HTMLWorld - eBook

eBook - Inhaltsverzeichnis

Einführung

von Jan Winker

Mit Erfindung des WWW und HTTP ist auch HTML immer bedeutender geworden und ist heutzutage kaum aus dem Internet wegzudenken. Allerdings wird es wohl keine weitere Version von HTML geben - HTML 4.01 wird vorraussichtlich die letzte gewesen sein.

Das World Wide Web Consortium (W3C), welches für solche Endscheidungen verantwortlich ist, hat mit dem Aufkommen von XML beschlossen HTML endgültig abzusagen und somit XML auf die Sprünge zu helfen. Nun soll eine auf XML basierende neue Sprache die Rolle von HTML übernehmen:

XHTML.

XML

XML steht für Extensible Markup Language (Erweiterbare Auszeichnungssprache). Die Vorteile von XML liegen klar auf der Hand: es ist fast beliebig erweiterbar, ist 'sauberer' Strukturiert und ist zugleich einfacher als SGML, von dem HTML abstammt.

Da es an sich ein sehr großen Umfang hat und diesen zu erklären wohl auch einige Seiten kosten würde, wird hier nun darauf verzichtet werden. Nur soviel dazu: XML ist eine Sprache mit der sich quasi jeder eigene Sprachen bzw. Elemente/Attribute schaffen kann. Diese wird und wurde schon als Grundlage für neue Sprachen genommen. Beispielsweise die Wireless Markup Language (Internetseiten fürs Handy), besser bekannt als WML, oder RDF sind von XML abgeleitet.

XHTML

XHTML steht für Extensible Hypertext Markup Language (Erweiterbare Hypertext Auszeichnungssprache). Von XML abgeleitet gibt es nun 3 XHTML DTD's (DTD ~ Beschreibungen, welche Elemente und Attribute ein Dokument enthalten darf und wie diese angeordnet sein müssen) die die letzte HTML Version im neuen XML wiederspiegeln. Prinzipiell soll XHTML aber kein neues HTML sein; vielmehr stellt es eine Möglichkeit dar, gleichzeitig fortschrittlich (weil XML) und trotzdem abwärtskompatibel (Elemente und Attribute entsprechen denen aus HTML) zu sein.

Von XHTML ausgehend sind nun wiederum einige neue Ideen und Sprachen abgeleitet worden. Zu denen gehören zum Beispiel *XHTML Basic* oder die *Modularization of XHTML*.

Gleichheiten XHTML und HTML

Prinzipiell sind sich die beiden Sprachen HTML und XHTML zum Verwechseln ähnlich. Das fängt dabei an, dass beide gleiche Elementnamen benutzen, beide haben gleiche bzw. ähnliche Attribute (bei den jeweiligen Elementen) und beide sollten sich nach den jeweils 3 bezeichneten DTD's richten. Dies hört sich nun erstmal etwas wenig an, beim näheren Betrachten wird man allerdings feststellen, das eigentlich alles "beim Alten" geblieben ist.

Unterschiede XHTML und HTML

Die Unterschiede der beiden Sprachen liegen klar darin, das XHTML von XML kommt und HTML "nur" von SGML. Dies bringt mit sich, dass XHTML durch die strengen XML-Regeln besser durchleuchtet werden muss um "echtes" XHTML zu sein. Wobei bei HTML hingegen etwas "wahrlos" mit Elementen und Attributen umgegangen werden konnte/wurde. Der größte Unterschied der beiden Auszeichnungssprachen liegt im wesentlichen also darin, bei XHTML die XML-Regeln zu befolgen und einzuhalten.

Unterschiede: Dokumente in XHTML

von Jan Winker

Da die Elemente und Attribute in etwa denen von HTML entsprechen, dürfte es hier keine großen Verwunderungen geben. Allerdings gibt es doch einige Dinge die in Hinsicht auf des gesamte Dokument zu beachten sind.

DTD's

Für XHTML wurden 3 DTD's definiert. Diese sind auf jeden Fall einzuhalten; Verstöße dagegen darf es nicht geben um korrektes XHTML zu schreiben.

