 Doctor: http://html5doctor.com/microdata/. Sind Sie ganz masochistisch veranlagt und möchten Sie die *Spezifikation im Ganzen* lesen, finden Sie diese auf http://www.w3.org/TR/microdata/.

Weitere *Informationen über Schema.org* erhalten Sie auf http://schema.org/, und die *Google-Dokumentation über Rich Snippets* befindet sich auf http://support.google.com/webmasters/bin/answer.py?hl=en&answer=99170. Das *Test-Tool* finden Sie auf http://www.google.com/webmasters/tools/richsnippets/.

John Resig schrieb in seinem Blog http://ejohn.org/blog/html-5-data-attributes/ eine einprägsame *Einführung zu den Datenattributen*, und die `data()`-Methode wird vollständig auf der jQuery-Website auf http://api.jquery.com/data/ dokumentiert.