DAS NEU ERFUNDENE WEB: HTML 5

EVOLUTION STATT REVOLUTION

Der nächste Web-Standard naht: HTML 5 bietet viele neue Funktionen. Tabellen, Kalenderblätter und direkt eingebettete Videos.

VON **SEBASTIAN NIEMANN**

Seit Ende der 1990er Jahre hat sich das Internet sehr verändert. Aus vormals eher statischen Webseiten sind komplexe Applikationen geworden. Die Webspezifikationen HTML 4 und XHTML 1 reichen für das heutige Web 2.0 gerade noch aus. Sie sind jedoch, gemessen an den Erwartungen an das künftige Netz, in die Jahre gekommen. Das World Wide Web Consortium (W3C) hatte sich mit der Entwicklung von XHTML 2 der Sache angenommen. Leider ist es nicht zu XHTML 1 kompatibel. Die *Web Hypertext Application Technology Working Group* (WHATWG) – eine Gruppe von Browserherstellern – war damit nicht einverstanden und hat in Konkurrenz HTML 5 forciert. Das W3C musste einlenken und ließ XHTML 2 schließlich fallen.

Browserunterstützung

Da die Initiative für HTML 5 seitens der Browserhersteller geführt wurde, ist zu erwarten, dass es sukzessive keine grundlegenden Probleme mit den Browsern der neuesten und nachfolgenden Generationen geben wird. Dies haben auf Anfrage des PC Magazins Opera, Mozilla (Firefox) und sogar auch Microsoft (Internet Explorer) bestätigt, obwohl Microsoft gar nicht zur Gruppe der WHATWG gehört. Der Vollständigkeit halber seien hier auch Apples Safari und Googles Chrome genannt. Das Gute an HTML 5 ist, dass es größtenteils abwärts kompatibel ist. Schreiben wir also in HTML 5, so muss ein älterer, nicht kompatibler Browser die Webseite trotzdem darstellen können. Das führt uns schon zur ersten Zeile des Source-Codes, die dann nur noch schlicht *‹!DOCTYPE html›* heißen kann.

Semantik

Strukturell kann eine HTML 5- Webseite übersichtlicher geschrieben werden. Bekannt sind bislang sehr verschachtelte *‹div›*-Elemente, um das Layoutraster einer Homepage von außen nach innen aufzubauen.

Mittels persönlicher Vorlieben des Webautors bekommen die semantisch bedeutungslosen *‹div›*-Elemente per *id* und *class* Namen und bestimmte Eigenschaften zugeschrieben. Damit man in dem hierarchisch sortierten DIV-Chaos selbst noch durchblickt, werden gerne noch zusätzliche Kommentarzeilen eingefügt. Beispiel für eine unsemantische Notation:

```
<body>
  <div id="kopfbereich">
    <div id="logo">Logo</div>
<div id="hauptnavi">Hauptmenü</div>
  </div>
```

```
  <div id="content">
    <h1>Überschrift 1</h1>
    <div class="teil1">
 <!-- Content 1 -->
      <h2>Überschrift 2</h2>
      Mein Text...
    </div>
 <div class="teil2">
      <h2>Überschrift 2</h2>
      <!-- Content 2 -->
 Mein Text...
    </div>
  </div>
  <div id="marginalspalterechts">
    <div id="subnavi">Untermenü</div>
  </div>
  <div id="footer">Fußbereich der Website</div>
</body>
```

HTML 5 bereinigt dieses Konstrukt und stellt eine klare (semantische) Sprache zur Verfügung, die der zukünftige Standard sein wird. Das gleiche Konstrukt wie oben sieht nun folgendermaßen aus:

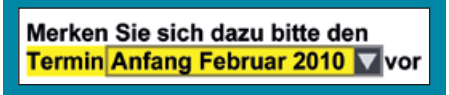
```
<body>
  <header>
    <div id="logo">Logo</div>
    <nav>Hauptmenü</nav>
  </header>

  <article>
    <h1>Überschrift 1</h1>
    <section>
      <h2>Überschrift 2</h2>
          Mein Text...
    </section>
    <section>
      <h2> Überschrift 2</h2>
          Mein Text...
    </section>
  </article>

  <aside>
    <nav>Untermenü in der Marginalspalte</nav>
  </aside>

  <footer>Fußbereich der Website</footer>
</body>
```

Die Elemente *header, nav, article, section, aside* und *footer* sind Sektionselemente. Sie fassen jeweils zusammengehörige Abschnitte bzw. für Webseiten typische Komponenten semantisch sinnvoll zusammen. Für das individuelle Layout sind natürlich Kennzeichnungen für die section-Elemente via id und class auch weiterhin notwendig, soll sich die semantische Struktur auch visuell im Browser abbilden.

