The Pragmatic Programmers

HTML5 & CSS3

Nutzen Sie schon heute die Webstandards von morgen

Deutsche Übersetzung von O'REILLY®

Brian P. Hogan

Übersetzt von Stefan Fröhlich

Brian P. Hogan, HTML5 & CSS3, O´Reilly, ISBN 9783897213166697838972131669

Inhaltsverzeichnis

Kapitel 2

Neue strukturelle Tags und Attribute

In den ersten Kapiteln dieses Buchs sprechen wir darüber, wie wir mit den Funktionen von HTML5 und CSS3 unseren Benutzern verbesserte Interfaces bieten können. Sie erfahren, wie Sie bessere Formulare erstellen, Tabellen einfacher gestalten und die Barrierefreiheit Ihrer Seiten für unterstützende Geräte verbessern. Außerdem erfahren Sie, wie Sie bei der Erzeugung von Inhalten die Usability von Stylesheets für den Druck optimieren, und Sie erforschen das „In-place-Editing" mit dem neuen Attribut `contenteditable`. Zunächst sehen wir uns aber an, wie wir mit den neuen Elementen von HTML5 unsere Seiten besser strukturieren können.

Zu Beginn würde ich gern mit Ihnen über ein ernst zu nehmendes Problem sprechen, von dem heutzutage viele Webentwickler betroffen sind: Die *Divitis* greift um sich – ein chronisches Syndrom, das dazu führt, dass Webentwickler Elemente in zusätzliche `div`-Tags mit IDs wie zum Beispiel `banner`, `sidebar`, `article` und `footer` verpacken. Dieses Syndrom ist in hohem Maße ansteckend. Divitis verbreitet sich extrem schnell von Entwickler zu Entwickler. Und da `divs` für das bloße Auge unsichtbar sind, bleiben insbesondere leichte Fälle von Divitis unter Umständen jahrelang unbemerkt.

Ein häufiges Symptom von Divitis sieht so aus:

```
<div id="navbar_wrapper">
  <div id="navbar">
    <ul>
      <li><a href="/">Home</a></li>
      <li><a href="/">Home</a></li>
    </ul>
  </div>
</div>
```

Hier wurde eine ungeordnete Liste, die ihrerseits ja bereits ein Blockelement ist[1], in zwei `div`-Tags eingebettet, die auch wiederum Blockelemente sind. Die `id`-Attribute dieser Wrapper-Elemente erklären zwar, welche Aufgabe sie haben. Aber Sie können zumindest einen dieser Wrapper entfernen und zum gleichen Ergebnis kommen. Diese übermäßige Verwendung von Markup führt zu aufgeblähten Seiten, die schwierig zu gestalten und zu pflegen sind.

Es gibt jedoch Hoffnung: Die HTML5-Spezifikation liefert ein Heilmittel in Form von neuen semantischen Tags, die ihren Inhalt beschreiben. Weil so viele Entwickler Seitenleisten, Kopfzeilen, Fußzeilen und Abschnitte in ihren Designs verwenden, werden mit der HTML5-Spezifikation spezielle Tags eingeführt, um Seiten in solche logischen Abschnitte zu unterteilen. Machen wir uns mit diesen neuen Elementen an die Arbeit. Mit HTML5 können wir der Divitis noch in unserer Generation ein Ende setzen.

Neben den neuen strukturellen Tags werden wir auch das `meter`-Element besprechen und Ihnen zeigen, wie Sie in HTML5 mit den neuen benutzerdefinierten Attributen Daten in Elemente einbetten können, ohne dafür Klassen oder existierende Attribute zu missbrauchen. Mit anderen Worten werden wir herausfinden, wie wir das richtige Tag für die richtige Aufgabe einsetzen.

In diesem Kapitel erforschen wie die folgenden neuen Elemente und Funktionen:[2]

1 Sie erinnern sich, dass Blockelemente eine eigene Zeile einnehmen, während Inline-Elemente keinen Zeilenumbruch erzwingen.

2 In den folgenden Beschreibungen wird die Unterstützung durch die verschiedenen Browser in eckigen Klammern mit einem Kurzcode und der mindestens erforderlichen Versionsnummer angegeben. Die verwendeten Codes lauten: *C:* Google Chrome, *F:* Firefox, *IE:* Internet Explorer, *O:* Opera, *S:* Safari, *IOS*: iOS-Geräte mit Mobile Safari und *A*: Android-Browser.

`<header>`
: Definiert den Kopfbereich einer Seite oder eines Abschnitts. *[C5, F3.6, IE8, S4, O10]*

`<footer>`
: Definiert die Fußzeile einer Seite oder eines Abschnitts. *[C5, F3.6, IE8, S4, O10]*

`<nav>`
: Definiert einen Navigationsbereich einer Seite oder eines Abschnitts. *[C5, F3.6, IE8, S4, O10]*

`<section>`
: Definiert einen logischen Abschnitt einer Seite oder einer Gruppe von Inhalten. *[C5, F3.6, IE8, S4, O10]*

`<article>`
: Definiert einen Artikel oder ein in sich abgeschlossenes Inhaltselement. *[C5, F3.6, IE8, S4, O10]*

`<aside>`
: Definiert sekundäre oder ähnliche Inhalte. *[C5, F3.6, IE8, S4, O10]*

Benutzerdefinierte Datenattribute
: Mit dem data-Muster lassen sich zu beliebigen Elementen benutzerdefinierte Attribute hinzufügen. [Alle Browser unterstützen das Einlesen dieser Attribute über die JavaScript-Methode `getAttribute()`.]

