Fortgeschrittene Programmierung

Prof. Dr. Thomas Richter

HyperText Markup Language (HTML)

Überblick

- Auszeichnungssprache
- Auf Klartext basierend
- Strukturierung von Inhalten (Texte, Bilder, Objekte, Links, …)
- Als Standardsprache zur Erstellung von Dokumenten die Grundlage des World Wide Webs
- HTML-Dokumente werden von Browsern geparst und angezeigt

Auszeichnungssprache

- Engl.: Markup Language
- Beschreibt den **Inhalt** eines Dokumentenformats
- Beschreibt mitunter auch das Bearbeitungsverfahren (z. B. früher im typografischen Drucksatz)
- Entwicklung hin zu komplexen Sprachen für digitale Typografie (HTML, XML, RSS, SVG, …)

Arten von Auszeichnungssprachen

- Descriptive ML (beschreibend)
  - HTML usw.

- Lightweight ML (leichtgewichtig, abgespeckt)
  - Vereinfachte Syntax für z. B. Wikis:

```
* Listenelement
* Listenelement
```

statt

```
<ul>
    <li>Listenelement</li>
    <li>Listenelement</li>
</ul>
```

- Procedural ML (prozedural)
  - PostScript
  - PDF
  - TeX / LaTeX

Grundsätze

- Wörter, Sätze oder Abschnitte bekommen eine Funktion (Auszeichnung), nicht wie in Textverarbeitungssystemen eine Gestaltung
- Der Auszeichnung wird dann das Aussehen zugeteilt
- Bei der Ausgabe wird die Gestaltung auf die Auszeichnung angewendet

- Im Web: Auszeichnung mit HTML, Gestaltung mit CSS (s. u.)

HTML Versionen

- 1992: HTML
  - Urversion von Tim Berners-Lee
- 1993: HTML
  - Attribute, Bilder
- 1995: HTML 2.0
  - u.A. Formulare, gemäß RFC 1866
- Januar 1997: HTML 3.2
  - Tabellen, Textflusskontrolle
- Dezember 1997: HTML 4.0
  - Stylesheets, Skripte, Frames

HTML Versionen

- 2000: XHTML 1.0
  - HTML 4.01 als XML-Dokument
- 2001: XHTML 1.1
  - Wegfall von Framesets
- 2006: XHTML 2.0
  - nie verabschiedet, Arbeiten zugunsten HTML 5 eingestellt
- 2009: HTML 5
  - Neufassung auf Basis von HTML 4.01 und XHTML 1.0
  - Erweiterung der DOM-Spezifikationen
  - Fortlaufende Weiterentwicklung ohne Versionsnummer

HTML Dokumentstruktur

- Dokumenttyp (HTML 5)
- Beginn des Dokuments

```
<!DOCTYPE html>
<html>
```

- Kopfbereich – Beschreibung des Dokuments und der Inhalte

```
<head>
</head>
```

- Inhaltsbereich – Angabe der Inhalte und deren Struktur

```
<body>
</body>
```

- Ende des Dokuments

```
</html>
```

Dokumenttyp

- Sagt dem Interpreter (meist Browser) was folgt, also HTML
- Steht immer in der ersten Zeile
- Seit HTML 5 ist der DOCTYPE einfach

```
<!DOCTYPE html>
```

<head>

- Enthält Meta-Daten des Dokumentes, die nicht angezeigt werden sollen
- Informiert den Browser
- Beispiele:

  `<title>` Titel der Seite, meist Titelleiste

  `<link>` logische Verweise z.B. RSS, externe Stylesheets

  `<script>` Code einer Skriptsprache

  `<style>` Stildeklaration (CSS)

<body>

- Der Anzeigebereich im Browser
- Unterscheidung zwischen Inline- und Block-Elementen
  - Block-Elemente immer selbstständige Zeile
  - Inline-Elemente stehen im Fließtext

HTML Element - Grundstruktur

```
<elementname attr="wert" ...>
   Inhalt des Elements
</elementname>
```

Oder

```
<elementname attr="wert"...>
```

(Empty Tag)

- Elemente legen die Semantik des Inhalts fest
  - Überschrift
  - Absatz
  - Bild usw.