Die 3 DTD's sind nachzulesen unter:

- XHTML-1.0-Strict - http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd
- XHTML-1.0-Transitional - http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd
- XHTML-1.0-Frameset - http://www.w3.org/TR/xhtml1/DTD/xhtml1-frameset.dtd

DOCTYPE

An die oben genannten DTD's schließt sich auch die DOCTYPE-Definition. In HTML konnte sie getrost weggelassen werden - der Browser hätte sich dann schon selbst die bestpassende ausgesucht. In XHTML ist dies nun nicht mehr der Fall: Will ein Dokument XML (bzw. XHTML) konform sein, muss es eine DOCTYPE-Definition enthalten. Diese muss sich vor dem Haupt-Element befinden und eine der drei oben genannten DTD's referenzieren. Die drei folgenden Beispiele entsprechen den in der Spezifikation beschriebenen DOCTYPE-Definitionen:

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN" "DTD/xhtml1-
strict.dtd">

<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN" "DTD/xhtml1-
transitional.dtd">

<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Frameset//EN" "DTD/xhtml1-
frameset.dtd">
```

Haupt-Element

Das Haupt-Element, also das Element, welches alle anderen Elemente einschließt, ist (weiterhin) das html-Element. Es muss gesetzt werden - fehlen darf es nicht.

Namespaces

Die durch XML herbeigebrachten Namespaces (Deklarationen auf DTD's bzw. weitere Elementinformationen) können in XHTML verwendet werden. Das Haupt-Element muss allerdings immer mit dem Namespace für XHTML ausgezeichnet werden. Dazu muss das xmlns-Attribut verwendet werden und mit der Adresse http://www.w3.org/1999/xhtml referenziert werden.

```
<html xmlns="http://www.w3.org/1999/xhtml">
```

Ebenso können auch weitere/zusätzliche Namespaces definiert werden. Die Verwendung und auch der

Einsatz davon wird allerdings jedem dabei frei gelassen.

XML-Deklaration

Zusätzlich sollte eine Deklaration als XML-Dokument in das Dokument mit einfließen. Dies wird zwar laut Spezifikation nicht unbedingt erwartet/verlangt, es wird den Autoren aber empfohlen diese stark zu fördern ("strongly encouraged"). Beispiel:

```
<?xml version="1.0" ?>
```

Der Nutzen dieser Deklaration liegt darin, zu helfen die richtigen Codierungsschematas zu finden, wenn andere als die normalen UTF-8 oder UTF-16 verwendet werden.

Zusammenfassung

Zusammenfassend nun nochmals ein Beispiel, welches die genannten Erneuerungen in einem XHTML-Dokument enthält. Beispiel:

```
<?xml version="1.0" ?>
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN" "DTD/xhtml1-
strict.dtd">
<html xmlns="http://www.w3.org/1999/xhtml">
 <head>
 <title>XHTML-Dokument</title>
 </head>
 <body>
 <p>Dieses Dokument ist ein XHTML Dokument.</p>
 <p><img src="name.gif" alt="ein Bild" /></p>
 </body>
</html>
```

Unterschiede: Notation

von Jan Winkler

Da in HTML, wie in der Einleitung schon erwähnt, etwas "wahrlos" in Bezug auf Notation, Elemente und Attribute umgegangen werden konnte/wurde, hat sich auch hier einiges geändert.

Wohlgeformtheit

Mit XML wurde das Konzept der Wohlgeformtheit eingeführt. Dies bedeutet, dass die Anordnung der Elemente nicht überlappend sein darf. Wird in einem geöffneten Element ein weiteres Element definiert, so muss dies geschlossen werden, bevor auch das erste Element geschlossen wird. Beispiel:

```
<!-- RICHTIG -->
<Element-1><Element-2> ... </Element-2></Element-1>

<!-- FALSCH -->
<Element-1><Element-2> ... </Element-1></Element-2>
```

Kleinschreibung

Alle Elementnamen und Attribute müssen klein geschrieben werden. Beispiel:

```
<!-- RICHTIG -->
<p><img src="name.gif" alt="ein Bild" /></p>

<!-- FALSCH -->
<P><IMG SRC="name.gif" Alt="ein Bild" /></P>
```

End-Tags

In HTML war es durchaus üblich End-Tags bei einigen Elementen einfach weg zu lassen. Dies darf nicht sein. Alle Elemente die Informationen jeglicher Art (Text, weitere Elemente; ausgenommen Attribute) enthalten, müssen einen End-Tag erhalten, der diese Informationen einschließt. Beispiel:

```
<!-- RICHTIG -->
<p> ... Inhalt 1 ... </p><p> ... Inhalt 2 ... </p>

<!-- FALSCH -->
<p> ... Inhalt 1 ... <p> ... Inhalt 2 ...
```

Sollte ein Element, welches in der jeweiligen DTD nicht mit EMPTY (also als leeres Element) definiert worden ist, einmal keine Informationen beinhalten, wird empfohlen nicht die verkürzte schreibweise zu verwenden, sondern besser das Element eben ohne Inhalt zu notieren. (Statt `<p />` besser `<p> </p>`)

Leere Elemente

Elemente, die keine Informationen (Text, weitere Elemente; ausgenommen Attribute) enthalten, brauchen weiterhin nur aus einem Tag bestehen. Diese müssen allerdings eine Markierung bekommen, dass es sich dabei um ein leeres Element handelt. Dies wird durch einen Schrägstrich (/) vor dem schließenden > notiert. Beispiel:

```
<!-- RICHTIG -->
<img src="name.gif" alt="ein Bild" />

<!-- FALSCH -->
<img src="name.gif" alt="ein Bild">
```

Zusätzlich wird geraten ein Leerzeichen vor diesen schließenden Schrägstrich zu setzten und leere Elemente auch als solche zu notieren und keinen schließenden Tag anzuhängen.

Verschachtelung

Bei den folgenden Elementen darf keine Verschachtelung mit dem selben oder bestimmten anderen Elementen stattfinden:

- `a` - Darf keine weiteren `a` enthalten
- `button` - Darf keine `button`, `fieldset`, `form`, `iframe`, `input`, `isindex`, `label`, `select` und `textarea` enthalten.
- `form` - Darf keine weiteren `form` enthalten.
- `label` - Darf keine weiteren `label` enthalten.
- `pre` - Darf keine `big`, `img`, `object`, `sub` und `sup` enthalten.

Attribute

In `XHTML` müssen alle Attributwerte, also das, was nach dem Ist-Gleich-Zeichen steht, in doppelte Anführungszeichen (`&`) gesetzt werden. Hier kann nicht, wie vielleicht zuvor, dies einfach vernachlässigt werden. Beispiel:

```
<!-- RICHTIG -->
<img src="name.jpg" alt="ein Bild"/>

<!-- FALSCH -->
<img src=name.jpg alt=ein Bild/>
```

Zusätzlich dürfen keine Attribute mehr abgekürzt werden. Die so in `HTML` beschriebenen Attribute müssen nun in `XHTML` einen Attributnamen sowie einen dem zugewiesenen Wert erhalten. Beispiel:

```
<!-- RICHTIG -->
<textarea disabled="disabled"> ... </textarea>
<area nohref="nohref"> ... </area>

<!-- FALSCH -->
<textarea disabled> ... </textarea>
<area nohref> ... </area>
```

Gleichfalls ist darauf zu achten, dass eventuelle Leerzeichen am Anfang und Ende eines Attributwertes weggeschnitten werden (z.B. vom Browser). Mehrere aneinanderhängende Leerezeichen und eventuelle Zeilenumbrüche werden zu einem Leerzeichen zusammengeschrupft (z.B. ebenfalls vom Browser).

Scripts und StyleSheets

Dadurch, dass die Elemente `script` und `style` in `XHTML` als Elemente mit dem Inhalt `#PCDATA`

(allg. Text) definiert worden sind, ergibt sich das Problem, dass Zeichen (z.B. < und >) nicht als Script- bzw. Style-Code gesehen werden könnten, sondern als ein Element. Dies soll vermieden werden, indem diese Script- und Styleangaben innerhalb eines CDATA-Bereiches notiert und nicht wie in HTML als Kommentar gesetzt werden. Der CDATA-Bereich wird durch <![CDATA[... Inhalt ...]]> beschrieben. Beispiel:

```
<!-- RICHTIG -->
<script>
 <![CDATA[ ... Inhalt ... ]]>
</script>

<style>
 <![CDATA[ ... Inhalt ... ]]>
</style>

<!-- FALSCH -->
<script> ... Inhalt ... </script>

<style> ... Inhalt ... </style>
```

Zusätzlich wird geraten, bei kritischen Fällen externe Scripte und Style-Angaben zu benutzen. In denen können dann sämtliche Zeichen verwendet werden, die nicht mit der Script-Sprache bzw. dem StyleSheet kollidieren.

Nicht festgeschriebene Unterschiede

von Jan Winkler

Zusätzlich zu den festgeschriebenen Unterschieden, die in den letzten zwei Seiten dokumentiert wurden, gibt es noch einige Dinge, die nur als Empfehlung zusätzlich gelten sollten. Dazu gehört z.B. das Setzten eines Leerzeichens vor den schließenden Schrägstrich bei leeren Elementen.

Isindex

Es wird geraten, das `isindex`-Element nicht mehr zu nutzen. Stattdessen sollte das mit einem `input`-Feld umschrieben/umgangen werden.

Bezeichnungen

Bitte achten Sie darauf, dass sich in `XML` `URI`'s wie `#ABC` nicht mehr auf Elemente mit dem Attribut `name=&ABC&` beziehen. Stattdessen besteht hier die Beziehung zu dem Element mit dem Attribut `id=&ABC&`. Um Abwärtskompatibilität zu gewährleisten wird geraten, bei solchen Dingen beide Attribute anzugeben.

Document Object Model

Da Unterschiede der Groß- und Kleinschreibung zwischen `HTML` und `XHTML` bestehen, die aber beide im `DOM` Level 1 beschrieben werden, wird geraten dieses Problem durch einen der zwei vorgeschlagenen Wege zu beseitigen.

- Dokumente die `text/html`-Type definiert sind, sollen das für `HTML` definierte Document Object Model (`DOM`) verwenden und demzufolge alle Elementnamen und Attribute in großen Buchstaben geschrieben werden.
- Dokumente die als`text/xml`-Type oder `application/xml`-Type definiert sind, sollen das für `XML` definierte Document Object Model (`DOM`) verwenden und demzufolge alle Elementnamen und Attribute in kleinen Buchstaben geschrieben werden.

Und-Zeichen in Attributen

Es wird geraten, in Attributen statt dem Und-Zeichen (`&`) den Zeichencode (`&`) zu verwenden. Aus dem Wert `/cgi-bin/name.pl?name1=abc&name2=def&` wird dann `/cgi-bin/name.pl?name1=abc&name2=def`.

Cascading Style Sheets (CSS)

Da Probleme beim Einlesen und Auswerten eines Dokuments bei der Verwendung von `CSS` auftreten können wird zu folgenden Dingen geraten:

- `CSS` in `XHTML` sollte Elemente und Attribute in kleingeschriebenen Buchstaben erhalten.
- Bei Tabellen sollte ein `tbody`-Element verwendet werden, wenn es sich dabei um einen `CSS`-Bereich handeln soll.
- Definierten StyleSheets sollte es möglich sein das Raute-Zeichen (`#`) weiterhin als Selektor zu benutzen (in Bezug auf die Namensvergebung mit dem `id`-Attribut).
- Definierten StyleSheets sollte es möglich sein das Punkt-Zeichen (`.`) weiterhin als Selektor zu benutzen (in Bezug auf die Namensvergebung mit dem `class`-Attribut).
- Es ist zu beachten, dass die unterschiedlichen Konformitätsregeln von `CSS` für das jeweilige Dokument eingehalten werden (in Bezug auf die Deklaration - siehe

Document Object Model)

Sonstige Empfehlungen

Die restlichen Empfehlungen sind größtenteils in der Dokumentation bereits genannt worden. Dazu gehören:

- Leerzeichen bei leeren Elementen
- Attributverkürzung
- Elementverkürzung
- Leerezeichen und Zeilenumbrüche in Attributwerten
- Zeichencodebenennung
- StyleSheet- und Scripteinbindung

XHTML Basic

von Jan Winkler