`<meter>`
: Beschreibt einen Wert innerhalb eines Wertebereichs. *[C5, F3.5, S4, O10]*

`<progress>`
: Steuerelement, das den Fortschritt zu einem bestimmten Ziel hin in Echtzeit anzeigt. [Zum Zeitpunkt der Veröffentlichung nicht unterstützt.]

1 Einen Blog mit semantischem Markup neu definieren

In einem Blog finden Sie jede Menge Inhalte, die nach strukturiertem Markup schreien. Sie brauchen Kopfzeilen, Fußzeilen, mehrere Navigationsmöglichkeiten (Archive, Blogrolls und interne Links) und natürlich Artikel oder Beiträge. Basteln wir mit HTML5 ein Mock-up für die Startseite des Blogs von AwesomeCo, einem der großartigsten Unternehmen überhaupt.

In Abbildung 2.1 auf der nächsten Seite sehen Sie, worum es geht. Es handelt sich um eine typische Blog-Struktur, mit einer Hauptkopfzeile und einer horizontalen Navigation darunter. Im Hauptbereich erhält jeder Artikel eine Kopf- und eine Fußzeile. Artikel können außerdem einen Zitatkasten oder Sekundärinhalte enthalten. Zum Abschluss erhält die Seite eine Fußzeile für Kontakt- und Copyright-Informationen. An der Struktur an sich ist nichts außergewöhnlich. Allerdings bauen wir sie diesmal nicht aus einer Menge `div`-Tags auf, sondern verwenden spezielle Tags, um die jeweiligen Abschnitte zu beschreiben.

Wenn wir fertig sind, erhalten wir etwas, das in etwa wie Abbildung 2.2 auf Seite 28 aussieht.

Auf den Doctype kommt es an

Wir möchten die neuen Elemente von HTML5 verwenden. Deshalb müssen wir Browsern und Validierern mitteilen, welche Tags wir verwenden. Erstellen Sie eine neue Seite mit dem Namen `index.html`, und schreiben Sie die folgende einfache HTML5-Vorlage in die Datei.

html5newtags/index.html

```
1  <!DOCTYPE html>
2  <html lang="en-US">
3    <head>
4      <meta http-equiv="Content-Type" content="text/html; charset=utf-8" />
5      <title>AwesomeCo Blog</title>
6    </head>
7
8    <body>
9    </body>
10 </html>
```

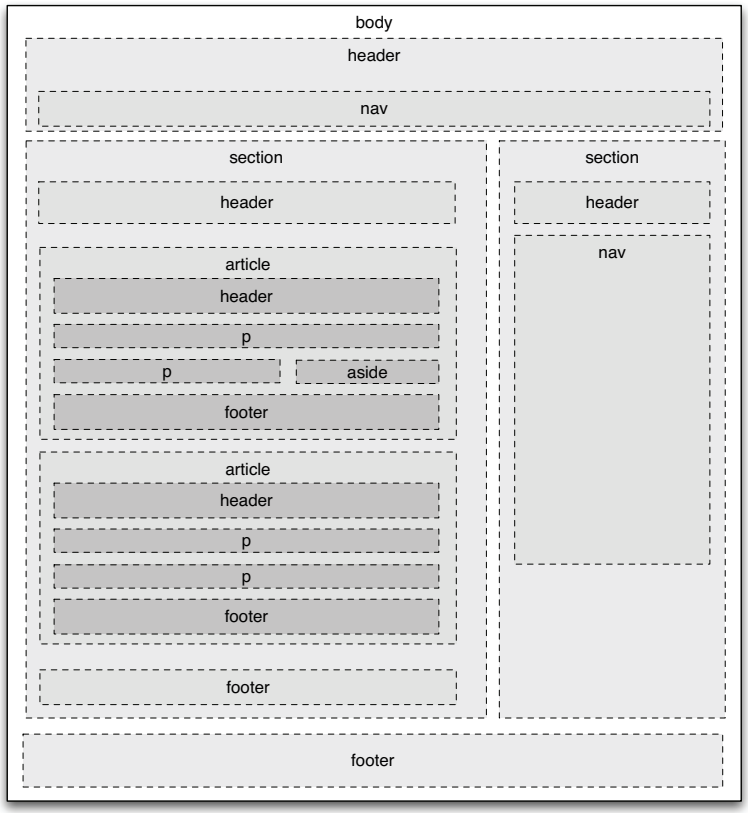

Abbildung 2.1: Die Blogstruktur bei Verwendung von semantischem HTML5-Markup

Sehen Sie sich mal den Doctype in Zeile 1 an. Mehr brauchen wir für den HTML5-Doctype nicht zu schreiben. Wenn Sie regelmäßig Webseiten erstellen, sind Sie wahrscheinlich eher mit diesen langen, schwer zu merkenden Doctypes für XHTML vertraut:

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
  "http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
```

Zum Vergleich noch einmal der HTML5-Doctype:

```
<!DOCTYPE HTML>
```

Das ist doch viel einfacher und viel leichter zu merken!

Ein Doctype hat zweierlei Aufgaben. Erstens hilft er Validierern festzustellen, welche Validierungsregeln für die Validierung des Codes herangezogen werden müssen. Außerdem zwingt der Doctype den Internet Explorer 6, 7 und 8, in den „Standards Mode“ zu wechseln, was von

entscheidender Bedeutung ist, wenn Sie Webseiten erstellen, die in allen Browsern funktionieren sollen. Der HTML5-Doctype erfüllt beide Anforderungen und wird *sogar vom Internet Explorer 6 erkannt*.

Abbildung 2.2: Das fertige Layout

Kopfzeilen

Kopfzeilen (nicht zu verwechseln mit Überschriften wie etwa `h1`, `h2` und `h3`) können alle möglichen Arten von Inhalten enthalten – vom Unternehmenslogo bis hin zum Suchfeld. Die Kopfzeile unseres Blogs soll zunächst nur den Titel des Blogs enthalten.

html5newtags/index.html

```
1  <header id="page_header">
2     <h1>AwesomeCo Blog!</h1>
3  </header>
```

Sie sind nicht auf eine Kopfzeile pro Seite beschränkt. Jeder einzelne Abschnitt oder Artikel kann eine eigene Kopfzeile enthalten. Da kann es hilfreich sein, Ihre Elemente mit dem `id`-Attribut eindeutig zu kennzeichnen, wie ich das in Zeile 1 getan habe. Mit einer eindeutigen ID sind Elemente mit CSS einfach zu gestalten oder mit JavaScript ausfindig zu machen.

Semantisches Markup

Semantisches Markup dient dazu, Ihren Inhalt zu beschreiben. Wenn Sie bereits seit mehreren Jahren Webseiten entwickeln, teilen Sie Ihre Seiten wahrscheinlich in verschiedene Abschnitte wie `header`, `footer` und `sidebar` auf. Dadurch können Sie die einzelnen Abschnitte der Seite leichter identifizieren, wenn Sie Stylesheets und andere Formatierungen darauf anwenden.

Semantisches Markup macht es Maschinen und Menschen gleichermaßen leicht, die Bedeutung und den Kontext des Inhalts zu verstehen. Die neuen Markup-Tags von HTML5 wie header und nav sollen Ihnen genau dabei helfen.

Fußzeilen

Das `footer`-Element definiert Fußzeileninformationen für ein Dokument oder einen angrenzenden Abschnitt. Fußzeilen auf Websites kennen Sie bereits. Üblicherweise stehen darin Informationen wie das Copyright-Datum oder darüber, wem die Website gehört. Der Spezifikation entsprechend können wir auch mehrere Fußzeilen in einem Dokument haben. Das bedeutet also, dass wir auch innerhalb unserer Blog-Artikel Fußzeilen verwenden können.

Für den Moment definieren wir einfach eine einfache Fußzeile für unsere Seite. Da wir mehr als eine Fußzeile verwenden können, geben wir dieser genauso wie der Kopfzeile eine ID. Dadurch können wir diese bestimmte Fußzeile eindeutig identifizieren, wenn wir das Element und seine Kinder stylen möchten.

html5newtags/index.html

```
<footer id="page_footer">
   <p>&copy; 2010 AwesomeCo.</p>
</footer>
```

Die Fußzeile enthält lediglich ein Copyright-Datum. Jedoch können Fußzeilen genau wie Kopfzeilen auch andere Elemente enthalten, wie etwa Navigationselemente.

Navigation

Die Navigation ist für den Erfolg einer Website entscheidend. Ihre Besucher werden nicht lange bleiben, wenn sie nicht das finden, was sie suchen. Daher ist es absolut sinnvoll, dass die Navigation ein eigenes HTML-Tag erhält.

Fügen wir einen Navigationsabschnitt in die Kopfzeile unseres Dokuments ein. Wir erstellen Links auf die Homepage des Blogs, das Archiv, eine Liste mit den Verfassern von Beiträgen zum Blog und einen Link auf die Kontaktseite.

html5newtags/index.html

```
<header id="page_header">
   <h1>AwesomeCo Blog!</h1>
   <nav>
      <ul>
         <li><a href="/">Latest Posts</a></li>
         <li><a href="archives">Archives</a></li>
         <li><a href="contributors">Contributors</a></li>
         <li><a href="contact">Contact Us</a></li>
      </ul>
   </nav>
</header>
```

Ihre Seite kann nicht nur mehrere Kopf- und Fußzeilen, sondern auch mehrere Navigationselemente enthalten. Die Navigation befindet sich häufig in der Kopfzeile und in der Fußzeile, die Sie nun explizit kennzeichnen können. Die Fußzeile unseres Blogs braucht Links auf die Homepage von AwesomeCo, eine Seite über das Unternehmen sowie Links auf die Nutzungsbedingungen und Datenschutzrichtlinien. Diese Links fügen wir in der Seite als eine weitere ungeordnete Liste im Element footer ein.

html5newtags/index.html