Tags

- Ein Element besteht meist aus zwei Tags:

```
<elementname attr="wert" ...>
  Inhalt des Elements
</elementname>
```

- Öffnendes Tag ggf. mit Attributliste
- Schließendes Tag ohne Attributliste, mit führendem Slash /
- außer bei empty Tags immer paarweise

Attribute

- Werden im öffnenden Tag eines Elements angegeben

```
<elementname attr="wert" ...>
  Inhalt des Elements
</elementname>
```

- Bestehen aus Schlüssel/Wert Paaren
- Werte in doppelten Anführungszeichen **"wert"**
- Trennung durch Leerzeichen
- Steuern den Umgang des Browsers mit dem Element
- Früher gern gemacht: Layoutinformationen

HTML5

- Fertigstellung des Standards durch W3C im Oktober 2014
- „alte“ Seiten entsprechen oft schon dem HTML 5 Standard mit wenigen Änderungen:
  - Doctype
  - Content-type Definition
  - Einbindung von Stylesheets und Skripten
- Referenz
  - http://dev.w3.org/html5/html-author/

Neue Features in HTML5 (Auswahl)

- Semantische Strukturierung
- In-Place Editing
- Canvas
- Audio / Video Einbettung
- Clientseitige Persistenz

Doctype Deklaration

- HTML 4.01

```
<!DOCTYPE HTML PUBLIC "-
//W3C//DTD HTML 4.01
Transitional//EN">
```

- HTML5

```
<!DOCTYPE html>
<!doctype html>
```

Nach Standard [1] ist die Angabe nicht case sensitiv.

[1] https://www.w3.org/TR/html5/syntax.html#the-doctype

Inhaltstyp und Zeichenkodierung

- HTML 4.01

```
<meta http-equiv="content-type"
  content="text/html;
  charset=UTF-8">
```

- HTML5

```
<meta charset="utf-8">
```

Stylesheets und Skripte

- Exkurs:
  - Stylesheets dienen der Definition des Aussehens von Elementen
  - Scripte dienen der Angabe von ausführbaren Funktionen (Programmierung)

- HTML 4.01

```
<link type="text/css" rel="stylesheet"
   href="style.css">
<script type="text/javascript" src="script.js">
```

- HTML5

```
<link rel="stylesheet" href="style.css">
<script src="script.js"></script>
```

Kompatibilität

- CSS ist der HTML Standard für Stylesheets.
- JavaScript ist der HTML Standard für Skripte.

- Die Browser haben das auch früher schon vermutet.

→ HTML5 Header Deklarationen werden in der Regel auch von alten Browsern verstanden.

Veraltet

Elemente

- center
- font
- frame
- frameset
- noframes
- applet → object
- dir → ul
- ...

Attribute

- align, valign
- bgcolor
- height, width
- hspace, vspace
- target
- ...

Problem: Überstrukturierung

```
<div id="nav_wrap">
   <div id="nav">
      <ul>
         <li><a href="/">Home</a></li>
         <li><a href="k.html">Kontakt</a></li>
      </ul>
   </div>
</div>
```

Semantische Elemente zur Inhaltsbeschreibung

`header`	Kopfbereich einer Seite oder eines Abschnitts
`footer`	Fußzeile einer Seite oder eines Abschnitts
`nav`	Navigationsbereich einer Seite oder eines Abschnitts
`section`	Logischer Abschnitt einer Seite oder eine Gruppe von Inhalten
`article`	Artikel oder in sich abgeschlossenes Inhaltselement (z. B. in Blogs)
`aside`	Sekundäre oder ähnliche Inhalte

Semantisches Markup

- Dient der **Inhalts**beschreibung
- Unterstützt maschinelle Identifikation von Inhalten
  - Newsreader
  - Suchmaschinen
  - Benutzer-Stylesheets