XHTML Basic ist eine Untergruppe von XHTML. Ein Problem von XHTML ist, dass zum (korrekten) Darstellen von Dokumenten meist aufwendige Programme, ein schneller Prozessor und/oder ein großer Arbeitsspeicher notwendig sind. Geräte mit begrenzten Kapazitäten, wie z.B. Mobiltelefone, PDAs, Settop-Boxen oder Pager, können dabei nicht schritthalten und geraten somit in einen Nachteil. Damit dies nicht der Fall ist und eine möglichst weitflächige Verbreitung und vielseitiger Zugang zu Informationen ermöglicht werden kann, wurde XHTML Basic geschaffen. Es bietet nur das Minimum an benötigten Elementen und Attributen (etc.) und wird somit leichter verwendbar und implementierbar sein. Des Weiteren wird es aufgrund ihrer Nähe zu XHTML von den meisten Webdesignern einfach und schnell zu verstehen sein.

Entstehung

Die Entstehung von XHTML Basic beginnt mit der Erfindung von mobilen Geräten mit denen man auf das Internet, WWW oder WAP-Dienste zugreifen kann. Seit dem existieren mehrere - meist firmenabhängige - Ideen und auch Sprachen, die eine Auszeichnung von Dokumenten für ebendiese (eingeschränkten) Geräte beschreiben und zulassen. Eine derartige Sprache ist beispielsweise WML die vom WAP Forum herausgegeben und vertrieben wird.
XHTML Basic greift bestimmte Grundideen und Elementtypen der bereits bestehenden Sprachen (wie z.B. Text-Dekoration, (Hyper-)Links, Formulare, Tabellen, Grafiken und Meta-Informationen) auf und verwendet diese. Dazu wurde Ende 2000 die erste Spezifikation (besser gesagt, die erste Recommendation zur Spezifikation von XHTML Basic) vom W3C herausgegeben.

XHTML Module

XHTML Basic ist von XHTML abgeleitet und wird durch die Verwendung von XHTML Modulen erweitert werden. Die Dokumenttyp-Definition (DTD) von XHTML Basic baut selbst auch schon auf der Ideen von Modulen auf und ist als solche implementiert.

Das verwenden von Modulen bringt einige wichtige Vorteile:

- Module können fast beliebig verwendet werden
- der Einsatz von bestimmten Modulen kann für bestimmte Geräte separat geregelt werden
- die Sprache an sich wird dadurch leichter (~ weniger)

Übereinstimmungen

Damit ein Dokument als ein korrektes XHTML Basic Dokument anerkannt werden kann muss es die folgenden Kennzeichen aufweisen:

- Das Dokument darf nur Elemente der XHTML Basic DTD sowie Elemente die mit den Regeln dieser DTD übereinstimmend sind verwenden.
- Das Haupt-Element muss das `html`-Element sein.
- Der Standard-Namespace des Haupt-Elements muss der von XHTML sein (`xmlns="http://www.w3.org/1999/xhtml"`).
- Es muss eine DOCTYPE-Beschreibung geben, die auf die XHTML Basic DTD verweist (`<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML Basic 1.0//EN" "http://www.w3.org/TR/xhtml-basic/xhtml-basic10.dtd">`).
- Es muss eine DOCTYPE-Beschreibung geben, die vor dem Haupt-Element zu finden ist.
- Wenn weitere DTDs eingebettet werden, darf keine Eigenschaft der XHTML Basic DTD überschrieben oder ersetzt werden.

Einschränkungen

Damit XHTML Basic funktionieren kann bedarf es einigen Einschränkungen. Diese sind wie folgt:

Formulare

Der Grundstamm von Forumlarelementen kann verwendet werden. Dazu gehören: form, input (text, password, checkbox, radio, submit, reset, hidden), label, select, option und textarea. Alle anderen Formularelemente werden aufgrund ihres geringen Nutzen für mobile Geräte nicht unterstützt. Des Weiteren sollte daran gedacht werden, dass die Benutzer möglicherweise nicht in der Lage dazu sein könnten lange Texte einzugeben.

Gestaltung und Rahmen