```
<footer id="page_footer">
  <p>&copy; 2010 AwesomeCo.</p>
  <nav>
    <ul>
      <li><a href="http://awesomeco.com/">Home</a></li>
      <li><a href="about">About</a></li>
      <li><a href="terms.html">Terms of Service</a></li>
      <li><a href="privacy.html">Privacy</a></li>
    </ul>
  </nav>
</footer>
```

Das Aussehen der beiden Navigationsleisten passen wir später mit CSS an. Achten Sie daher zunächst nicht so sehr auf die Optik. Die Aufgabe dieser neuen Elemente ist es, den Inhalt zu beschreiben, nicht aber, das Aussehen des Inhalt zu bestimmen.

Abschnitte und Artikel

Abschnitte sind die logischen Bereiche einer Seite. Deshalb gibt es nun das Element `section`, um das oft missbrauchte `div`-Tag bei der Beschreibung logischer Abschnitte einer Seite abzulösen.

html5newtags/index.html

```
<section id="posts">
</section>
```

Übertreiben Sie es aber nicht mit den Abschnitten. Setzen Sie sie ein, um Ihren Inhalt logisch zu gliedern. Hier haben wir einen Abschnitt erstellt, der alle Beiträge in einem Blog enthält. Jedoch soll nicht jeder Beitrag einen eigenen Abschnitt erhalten. Dafür gibt es ein besser geeignetes Tag.

Artikel

Das `article`-Tag ist das perfekte Element, um den Inhalt einer Webseite zu beschreiben. Mit den vielen Elementen auf einer Seite – Kopfzeilen, Fußzeilen, Navigationselemente, Werbung, Widgets, Blogrolls und Lesezeichen für soziale Medien – könnten Sie glatt vergessen, dass die Benutzer Ihre Website wegen des von Ihnen angebotenen Inhalts besuchen. Das `article`-Tag hilft Ihnen dabei, diesen Inhalt zu beschreiben.

Jeder unserer Artikel besteht aus einer Kopfzeile, dem eigentlichen Inhalt und einer Fußzeile. So definieren wir einen vollständigen Artikel:

html5newtags/index.html

```
<article class="post">
  <header>
    <h2>How Many Should We Put You Down For?</h2>
    <p>Posted by Brian on
      <time datetime="2010-10-01T14:39">October 1st, 2010 at 2:39PM</time>
    </p>
  </header>
  <p>
    The first big rule in sales is that if the person leaves empty-handed,
    they're likely not going to come back. That's why you have to be
    somewhat aggressive when you're working with a customer, but you have
    to make sure you don't overdo it and scare them away.
  </p>
```

```
    <p>
      One way you can keep a conversation going is to avoid asking questions
      that have yes or no answers. For example, if you're selling a service
      plan, don't ever ask &amp;quot;Are you interested in our 3 or 5 year
      service plan?&amp;quot; Instead, ask &amp;quot;Are you interested in the 3
      year service plan or the 5 year plan, which is a better value?&amp;quot;
      At first glance, they appear to be asking the same thing, and while
      a customer can still opt out, it's harder for them to opt out of
      the second question because they have to say more than just &amp;quot;no.&amp;quot;
    </p>
    <footer>
      <p><a href="comments"><i>25 Comments</i></a>...</p>
    </footer>
  </article>
```


Joe fragt ...

Was ist der Unterschied zwischen Artikeln und Abschnitten?

Stellen Sie sich einen Abschnitt als logischen Teil eines Dokuments vor. Einen Artikel können Sie sich als den eigentlichen Inhalt vorstellen, wie etwa einen Artikel in einer Zeitschrift, einen Beitrag in einem Blog oder eine aktuelle Meldung.

Die neuen Tags beschreiben genau den Inhalt, den sie enthalten. Abschnitte können mehrere Artikel enthalten, und Artikel können aus mehreren Abschnitten bestehen. Ein Abschnitt ist wie der Sportteil einer Zeitung. Der Sportteil enthält viele Artikel. Jeder dieser Artikel kann wiederum aus mehreren eigenständigen Abschnitten bestehen. Bestimmte Abschnitte wie Kopfzeilen und Fußzeilen erhalten eigene Tags. Ein Abschnitt ist ein allgemein gehalteneres Element, mit dem Sie andere Elemente logisch gruppieren können.

Bei semantischem Markup geht es darum, die Bedeutung Ihres Inhalts zu vermitteln.

In einem Artikel können wir die Elemente `header` und `footer` verwenden, wodurch es wesentlich einfacher wird, diese bestimmten Abschnitte zu beschreiben. Wir können unseren Artikel auch mit dem Element `section` in mehrere Abschnitte unterteilen.

Das aside-Tag und Seitenleisten

Manche Inhalte ergänzen etwas zum eigentlichen Inhalt, wie etwa Zitatkästen, Diagramme, weiterführende Gedanken oder entsprechende Links. Mit dem neuen `aside`-Tag können Sie diese Elemente kenntlich machen.

html5newtags/index.html