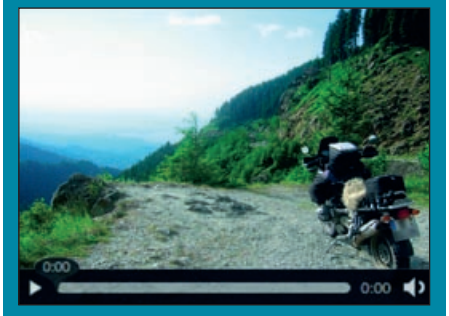
Ein Einwand mag hierbei sein, dass die neuen Elemente deshalb im Grunde genauso unspezifisch sind, wie die *‹span›*- und *‹div›*-Elemente. Allerdings werden diese semantisch bedeutungslosen Elemente zukünftig weniger werden – zugunsten einer noch kompatibleren, zukunftssicheren und geräteunabhängigeren Auszeichnungssprache HTML 5.

Der Vorteil des semantischen Ansatzes vollzieht sich auch herunter bis auf die Textebene. Eine interoperable Anwendung ist z.B. das *‹time›*-Element, das in Interaktion mit einer Kalenderfunktion des Computers Zeitangaben maschinenlesbar macht. Ein Beispiel:

```
<article>Merken Sie sich dazu bitte
den Termin <time datetime="2010-02-
09">Anfang Februar 2010</time> vor.</
article>
```



Der HTML-Autor hat in diesem Beispiel die Elemente <mark> und <time> miteinander kombiniert.



Firefox 3.5 stellt das <video>-Element mit einer neuen, nur temporär eingeblendeten Steuerleiste dar.

Bei Klick auf dieses Datum öffnet sich im Idealfall der persönliche digitale Terminplaner und man kann sich so ganz einfach den 10. Februar 2010 vormerken. In Kombination mit dem *‹mark›*-Element wird das Textbeispiel, wie mit einem Textmarker, nun einfach hervorgehoben:

```
<article>Merken Sie sich dazu bitte den
<mark>Termin <time datetime="2010-02-
09">Anfang Februar 2010</time></mark>
vor.</article>.
```

Auch in HTML eingebettete Bilder lassen sich nun semantisch korrekt *(‹figure›)* mit Bildunterschriften *(‹legend›)* auszeichnen:

```
<figure>
<img src="meinbild.jpg" alt="mein
Bild"/>
<legend>Meine Bildunterschrift<legend>
</figure>
```

Ein Browser mit computergesteuerter synthetischer Sprachausgabe oder einer angeschlossenen Brailletastatur für blinde Menschen könnte einfach per CSS angewiesen werden, Bilder global erst gar nicht darzustellen. Eine Voreinstellung im Browser gibt es hierfür allerdings bereits seit der Frühzeit der Internet-Browser. Nur wäre mit HTML 5 die semantisch korrekte Bildauszeichnung verlässlicher, flexibler und vor allem eigenständig.

Multimedia

In Anbetracht der inflationär gestiegenen Multimedia-Inhalte – Audiodateien und vor allem Videos – kommt HTML 5 einen großen Schritt entgegen. Bislang ist es umständlich multimediale Inhalte in HTML einzubinden, damit sie browser- und plattformübergreifend zur Verfügung stehen. Das proprietäre Dateiformat Flash sorgte bislang für erfolgversprechende Abhilfe, wie es auf YouTube millionenfach bewiesen wird. Flash-Formate benötigen allerdings viel Rechenleistung. Die Performance des gesamten Computers leidet, trotz Taktraten des Prozessors im Gigahertz-Bereich. Zudem sind zusätzlich Java-Scripte erforderlich, um eine reibungslose Darstellung eines Videos auf jedem Browser zu gewährleisten. Mit den HTML 5-Elementen *‹video›* und *‹audio›* entledigt man sich der komplexen Materie und auch notwendiger Plug-ins. Ein Code-Beispiel für Multimedia-Inhalte:

```
<video src="meinfilm.ogv" type="video/
ogg; codecs='theora, vorbis'"
controls="controls"></video>
<audio src="meinaudio.oga" type="audio/
ogg; codecs='vorbis'"></audio>
```

