Tutorium – Webdesign

von Maria Lechner

Unterschied zwischen HTML & XHTML?

HTML – HyperText Markup Language (Hypertext-Auszeichnungssprache)
– SGML-basierte HTML

XHTML – Extensible HyperText Markup Language (erweiterbare Hypertext-Auszeichnungssprache)
– XML-basierte HTML

XHTML 1.0 ist nichts anderes als der Versuch, das SGML-basierte HTML 4.0 mit Hilfe von XML "nachzubauen"

XHTML ist also XML-gerechtes HTML.

Systembedingt durch die Syntax von XML gibt es jedoch diverse Unterschiede im Detail, die Sie kennen müssen, wenn Sie Ihre Web-Seiten in XHTML statt in HTML schreiben wollen.

Versionen von XHTML

1.0 / 1.1
Derzeit ist beim Attribut **version** der Wert 1.0 sinnvoll.

Dateiname

.htm oder .html – Browser verwendet HTML-Parser
.xhtml – Browser verwendet XML-Parser

XML-Deklaration & Zeichenkodierung

<?xml version="1.0" encoding="ISO-8859-1" ?>
Diese steht am Anfang einer XHTML-Datei. Eine solche Deklaration gibt es in HTML 4.0 nicht, sie ist also XHTML-spezifisch.

Möglichkeiten der Zeichenkodierung, nach der die die XHTML-Datei verarbeiten wird:

encoding="ISO-8859-1" – Kodierung ISO-8859-1 (Latin-1) für westeuropäische Sprachen. Ein Zeichen belegt in der Datei immer ein Byte.
encoding="UTF-8" – Internationale Kodierung auf Basis der ISO/IEC-10646-Norm (Unicode). Ein Zeichen belegt in der Datei eine variable Breite, z.B. ein Byte, aber auch zwei, drei oder vier Bytes.
encoding="UTF-16" – Internationale Kodierung auf Basis der ISO/IEC-10646-Norm (Unicode). Ein Zeichen belegt in der Datei je nach Unicode-Position entweder zwei oder vier Bytes.

Dokumenttyp-Deklaration

Beispiel für HTML 4.01
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01//EN"
"http://www.w3.org/TR/html4/strict.dtd">

Entsprechendes Beispiel für XHTML 1.0
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd">

Genauso wie in HTML gibt es in XHTML drei Varianten für den Doctype
Strict, Transitional und Frameset

HTML-Wurzelement mit Namensraumangabe

Bei HTML 4.01 hat der <html>-Tag meist keine Attribute.
<html>
<!-- Inhalt der Datei -->
<html>

Bei XHMTL 1.0 muss nun ein **Namensraum** explizit angegeben werden.
<html **xmlns="http://www.w3.org/1999/xhtml"**>
<!-- Inhalt der Datei -->
</html>

Strenges Einhalten des HTML-Grundgerüst
In HTML 4.01 war es möglich, den <body>-, <head>- oder <html>-Tag wegzulassen, stattdessen nur den <title>-Tag zu verwenden und trotzdem ein vollständiges und gültiges HTML-Dokument zu erhalten.

In XHTML besteht diese Freiheit nicht. Hier muss eine HTML-Datei zwingend das übliche Grundgerüst einhalten.

Beispiel für XHTML 1.0

```
<?xml version="1.0" encoding="ISO-8859-1" ?>
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
      "http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
<html xmlns="http://www.w3.org/1999/xhtml">
<head>
<title>Text</title>
<!-- gegebenenfalls andere Elemente im Kopfbereich -->
</head>
<body>
<h1>Text</h1>
</body>
</html>
```

Kleinschreibung

In HTML 4.01 war es egal, ob man die Tags klein, oder groß geschrieben hat. z. b. <TABLE> = <table>
Nicht so bei XHTML. XML unterscheidet nämlich strikt zwischen Groß- und Kleinschreibung. Das bedeutet, <TABLE> ist etwas anderes als <table>. **Für XHTML wurde festgelegt, dass alle Elementnamen und Attributnamen klein geschrieben werden.**

Leere Elemente

In HTML 4.0 gibt es diverse leere Elemente. Das sind Elemente ohne Inhalt. Der Abschluss-Tag wird bei leeren Elementen nicht verwendet, wie z. B. bei <img>,
, <input>, <hr>.