```
<aside>
  <p>
    &quot;Never give someone a chance to say no when selling your
    product.&quot;
  </p>
</aside>
```

Das Zitat schreiben wir in ein aside-Element. Das aside-Element verschachteln wir wiederum in den Artikel und platzieren es damit nahe beim zugehörigen Inhalt.

So sieht unser vollständiger Abschnitt mit dem aside-Element aus:

html5newtags/index.html

```
<section id="posts">
  <article class="post">
    <header>
      <h2>How Many Should We Put You Down For?</h2>
      <p>Posted by Brian on
        <time datetime="2010-10-01T14:39">October 1st, 2010 at 2:39PM</time>
      </p>
    </header>

    <aside>
      <p>
        &quot;Never give someone a chance to say no when selling your
        product.&quot;
      </p>
    </aside>
    <p>
      The first big rule in sales is that if the person leaves empty-handed,
      they're likely not going to come back. That's why you have to be
      somewhat aggressive when you're working with a customer, but you have
      to make sure you don't overdo it and scare them away.
    </p>
    <p>
     One way you can keep a conversation going is to avoid asking questions
     that have yes or no answers. For example, if you're selling a service
     plan, don't ever ask &quot;Are you interested in our 3 or 5 year service
     plan?&quot; Instead, ask &quot;Are you interested in the 3
     year service plan or the 5 year plan, which is a better value?&quot;
     At first glance, they appear to be asking the same thing, and while
     a customer can still opt out, it's harder for them to opt out of
     the second question because they have to say more than just
     &quot;no.&quot;
    </p>
    <footer>
      <p><a href="comments"><i>25 Comments</i></a>...</p>
    </footer>
  </article>
</section>
```

Nun müssen wir nur noch den Abschnitt für die Seitenleiste einfügen.

aside-Elemente sind keine Seitenleisten

Unser Blog enthält eine rechte Seitenleiste mit den Archiven für den Blog. Falls Sie jetzt glauben, wir könnten das `aside`-Tag verwenden, um die Seitenleiste unseres Blogs zu definieren, liegen Sie leider falsch. Sie *könnten* das zwar so machen, aber es widerspricht dem Sinn der Spezifikation. `aside` wurde entwickelt, um die Inhalte anzuzeigen, die zu einem Artikel gehören: die richtige Stelle also, um entsprechende Links, ein Glossar oder einen Zitatkasten anzuzeigen.

Als Markup für unsere Seitenleiste mit der Liste der bisherigen Archive verwenden wir ein weiteres `section`-Tag sowie ein `nav`-Tag:

html5newtags/index.html

```
<section id="sidebar">
  <nav>
    <h3>Archives</h3>
    <ul>
      <li><a href="2010/10">October 2010</a></li>
      <li><a href="2010/09">September 2010</a></li>
      <li><a href="2010/08">August 2010</a></li>
      <li><a href="2010/07">July 2010</a></li>
      <li><a href="2010/06">June 2010</a></li>
      <li><a href="2010/05">May 2010</a></li>
      <li><a href="2010/04">April 2010</a></li>
      <li><a href="2010/03">March 2010</a></li>
      <li><a href="2010/02">February 2010</a></li>
      <li><a href="2010/01">January 2010</a></li>
    </ul>
  </nav>
</section>
```

Damit haben wir schon unsere Blog-Struktur. Jetzt können wir damit beginnen, die neuen Elemente zu stylen.

Styling

Genau wie auf `div`-Tags können wir auch auf die neuen Elemente Stilregeln anwenden. Zunächst erstellen wir ein neues Stylesheet mit dem Namen `style.css`. Anschließend verknüpfen wir es mit unserem HTML-Dokument, indem wir im Header einen Link auf das Stylesheet einfügen:

html5newtags/index.html

```
<link rel="stylesheet" href="style.css" type="text/css">
```

Als Erstes zentrieren wir den Inhalt der Seite und legen einige grundlegenden Schriftarten fest:

html5newtags/style.css

```
body{
  width:960px;
  margin:15px auto;
  font-family: Arial, "MS Trebuchet", sans-serif;
}

p{
  margin:0 0 20px 0;
}

p, li{
  line-height:20px;
}
```

Dann definieren wir die Breite der Kopfzeile:

html5newtags/style.css

```
header#page_header{
  width:100%;
}
```

Für die Navigationslinks wandeln wir die Listen mit Aufzählungszeichen in horizontale Navigationsleisten um:

html5newtags/style.css

```
header#page_header nav ul, #page_footer nav ul{
  list-style: none;
  margin: 0;
  padding: 0;
}
#page_header nav ul li, footer#page_footer nav ul li{
  padding:0;
  margin: 0 20px 0 0;
  display:inline;
}
```

Der Abschnitt `posts` muss gefloatet werden und eine feste Breite erhalten. Außerdem müssen wir auch den Callout im Artikel floaten. Bei der Gelegenheit vergrößern wir gleich noch die Schrift für den Callout.

html5newtags/style.css