HTML5 Fazit

- Versuch, Ordnung in den Versions- und Standardzoo zu bringen
- Viele bestehende HTML 4.01 und XHTML 1.0 sind valide HTML5-Dokumente

Was heißt eigentlich **valide**?

eXtensible Markup Language (XML)

- XML: Definition neuer Markupsprachen mit beliebigen Elementen und Attributen
- Metasprache zur Festlegung syntaktischer Regeln
- Wird u. a. verwendet zum dokumentenorientierten Datenaustausch
- HTML selbst ist nicht durch Benutzer (Entwickler) erweiterbar, wurde aber auf der Basis von XML entwickelt.

Wohlgeformtheit eines XML-Dokuments

1. Erste Zeile deklariert Sprachversion
2. Jedes Element besteht aus öffnendem UND schließendem Tag oder aus einem empty Tag

   `<img src="x.jpg">` nicht erlaubt
3. Genau ein Wurzelelement (`<html>`)
4. Saubere Schachtelung der Elemente
5. Attributwerte müssen quotiert sein (`"  "`)

Wohlgeformtheit, Beispiel

- Nicht valide:

```
<div><span>Inhalt</div></span>
```

- Valide:

```
<div><span>Inhalt</span></div>
```

Validität

- Gültigkeit von
  - Elementschachtelungen
  - Elementfolgen
  - Attributzuordnungen

  soll festgelegt und geprüft werden können.

- Wird erreicht durch die Bindung des Dokuments an
  - Document Type Definitions (DTD)
  - XML Schema Definitions (XSD)

XML Beispiel

```
<?xml version="1.0"?>
<BUCH>
   <TITEL>Engineering Web
Applications</TITEL>
   <AUTOR>Casteleyn et al.</AUTOR>
</BUCH>
```

- Das Beispiel hat eine gewisse Ähnlichkeit mit HTML
- XML und HTML wurden von SGML (Standard Generalized Markup Language) definiert

XSD und DTD

- XSD: XML Schema Definition
- DTD: Document Type Definition
- Beides dient der Beschreibung von XML-Dokumenten
- Bestimmt die Struktur bzw. den Aufbau eines XML-Dokumentes
- Klassen, Elemente, Attribute und deren gültige Zusammenhänge werden deklariert
- Hilfsmittel zur Validierung („Regeln“ die eingehalten werden müssen)
- XML-Dokument ist eine Instanz eines Schemas bzw. der DTD

Document Type Definition

- Vorteile
  - Hebt sich durch eigene Syntax vom XML-Dokument ab
- Nachteile
  - DTD ist kein XML-Dokument
  - Dokumenten-orientiert
  - Verschiedene Datentypen nur für Attribute (nicht für Elemente)
  - Keine Unterstützung für Namespaces
  - Eigene Syntax erfordert Erlernen dieser
  - Keine Mehrfachnutzung von Namen möglich

DTD Einbindung

```
<?xml version="1.0"?>
<!DOCTYPE Beispiel [
<!ELEMENT myElement (#PCDATA)> ]>
```

- Interne DTD kann direkt in die XML Datei geschrieben werden

```
<?xml version="1.0">
<!DOCTYPE foo SYSTEM „Beispiel.dtd">
```

- Der XML-Parser sucht nach der Datei Beispiel.dtd

DTD Beispiel

```
<!ELEMENT buch (titel, kapitel+)>

<!ELEMENT titel (#PCDATA)>
<!ELEMENT kapitel (headline, (text | zitat)*)>

<!ELEMENT headline (#PCDATA)>
<!ELEMENT text (#PCDATA)>
<!ELEMENT zitat (#PCDATA)>
```

- Die DTD schreibt vor, dass es immer einen Titel mit mindestens einem Kapitel gibt
- Im Kapitel steht immer zuerst die Überschrift, dann ein Text oder ein Zitat, wobei die Reihenfolge unterschiedlich sein kann