In XHTML werden diese leeren Tags nun mit Leertaste plus Schrägstrich notiert:
, <input />, <hr />, <img />

Attributwerte in Anführungszeichen

In XHTML ist es zwingend erforderlich, dass Attribute innerhalb der Anführungszeichen stehen.
Beispiel HTML 4.01: Anker
Beispiel XHTML 1.0: Anker

Leerraum in Wertzuweisungen

Bei Wertzuweisungen an Attribute muss in XHTML besser aufgepasst werden als in HTML. Leerzeichen sind erlaubt wo erforderlich, doch Zeilenumbrüche sollten vermieden werden.

Empfohlene Notation in XHTML 1.0:

Text Text Text

Problematische Notation in XHTML 1.0:
<p title="Anfang
der
Geschichte">Text Text Text

Verweise zu Ankern

Verweise zu Anker gehören in XHTML 1.0 nun so notiert:
Verweis
viel Inhalt
<a **id="anker" name="anker"**>irgendwas

Hinweis: id und name müssen den gleichen Namen vorweisen.

CSS

CSS – Cascading Style Sheets

Stylesheets sind eine Ergänzung zu HTML. Es handelt sich dabei um eine Sprache zur Definition von Formateigenschaften einzelner HTML-Elemente.

Ein wichtiges Leistungsmerkmal von CSS ist die Möglichkeit, zentrale Formate zu definieren. So kann man beispielsweise in einer externen Datei zentrale Definitionen zum Aussehen eines Elements notieren und dieses Stylesheet in viele HTML-Seiten parallel einbinden

Stylesheets unterstützen also die professionelle Gestaltung beim Web-Design und helfen beim Corporate Design für große Projekte oder für unternehmensspezifische Layouts.

Wichtig: Trennung von HTML und CSS. HTML ist das Grundgerüst mit dem Inhalt. Die Formatierung wird komplett in CSS definiert. Die Formatvorlagen werden wiederum in HTML angewendet.
Keine Formatdefinitionen im HTML-Dokument.

Syntaktische Noation

Selektor { Eigenschaft:Wert; }

Hinweis: Die Kombination aus Eigenschaft und Wert wird als Deklaration bezeichnet.

Formate zentral in separater CSS-Datei definieren

```
<html>
	<head>
		<title>text</title>
	<link rel="stylesheet" type="text/css" href="formate.css">
	</head>
<body>
	<h1>text</h1>
</body>
</html>
```

Verschiedene separate Stylesheets für unterschiedliche Ausgabemedien

```
<link rel="stylesheet" media="screen" href="website.css">
<link rel="stylesheet" media="print, embossed" href="druck.css">
<link rel="stylesheet" media="aural" href="speaker.css">
```

Siehe
http://de.selfhtml.org/css/formate/einbinden.htm#link_media

Kommentare innerhalb von Stylesheet-Bereichen

```
h1 {
color: blue; /* Basis-Stylesheet */
}
```

Formate für HTML-Elemente definieren

```
body {
	background-color:#FFFFCC;
	margin-left:100px;
}
```

```
h1 {
	font-size:300%;
	color:#FF0000;
	font-style:italic;
	border-bottom:solid thin black;
}

p,li  {
	font-size:110%;
	line-height:140%;
	font-family:Helvetica,Arial,sans-serif;
	letter-spacing:0.1em;
	word-spacing:0.3em;
}
```