```
section#posts{
  float: left;
  width: 74%;
}

section#posts aside{
  float: right;
  width: 35%;
```

Messwerte und Fortschrittsbalken

Wenn Sie ein Spendenbarometer oder einen Fortschrittsbalken für den Upload in einer Webanwendung implementieren möchten, sollten Sie einen Blick auf die Elemente `meter` und `progress` werfen, die mit HTML5 eingeführt wurden.

Mit dem Element meter können wir einen festen Punkt auf einer Messskala mit einem Mindest- und einem Höchstwert semantisch beschreiben. Damit Ihr Messwert mit der Spezifikation harmoniert, sollten Sie ihn nicht für beliebige Minimal- oder Maximalwerte wie Höhe oder Gewicht verwenden, außer Sie haben einen bestimmten Grenzwert festgelegt. Wenn wir beispielsweise auf einer Website zum Sammeln von Spenden zeigen möchten, wie weit wir noch von dem Ziel $5.000 entfernt sind, lässt sich das so beschreiben:

html5_meter/index.html

```
<section id="pledge">
  <header>
    <h3>Our Fundraising Goal</h3>
  </header>
  <meter title="USD" id="pledge_goal"
         value="2500" min="0" max="5000">
    $2500.00
  </meter>
  <p>Help us reach our goal of $5000!</p>
</section>
```

Das Element progress ist einem Messwert sehr ähnlich, wurde aber dafür entwickelt, einen aktiven Fortschritt darzustellen, zum Beispiel beim Hochladen einer Datei. Ein Messwert zeigt dagegen eine Messung an, die nicht mehr verändert wird, wie etwa einen Schnappschuss des Speicherplatzes, der auf einem Server einem bestimmten Benutzer noch zur Verfügung steht. Das Markup für einen Fortschrittsbalken ist dem eines `meter`-Elements aber sehr ähnlich:

html5_meter/progress.html

```
<progress id="progressbar" max=100><span>0</span>%</progress>
```

Die Elemente `meter` und `progress` lassen sich noch in keinem Browser darstellen. Sie können jedoch mit JavaScript die Werte aus dem `meter`-Element auslesen und selbst abbilden, das Element meter oder progress also dafür verwenden, die Daten semantisch zu beschreiben. Ein Beispiel dafür finden Sie in den Beispieldateien dieses Buchs für das `meter`-Element.

```
  margin-left: 5%;
  font-size: 20px;
  line-height: 40px;
}
```

Außerdem müssen wir die Seitenleiste floaten und eine feste Breite dafür festlegen:

html5newtags/style.css

```
section#sidebar{
  float: left;
  width: 25%;
}
```

Und wir müssen die Fußzeile definieren. Wir löschen die Floats in der Fußzeile, sodass sie im unteren Teil der Seite zu liegen kommt.

html5newtags/style.css

```
footer#page_footer{
  clear: both;
  width: 100%;
  display: block;
  text-align: center;
}
```

Das sind nur die wichtigsten Stilregeln in Kürze. Ich bin mir sicher, dass Sie das alles noch viel, viel besser aussehen lassen können.

Ausweichlösung

Das funktioniert bereits alles wunderbar in Firefox, Chrome und Safari. Allerdings werden die Leute im Management nicht allzu glücklich sein, wenn sie das Chaos sehen, das der Internet Explorer aus unserer Seite macht. Der Inhalt wird zwar wunderbar dargestellt. Aber da der IE diese Elemente nicht kennt, kann er auch keine Stilregeln darauf anwenden. Die Seite erinnert deshalb eher an die Mitte der Neunziger-Jahre.

Mit dem IE können wir diese Elemente nur stylen, wenn wir sie mit JavaScript als Teil des Dokuments definieren. Wie sich herausstellt, ist das wirklich einfach. Wir fügen den Code in den Abschnitt `head` der Seite ein, sodass er ausgeführt wird, bevor der Browser irgendwelche Elemente rendert. Außerdem schreiben wir den Code in einen *bedingten Kommentar* – eine besondere Art von Kommentar, die nur der Internet Explorer verarbeitet.

html5newtags/index.html