XML Beispiel 2

```
<?xml version="1.0"?>
<!DOCTYPE buch SYSTEM "buch.dtd">

<buch>
   <titel>Mein erstes Buch</titel>
   <kapitel>
      <headline>Einleitung</headline>
      <text>Einleitungstext</text>
      <zitat>Zitat in der Einleitung</zitat>
      <text>Weiterer Einleitungstext</text>
   </kapitel>
</buch>
```

XSD

- Vorteile
  - Gleiche Syntax wie XML-Dokument
  - Unterstützt alle modernen Datentypen
  - Wertebereich von Elementen
  - Jedes XML-Programm kann zur Erstellung genutzt werden
  - Leichter umzusetzen durch gleiche Syntax wie XML
  - Vererbung möglich
- Nachteile
  - Nicht abwärtskompatibel da basierend auf moderner Technologie
  - Komplexer als DTD
  - Nicht alle Parser verstehen XSD

Beispiel – artikel.xml

```
<?xml version="1.0">
<!DOCTYPE artikel SYSTEM "artikel.dtd">
<artikel>
   <titel>Der Titel</titel>
   <teaser>Der Teaser</teaser>
   <inhalt>Der Inhalt</inhalt>
</artikel>
```

Beispiel

artikel.dtd

```
<!ELEMENT artikel (titel,
teaser, inhalt )>
<!ELEMENT titel (#PCDATA)>
<!ELEMENT teaser (#PCDATA)>
<!ELEMENT inhalt (#PCDATA)>
```

artikel.xsd

```
<?xml version="1.0"?>
<xs:schema
xmlns:xs="http://www.w3.org/2001/
XMLSchema">
  <xs:element name="artikel">
    <xs:complexType>
      <xs:sequence>
        <xs:element
          name="titel"
          type="xs:string"/>
        <xs:element
          name="teaser"
          type="xs:string"/>
        <xs:element
          name="inhalt"
          type="xs:string"/>
      </xs:sequence>
    </xs:complexType>
  </xs:element>
</xs:schema>
```

Metainformationen

- Informationen über den Inhalt und Autor des Dokuments
- Syntax:

```
<meta charset="value" />
<meta {name|http-equiv}="key"
      content="value" />
```

- Attribut `name`: Informationen für den Client
- Attribut `http-equiv`: Informationen für den Server, der dann die HTTP-Header anpasst (wird aber nicht zwingend beachtet...)

Metainformationen - name

- description: Beschreibung des Dokuments, wird von Suchmaschinen berücksichtigt
- author: Autor des Dokuments
- keywords: Schlüsselworte, wird mit Einschränkungen von Suchmaschinen berücksichtigt
- date: Datum der Publikation
- robots: Suchmaschinen, Werte: index, noindex, nofollow, all und Kombinationen

Metainformationen – `http-equiv`

- expires: Verfallsdatum
- refresh: Weiterleitung (Zeitangabe und URL, getrennt durch Semikolon)
- content-language: Sprache, Sprachkürzel (de, en, ...)
- generator: Softwareverweis, oft automatisch eingebaut
- set-cookie: Cookie setzen
- cache-control: Caching des Browsers steuern

Metaangaben nach Dublin Core

- http://dublincore.org/: Dublin Core Metadata Initiative
  - Metadaten für alle möglichen "Dinge", unter anderem auch HTML-Dokumente
  - Werden als Standard vom W3C positiv beurteilt

- Eigenschaften, die als Elemente bezeichnet werden (Referenz: http://dublincore.org/documents/dcmi-terms/)

- Eigenschaften adressieren das name Attribut des `<meta>` Elements:

```
<meta name="DC.creator" content="Autor" />
<meta name="DC.description"
      content="Inhalt" />
```

Kommentare

- Syntax:

```
<!-- Das ist ein Kommentar -->
```

- Werden mitunter von Editoren verwendet, um bestimmte Bereiche zu kennzeichnen, Bsp. Dreamweaver:

```
<!-- #BeginEditable "Inhalt" -->
Text
<!-- #EndEditable -->
```

→Von Autorenwerkzeugen eingefügte Kommentare nicht verändern!