Beispiel: elemente.htm

Formate für Klassen definieren

```
h1 {
	font-family:Arial,sans-serif;
	font-size:2em;
	font-weight:normal;
}

h1.hinterlegt {
	background-color:#FFFF00
}

*.hinterlegt {
	background-color:#00FFFF
}


.extra {
	background-color:#FF99FF
}
```

Hinweis:

Es gibt zwei Möglichkeiten, Klassen für HTML-Elemente zu notieren: entweder für einen bestimmten HTML-Elementtyp, oder für keinen bestimmten.
Der Stern gilt als Universalselektor und bedeutet so viel wie "für alle Elemente".
Klassen werden mit einem Punkt eingeleitet, darauf folgt der Klassenname.
Klassen verwendet man für Formatierungen.

Beispiel: klassen.htm

Klassen in Verbindung mit den Elementen div und span

div – division (Bereich)
Division sind Blockelemente und werden in CSS definiert. Sie sind für das Design und die Positionierung in HTML notwenig.

span
Span ist ein Inline-Element, das selbst keinerlei Eigenschaften hat und nichts bewirkt. Es ist dazu gedacht, um mit Hilfe von CSS formatiert zu werdne.

Der Unterschied zwischen div und span besteht darin, dass das div-Element eine neue Zeile im Textfluss erzwingt, während span innerhalb eines Texte verwendet werden kann und keinen neuen Absatz erzeugt.
Hier werden Klassen für die Formatierung innerhalb der divs bzw. spans verwendet.

```
.beitrag {
	border:1px outset gray;
	margin:.5em;
	padding:.5em;
	background-color:#efd;
}
.uebersetzung {
	border-bottom:1px dotted #900;
}
.autor {
	font-style:italic;
}
.datum {
	font-size:80%; color:#444;
}
```

```
<div class="beitrag">
  <p>Wo finde ich weitere Information über HTML?
  <p class="autor">Moritz Ratlos, <span class="datum">21.05.2006</span></p>
</div>
```

Beispiel: divspan.htm

Individualformate definieren

So wie man Formate für Klassen definieren können, die in HTML mit dem Universalattribut **class** angesprochen werden, kann man auch Formate definieren, die über das Universalattribut **id** angesprochen werden.

```
#roterBereich {
   position:absolute;
   top:130px;
   left:30px;
   width:320px;
   padding:10px;
   margin:0px;
   border:4px solid #EE0000;
}
#blauerBereich {
   position:absolute;
   top:130px;
   left:400px;
   width:320px;
   padding:10px;
   margin:0px;
   border:4px solid #0000EE;
}
h1#Titel {
   font-family:Arial,sans-serif;
   font-size:2em;
   font-weight:normal;
   color:green;
}
```

Hinweis:
Individualformate beginnen mit dem Gatterzeichen #, gefolgt von dem Namen. Ein HTML-Element, das diesen Namen als Wertzuweisung an das id-Attribut benutzt, bekommt dann die entsprechenden Formate zugewiesen.

Beispiel: individual.htm

Pseudoelemente und Pseudoklassen definieren

Mittels Pseudoklassen und Pseudoelementen kann man Deklarationen für HTML-Bestandteile definieren, die sich nicht durch ein eindeutiges HTML-Element ausdrücken lassen, z.B. "noch nicht besuchte Verweise".

```
body { font-family:Arial,sans-serif; }
 a:link { color:#EE0000; text-decoration:none; font-weight:bold; }
 a:visited { color:#EEAAAA; text-decoration:none; font-weight:bold; }
 a:hover { color:#EE0000; text-decoration:none; background-color:#FFFF99; font-weight:bold; }
 a:active { color:#0000EE; background-color:#FFFF99; font-weight:bold; }
```