```
<!--[if lt IE 9]>
<script type="text/javascript">
  document.createElement("nav");
  document.createElement("header");
  document.createElement("footer");
  document.createElement("section");
  document.createElement("aside");
  document.createElement("article");
</script>
<![endif]-->
```

Solche Kommentare gelten nur für Versionen des Internet Explorer, die älter als Version 9.0 sind. Wenn wir die Seite jetzt noch einmal laden, wird sie richtig dargestellt.

Allerdings machen wir uns dadurch von JavaScript abhängig. Das sollten Sie bedenken. Für den verbesserten Aufbau und die Lesbarkeit des Dokuments lohnt sich das durchaus. Außerdem gibt es keine Probleme bezüglich der Barrierefreiheit, da der Inhalt nach wie vor angezeigt und von einem Bildschirmlesegerät vorgelesen werden kann. Die Darstellung wirkt lediglich hoffnungslos altmodisch für Benutzer, die JavaScript absichtlich deaktiviert haben.

Mit diesem Ansatz können Sie auch wunderbar zusätzliche Elemente unterstützen oder verstehen, wie Sie eine entsprechende Unterstützung schreiben können. Die brillante Lösung HTMLShiv[3] von Remy Sharp treibt die Dinge noch wesentlich weiter, ist aber wahrscheinlich eher dann angebracht, wenn Sie eine Ausweichlösung für viele zusätzliche Elemente integrieren möchten.

3 http://code.google.com/p/html5shiv/

2 Pop-up-Fenster mit benutzerdefinierten Datenattributen

Wenn Sie schon mal eine Webanwendung entwickelt haben, die über JavaScript Dokumentinformationen ausliest, dann wissen Sie, dass dafür manchmal ein bisschen Hackerei und Parsen erforderlich sind. Letztendlich fügen Sie zusätzliche Informationen in Event-Handler ein oder missbrauchen die Attribute `rel` oder `class`, damit es funktioniert. Dank der Einführung benutzerdefinierter Datenattribute sind diese Zeiten nun jedoch vorbei.

Benutzerdefinierte Datenattribute beginnen mit dem Präfix `data-` und werden von Validierern für HTML5-Dokumente ignoriert. Sie können benutzerdefinierte Datenattribute zu jedem beliebigen Element hinzufügen – ganz egal, ob es sich um Metadaten zu einem Foto, Längen- und Breitengradkoordinaten oder, wie in diesen Tipp, um die Maße eines Pop-up-Fensters handelt. Das Beste daran ist, dass Sie benutzerdefinierte Datenattribute schon jetzt in beinahe jedem Browser verwenden und sie ganz leicht mit JavaScript ausgelesen können.

Das Verhalten vom Inhalt trennen oder warum onclick böse ist

Im Laufe der Jahre haben Pop-up-Fenster einen schlechten Ruf bekommen – oft auch zu Recht. Sie werden eingesetzt, um Werbung anzuzeigen, um nichts ahnende Websurfer dazu zu bringen, Spyware oder Viren zu installieren, oder sogar, um persönliche Informationen weiterzugeben, die später verkauft werden. Es überrascht daher nicht weiter, dass die meisten Browser über einen Pop-up-Blocker verfügen.

Pop-ups an sich sind aber gar nichts Schlechtes. Entwickler von Webanwendungen verwenden Pop-up-Fenster häufig, um die Online-Hilfe, zusätzliche Optionen oder andere wichtige Funktionen der Benutzeroberfläche anzuzeigen. Damit Pop-ups weniger nerven, müssen wir sie lediglich entsprechend unaufdringlich implementieren. Wenn Sie sich die Seite der Personalabteilung von AwesomeCo ansehen, finden Sie mehrere Links, über die Richtlinien in Pop-up-Fenstern angezeigt werden. Die meisten sehen so aus:

html5_popups_with_custom_data/original_example_1.html

```
<a href='#'
  onclick="window.open('holiday_pay.html',WinName,'width=300,height=300);">
  Holiday pay
</a>
```

Das ist ein gebräuchlicher Ansatz, um Links zum Öffnen von Pop-ups zu erstellen. Genau genommen ist das der Ansatz, den JavaScript-Neulinge lernen, um Pop-up-Fenster zu öffnen. Diese Methode bringt jedoch einige Probleme mit sich, derer wir uns annehmen sollten, bevor wir weitermachen.

Mehr Barrierefreiheit

Das Ziel des Links wurde nicht festgelegt! Wenn JavaScript deaktiviert ist, führt der Link den Benutzer nicht zur entsprechenden Seite. Das ist ein großes Problem, um das wir uns sofort kümmern müssen. Lassen Sie *niemals* das Attribut `href` aus, und vergeben Sie dafür *unter keinen Umständen* einen Wert wie im vorherigen Beispiel. Geben Sie die Adresse der Ressource an, die normalerweise angezeigt werden soll.

html5_popups_with_custom_data/original_example_2.html

```
<a href='holiday_pay.html'
  onclick="window.open(this.href,WinName,'width=300,height=300);">
  Holiday pay
</a>
```

Der JavaScript-Code liest dann seinerseits das Ziel des Links aus dem `href`-Attribut des verknüpften Elements aus.

Als erster Schritt zu einer barrierefreien Seite muss die gesamte Funktionalität auch *ohne* JavaScript gewährleistet sein.

onclick beseitigen

Trennen Sie das Verhalten vom Inhalt, genauso wie Sie auch die Darstellungsinformationen in separaten und verknüpften Stylesheets ablegen. Auf den ersten Blick ist `onclick` einfach zu verwenden. Aber wenn Sie sich eine Seite mit 50 Links vorstellen, werden Sie verstehen, dass die `onclick`-Methode schnell außer Kontrolle gerät. Sie wiederholen dasselbe JavaScript immer und immer wieder. Und wenn Sie diesen Code serverseitig generieren, erhöhen Sie lediglich die Anzahl von JavaScript-Events und machen das so entstandene HTML viel umfangreicher als nötig.

Weisen Sie stattdessen jedem Anker auf der Seite eine entsprechende Klasse zu:

html5_popups_with_custom_data/original_example_3.html

```
<a href="holiday_pay" class="popup">Holiday Pay</a>
```

html5_popups_with_custom_data/original_example_3.html