Der Browser bringt die Steuerelemente von Haus aus mit. Man muss sie nur via *controls* ansprechen. Wie im Beispiel ersichtlich, sind hier die Dateiformate *.ogv* (und *.ogg*) und *.oga* eingebettet. Das sind freie Videoformate, die sich abseits der ebenfalls proprietären und bislang typischen Multimediaformate von Microsoft und Apple ihren Weg suchen werden. So bleibt es vorerst dabei, im Zweifelsfall verschiedene Dateiformate anzubieten oder eine Fallback-Lösung für Computer, die keines der beiden Beispielformate unterstützen:

```
<video controls>
  <source src="meinfilm.ogg"
type="video/ogg" />
  <source src="meinfilm.mp4"
type='video/mp4';codecs="mp4v.20.240"'
/>
  Ihr Browser unterstützt das Element
„video" nicht oder beide Formate
  (mp4, ogg) der Filmdateien sind un-
bekannt.
</video>
```

Formulare

Mit HTML 5 werden gerade bei Formularen einige Auswertungen bereits im Browser abgefangen. Das bedeutet weniger Arbeit für den Programmierer und den in der Regel Form-Scripte evaluierenden Server. Sie haben dadurch eine unmittelbarere Steuerkontrolle der Formfelder. Sie können nun Formfelder genauer spezifizieren. ‹type="email"› erfragt eine E-Mail-Adresse. Praktisch sind in dem Zusammenhang auch weiter ‹type›-Attribute, wie z.B. url (für Webadressen), number (für Zahlen), date für die Datumseingabe, search für das Formfeld Suche – um nur einige zu nennen.

Eine Überprüfung durch den Browser erfolgt übrigens auf semantischer Grundlage: Beinhaltet die Webadresse gültige Parameter, also *http://www.MeineURL.de?* Ist die E-Mail mit *name@url.de* vollständig und richtig ausgezeichnet, wurde also z.B. das @-Zeichnen nicht vergessen? Sind bei number wirklich nur Zahlen eingegeben worden? Die Fehlermeldung bei falsch ausgefüllten Formfeldern generiert der Browser auch gleich selbst.

Interessant ist die meist direkt im Suchfenster des Browsers implementierte und landläufig als *Google-Suggest* bezeichnete Technik, die man nun selbst direkt auf der Homepage nutzen könnte.

Bei Klick in das Suchfeld werden automatisch vordefinierte und empfehlenswerte Begriffe vorgeschlagen:

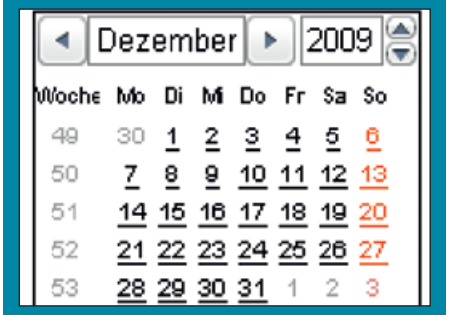
```
<label for="datalist">Suche nach…</
label>
<input id="datalist" type="search"
name="name" list="mylist" />
<datalist id="mylist">
<option value="Europa" />
<option value="Asien" />
<option value="Afrika" />
</datalist>
```

Besondere Beachtung sollte man auch den *‹type›*-Attributen *date*, *month*, oder *week* schenken. Hier klappt ein kompletter Kalender auf, den wir so in der Regel aus Online-Flug- und Reisebuchungen kennen. Umfangreiche CSS-Formatierungen und Java-Scripte entfallen.

```
<input type="date" name="date">
```

Abschließend sollte noch das Attribut *‹required›* genannt werden, das den Cursor nach dem Laden der Seite direkt in das vom Webautor gewünschte Formfeld setzt. Außerdem ermöglicht *‹pattern›* fortgeschrittenen Webautoren, die bereits Erfahrungen in Javascript oder PHP gesammelt haben, das individuelle Anlegen einer eigener Formfeld-Logik, die bei der Eingabe erfüllt werden muss.