Beispiel: pseudo.htm

Beispiel – CSS

```
*{
	padding: 0;
	margin: 0;
	list-style: none;
	}

html,body{
	height: 100%;
	}

body{
	font-family: Trebuchet MS, Verdana, Arial, sans-serif;
	font-size: 68.5%;
	text-align: left;
	color: #000000;
	}

h1, h2, h3, h5{
	color: #9e9e9e;
	font-weight: normal;
	}

h4 {
	font-size:1.1em;
	line-height: 1.3em;
```

```
        text-transform: uppercase;
        letter-spacing: 2em;
        margin-top:2em;
        margin-bottom: 2em;
}

p{
        font-size: 1em;
        line-height: 1.4em;
        margin-bottom: 2em;
        }

strong{
        background: #c4c4c4;
        color: #000000;
        padding: #000000;
        padding: 0.2em;
        }

a{
      text-decoration: none;
      color: #9e9e9e;
      background: inherit;
      }

a:visited {
      color:#9e9e9e;
      background: inherit;
}

a:active, a:hover, a:visited:hover{
      background-color:#666666;
      color: #ffffff;
}

#navi a:link, #navi a:visited {
        font-weight: normal;
        line-height: 1.5em;
}

img.right {
        float: right;
        }
```

```
#background{
	background: url(images/hg/background_stripes.jpg);
	background-repeat: repeat;
	position: absolute;
	height: 710px;
	width: 1100px;
	margin: 5px 0px 0px 5px;
	border: 0px;
	overflow: hidden;
	z-index: 0;
	}

#logo {
	position: absolute;
	height: 100px;
	width: 80px;
	margin: 600px 0px 0px 80px;
	padding: 0px;
	border: 0px;
	overflow: hidden;
	z-index: 6;
	}

#navi {
	background-color: #ffffff;
	position: absolute;
	height: 630px;
	width: 198px;
	margin: -10px 0px 0px 150px;
	padding: 70px 0px 0px 2px;
	border: 5px;
	border-color: #d2d2d2;
	border-style:solid;
	overflow: hidden;
	z-index: 7;
	vertical-align: bottom;
}

#impressum{
	position: absolute;
```

```
        height: 30px;
        width: 198px;
        top: 664px;
        line-height: 1em;
        overflow: hidden;
        z-index: 8;
}

#content {
        position: absolute;
        height: 615px;
        width: 1090px;
        margin: 80px 0px 0px 0px;
        border: 0px;
        overflow: hidden;
        z-index: 1;
}
```

Beispiel: css_beispiel.css

Maßeinheiten, Farbangaben und Wertzuweisung

Siehe
http://de.selfhtml.org/css/formate/wertzuweisung.htm

Numerische Angaben
pt – Punkt (z. B. font-size:12pt;)
px – Pixel (z. B. border-width:3px;)
em – relative Maßangabe (z. B. font-size:1.2em;)

Das Box-Modell

Siehe
http://de.selfhtml.org/css/formate/box_modell.htm

Das Box-Modell definiert die Berechnung der Breite und Höhe von Elementen.
Die Gesamtbreite eines Elements errechnet sich aus einer Addition von –
width – Breite des Elementinhalts
padding – Innenabstands
border-width – Rahmenstärke
margin – Außenabstands

Dies gilt analog für die Höhe (**height**).

Beispiel: box_modell.png

Beispiel: html_css_promton

Abkürzungen

SGML – Standard Generalized Markup Language (standardisierte verallgemeinerte Auszeichnungssprache)
DTDs – Document Type Definitions (Dokumenttyp-Definitionen)
XML – Extensible Markup Language (Erweiterbare Auszeichnungssprache)
MIME – Multipurpose Internet Mail Extensions
HTML-Parser – (to parse – analysieren) Der in einem Webbrowser enthaltene Parser analysiert das HTML und erstellt daraus eine Beschreibung der Webseite als Datenstruktur, welche die Grafik-Maschine des Browsers anschließend graphisch auf den Bildschirm überträgt.
XML-Parser – analysieren XML-Dokumente und stellen die darin enthaltenen Informationen (also Elemente, Attribute usw.) für die weitere Verarbeitung zur Verfügung.
CSS – Cascading Style Sheets

Links

http://validator.w3.org/

--
Quelle: wikipedia.org