```
var links = $("a.popup");

links.click(function(event){
    event.preventDefault();
    window.open($(this).attr('href'));
});
```

Mit einem jQuery-Selektor erfassen wir alle Elemente der Klasse `popup`. Anschließend fügen wir einen Listener zum `click`-Event der jeweiligen Elemente hinzu. Der an die `click`-Methode übergebene Code wird ausgeführt, wenn jemand auf den Link klickt. Die Methode `preventDefault` unterbindet dabei das standardmäßige Verhalten für das `click`-Event. In diesem Fall verhindert sie, dass der Browser dem Link folgt und eine neue Seite anzeigt.

Was wir dabei im Vergleich zum ursprünglichen Beispiel verlieren, sind allerdings die Informationen zu Größe und Position des Fensters. Wir möchten aber, dass auch ein Designer, der sich nicht allzu gut mit JavaScript auskennt, für jeden Link die Maße des Fensters festlegen kann.

Benutzerdefinierte Datenattribute zu Hilfe!

Bei der Erstellung von JavaScript-gestützten Anwendungen kommt dieses Problem häufig vor. Wie wir gesehen haben, möchten wir die gewünschte Höhe und Breite im Code ablegen. Der Ansatz mit `onclick` hat aber einige Nachteile. Stattdessen können wir diese Informationen als Attribute des jeweiligen Elements einbetten. Hierfür müssen wir lediglich den Link folgendermaßen aufbauen:

html5_popups_with_custom_data/popup.html

```
<a href="help/holiday_pay.html"
   data-width="600"
   data-height="400"
   title="Holiday Pay"
   class="popup">Holiday pay</a>
```

Jetzt ändern wir einfach das `click`-Event, das wir gerade geschrieben haben, so, dass es die Optionen aus den benutzerdefinierten Datenattributen des Links ausliest und an die Methode `window.open` übergibt:

html5_popups_with_custom_data/popup.html

```
$(function(){
  $(".popup").click(function(event){
    event.preventDefault();
    var href = $(this).attr("href");
    var width = $(this).attr("data-width");
    var height = $(this).attr("data-height");
    var popup = window.open(href,"popup",
      "height=" + height +",width=" + width + "");
   });
 });
```

Das ist alles! Der Link wird nun in einem neuen Fenster geöffnet.

> **Eine kleine Warnung**
>
> In diesem Beispiel haben wir benutzerdefinierte Datenattribute verwendet, um zusätzliche Informationen an das clientseitige Skript zu übergeben. Das ist ein cleverer Ansatz, um ein bestimmtes Problem zu lösen und eine mögliche Verwendung dieser Attribute zu zeigen. Zwar wird unser Markup dadurch mit Darstellungsinformationen vermischt, dafür sehen Sie aber, wie leicht es ist, mit JavaScript in Ihre Seite eingebettete Werte auszulesen.

Ausweichlösung

Diese Attribute funktionieren schon jetzt auch in älteren Browsern, solange diese JavaScript unterstützen. Der Browser hat keine Schwierigkeiten mit benutzerdefinierten Datenattributen. Außerdem ist Ihr Dokument auch valide, da Sie den Doctype HTML5 verwenden und Attribute, die mit `data-` beginnen, daher ignoriert werden.

Die Zukunft

Sobald diese neuen Tags und Attribute auf breiter Basis unterstützt werden, können wir einige interessante Dinge damit anstellen. So können wir beispielsweise die Navigation und Fußzeilen von Artikeln kennzeichnen und in Stylesheets für den Druck ganz einfach deaktivieren.

```
nav, article>footer{display:none}
```

Mit Skriptsprachen können wir schnell alle Artikel auf einer Seite oder einer Website erkennen. Und noch besser: Wir zeichnen Inhalte mit Tags aus, die diese korrekt beschreiben, und können so bessere Stylesheets und besseres JavaScript schreiben.

Benutzerdefinierte Datenattribute bieten Entwicklern die Flexibilität, alle Arten von Informationen in ihr Markup einzubetten. Wir verwenden sie wieder in Kapitel 6, *Zeichnen mit dem canvas-Element*, auf Seite 113.

In Verbindung mit JavaScript können Sie mit benutzerdefinierten Datenattributen festlegen, ob ein `form`-Tag über Ajax übertragen werden soll. Dafür müssen Sie lediglich einem beliebigen `form`-Tag `data-remote=true` zuweisen – so macht es zum Beispiel das *Ruby on Rails*-Framework.

Außerdem können Sie damit Datum und Uhrzeit in der Zeitzone Ihrer Benutzer anzeigen, auch wenn sich die Seite im Cache befindet. Dazu geben Sie das Datum in der HTML-Seite einfach als UTC an und konvertieren es clientseitig in das lokale Format der Benutzer. Mit diesen Attributen können Sie real greifbare und brauchbare Daten in Ihre Seiten einbetten und sich darauf verlassen, dass immer mehr Frameworks und Bibliotheken diese nutzen. Ich bin mir sicher, Sie finden dafür jede Menge großartiger Anwendungsmöglichkeiten in Ihrer eigenen Arbeit.

Und außerdem können wir damit der Divitis ein für alle Mal den Garaus machen!