Da die meisten mobilen Geräte kaum oder nur wenige Möglichkeiten bieten Text zu gestalten, werden einige Text-Elemente nicht unterstützt. Dazu gehören u.a. b, i, u, bdo, del und ins, big und small, sub und sup sowie tt.

Sämtliche Arten von Rahmen werden aufgrund der meist zu kleinen Anzeigefläche nicht unterstützt.

Skripte und Events (Ereignisse)

Scripte werden nicht unterstützt (script, noscript) da bei mobilen Geräten meist eine zu kleine Rechenleistung vorliegt, sodass diese nicht in der Lage sind einfache oder komplexe Skripte auszuführen. Deshalb wird davon abgeraten.

Ebenso werden keine Events d.h. event-Attribute oder -Elemente unterstützt, da es kein einheitliches System gibt, welche Events verwendet werden sollten.

StyleSheets

StyleSheets (z.B. CSS) sollten nur in externen Dateien angeboten werden. Dies geschieht dazu, dass Geräte, die keine StyleSheets unterstützen keine unnötigen Daten laden müssen. Demzufolge ist das style-Element nicht in XHTML Basic enthalten. Um externe StyleSheets anzubieten kann - wie gewohnt - das link-Element verwendet werden. Des weiteren werden die Elemente span und div sowie das class sowie das media-Attribut zur stilistischen Anpassung durch StyleSheets unterstützt.

Tabellen

XHTML Basic unterstützt Tabellen nur in ihren einfachsten Gefügen. So sind nur unverschatelte Tabellen mit den folgenden Elementen möglich (alle anderen Tabellen-Elemente werden aufgrund ihres geringeren Nutzens für mobile Geräte nicht unterstützt): table, tr, th, td und caption.

Nicht enthalten sind u.a.: col und colgroup sowie thead, tbody und tfoot.

Elemente und Attribute

Die folgenden Elemente (mit ihren Attributen) sind in XHTML Basic:

- abbr, acronym, address, body, caption, cite, code, dfn, div, dl, dt, dd, em, h1 - h6, kbd, li, p, ol, samp, span, strong, ul, var
  class, id, title
  xml:lang
- a
  accesskey

charset
href
hreflang
rel
rev
tabindex
type
class, id, title
xml:lang

- `base`
  href
- `blockquote`, `q`
  cite
  class, id, title
  xml:lang
- `br`
  class, id, title
- `form`
  action
  method (get | post)
  enctype
  class, id, title
  xml:lang
- `head`
  profile
  xml:lang
- `html`
  version
  xmlns (http://www.w3.org/1999/xhtml)
  xml:lang
- `img`
  alt
  height
  longdesc
  src
  width
  class, id, title
  xml:lang
- `input`
  accesskey
  checked (checked)
  maxlength
  name
  size
  src
  tabindex
  type (text | password | checkbox | radio | submit | reset | hidden)
  value
  class, id, title
  xml:lang
- `label`
  accesskey
  for
  class, id, title
  xml:lang
- `link`
  charset
  href
  hreflang
  media
  rel
  rev

type
class, id, title
xml:lang

- `meta`
  content
  http-equiv
  name
  scheme
  xml:lang
- `object`
  archive
  classid
  codebase
  codetype
  data
  declare (declare)
  name
  standby
  tabindex
  type
  width
  class, id, title
  xml:lang
- `option`
  selected (selected)
  value
  class, id, title
  xml:lang
- `param`
  id
  name
  type
  value
  valuetype (data | ref | object)
- `pre`
  xml:space (preserve)
  class, id, title
  xml:lang
- `select`
  multiple (multiple)
  name
  size
  tabindex
  class, id, title
  xml:lang
- `table`
  summary
  width
  class, id, title
  xml:lang
- `td`, `th`
  abbr
  align (left | center | right)
  axis
  colspan
  headers
  rowspan
  scope (row | col)
  valign (top | middle | bottom)
  class, id, title
  xml:lang
- `textarea`

accesskey
cols
name
rows
tabindex
class, id, title
xml:lang

- `title`
  xml:lang
- `tr`
  align (left | center | right)
  valign (top | middle | bottom)
  class, id, title
  xml:lang