Textstrukturierung

- Überschriften
  - `<h1></h1> ... <h6></h6>`
- Absatz
  - `<p></p>`
- Bereich
  - `<section></section>`
- Artikel
  - `<article></article>`
- Kopf- und Fußbereiche
  - `<header></header>` und `<footer></footer>`

Inline- und Blockelemente

- HTML-Dokumente werden wie ein Text von links oben nach rechts unten ausgegeben.
- Unterbrechungen erfolgen nur durch Blockelemente, die eine neue Zeile beginnen und die gesamte Breite einnehmen.
- Inline (Bsp.): a, img!!!, input, label, span, …

- Block (Bsp.): article, aside, div, figure, footer, form, h1 – h6, header, p, section, table, ul, …

Umlaute / Ligaturen / Sonderzeichen

- Ä, ä: `&Auml; &auml;`
- Ö, ö: `&Ouml; &ouml;`
- Ü, ü: `&Uuml; &uuml;`
- ß: `&szlig;`

Diese Codes sind bei Verwendung von UTF-8 obsolet.

- &: `&`
- Sicheres Leerzeichen: ` `

Listen

- Aufzählungen `<ul></ul>`
- Nummerierungen `<ol></ol>`
- Listeneinträge für `<ul>` und `<ol>`: `<li></li>`
- Definitionslisten `<dl>`, `<dt>` und `<dd>`
- Ungeordnete Listen werden häufig für Navigationsstrukturen verwendet:

```
<ul>
  <li>Produkte</li>
  <li>Services
    <ul>
      <li>Beratung</li>
      <li>Realisierung</li>
    </ul>
  </li>
</ul>
```

Tabellen

- Verwendung zur strukturierten Darstellung von Daten, **nicht** zum Layout einer Seite.
- Wurzelelement `<table>`
- Strukturelemente `<thead>`, `<tbody>`, `<tfoot>`
- Zeilenelement `<tr>`
- Zellenelemente `<td>`, `<th>` (nur im Tabellenkopf)
- Spalten verbinden: Zellen-Attribut `colspan`
- Zeilen verbinden: Zellen-Attribut `rowspan`

Tabellen

Stundenplan Semester					
Zeit	**Mo**	**Di**	**Mi**	**Do**	**Fr**
08:15 – 09:45			Projekt		
10:00 – 11:30	FP				
12:15 – 13:45					
14:00 – 15:30					
Alle Angaben ohne Gewähr					

<table>
<thead>
<tbody>
<tr>
<th>
<td>
<tfoot>

Grafiken einbinden

- `<img>` Element (Empty tag)
  - Attribut `src` bestimmt die Bildquelle:
  - `<img src="http://www.hochschule-rhein-waal.de/fileadmin/templates/_img/logo.gif" />`
  - Attribut `alt`: Text für Sehbehinderte, obligatorisch!
  - Inline-Element!!!

- Bild mit Bildunterschrift:

```
<figure>
  <img src="..." />
  <figcaption>Bildunterschrift</figcaption>
</figure>
```

  - Blockelement

Formulare

- Element `<form>`
  - Attribut `action`: Wohin wird versandt, üblicherweise die Adresse eines Skripts
  - Attribut `method`: Wie wird versandt, üblicherweise `get` oder `post`

- Diverse Arten von Controls
  - Texteingabe ein- und mehrzeilig, Radiobuttons, Mehrfachauswahl, Buttons, Passworte, verdeckte Felder, ...

Hyperlinks

- Verknüpfung verschiedener Dokumente, Domains, oder Ankerpunkte
- Element `<a>`
- Verweis mit Attribut href:

```
<a href="http://www.hsrw.eu">HSRW</a>
```

- Anker im Dokument:

```
<a name="Abschnitt1">
  <h2>Abschnitt 1</h2>
</a>

<a href="dok.html#Abschnitt1">Text</a>
```

Literatur

- Praxisreferenz: de.selfhtml.org
- http://www.w3schools.com/html/