<search> lässt sich mit <datalist> um eine Vorschlagsfunktion erweitern, wie von Google-Suggest bekannt (hier: in Opera 10).



Einen hübschen Kalender erzeugen die Browser mit HTML 5 von alleine mit dem Element <date> (Hier: Opera 10 auf MacOsX).

```
<p>Geben Sie nun Ihren Wert ein:
<input type="text"
name="speziellerWert" pattern="meine
Bedingungen für dieses Textfeld" /></p>
```

TIPP

Wer mit den neuen Form-Attributen experimentieren möchte, sollte möglichst Opera ab Version 10.10 als Entwicklungsumgebung nutzen. Die neuen Features sind vielversprechend, schlank und viele Java-Scripte werden obsolet. Sie sind bislang aber nur teilweise und unzureichend in den aktuellen Browsern implementiert. Eine Besserung ist nach Aussagen der Browserhersteller, wie eingangs erwähnt, offensichtlich stückchenweise zu erwarten.

Canvas

Wir gelangen nun zu einem erwähnenswerten Element, nämlich *‹canvas›*. Es stellt – zunächst rein aus technischer Warte – eine zwei- oder sogar dreidimensionale Zeichenfläche zur Verfügung.

```
<canvas width="200" height="200"
id="canvas" />
```

In Kombination mit Javascript wird dieses Element angesprochen und mit Formen und Linien gefüllt. Diesen können Farben, Transparenz und Bewegungen zugewiesen werden. Scheinbar doppeln sich die Ansätze für Canvas mit dem bekannten Format Scalable Vektor Graphics (SVG), das sich aber nie wirklich durchsetzen konnte. Es ist eine eigenständige Technologie, Canvas integraler Bestandteil von HTML 5. Was man mittels Canvas und Javascript Sinnvolles im Internet anfangen kann, wird die Zeit zeigen. Denkbar sind z.B. Datenauswertungen in Form von Diagrammen. Auch Animiertes ist denkbar. Erste experimentelle Gehversuche im Internet zeigen das Potenzial von Canvas. Man wähnt sich bei der Durchsicht der im Folgenden genannten Beispielwebseiten teilweise auf flashbasierten Webseiten. Canvas könnte sich langfristig also zu einer Killerapplikation für das Flashformat entwickeln:

randomibis.com/coolclock/
9elements.com/io/projects/html5/canvas/
www.chromeexperiments.com

Zusammenfassung und Ausblick

Wie immer scheitern neue Technologien im Internet zuerst an der mangelnden Browserunterstützung, weil viele ältere Modelle noch im Einsatz sind. Auch HTML 5 befindet sich im Entwicklungsstadium. Ein grundlegender Unterschied besteht allerdings zu den vergangenen Jahren der Browserentwicklung: Die Browserhersteller setzen mehr und mehr gemeinsam auf universell einsetzbare Webtechnologien. Dank stetig steigender Prozessorleistung werden auch komplexe Javascript generierte AJAX-Module und auch durch API generierte Inhalte (API = Schnittstelle zur Anwendungsprogrammierung) zuverlässig (z.B. Google-Maps) und ohne Ruckeln dargestellt.

Ein deutlicher Trend geht also dahin, den Browser mehr Daten verarbeiten zu lassen. Ein Rückkanal zu einem Server, der Daten verarbeitet und dann wieder zum Browser sendet, bedeutet immer Zeit- und Perfomanceverlust. Datenverarbeitung in Echtzeit direkt im Browser des Nutzers wird die zukünftigen Einatzfelder von HTML 5 bestimmen. Die Kollaboration, d.h. z.B. die gleichzeitige Bearbeitung eines Dokuments durch mehrere Nutzer ist nun möglich. Es wird nicht mehr lange dauern und Sie können via Internet in Echtzeit 3D-Schach im Browser mit jemanden spielen.

Das jüngst veröffentlichte Google Wave ermöglicht im Zusammenspiel aktuellster Webtechnologien sogar ein webbasiertes und vollkommen plattformunabhängiges Betriebssystem. Der Browser und das Internet werden mehr denn je zum zentralen omnipotenten Werkzeug. Im Grunde würde man nicht mal mehr einen eigenen Computer benötigen. **whs**