8 XML

- eXtensible Markup Language
- abgeleitet von SGML
- definierbare logische Struktur:
  markierte, sortierte Bäume
- mächtige Hyperlinkmöglichkeiten (XLink)
- Transformationssprachen (XPath, XSL)

- Anwendungen: WWW, Datenaustausch, E-Commerce
- spezialisierte Versionen: XHTML, WAP/WML, VoiceXML, DSML (directory services), UPnP (Universal Plug and Play), SyncML, Astronomical Markup Language, GedML: Genealogical Data in XML, XML/EDI (electronic data interchange), Synchronized Multimedia Integration Language (SMIL), Mathematical Markup Language, Open Trading Protocol (OTP), Weather Observation Markup Format (OMF), BIOpolymer Markup Language (BIOML), Bioinformatic Sequence Markup Language (BSML), Chemical Markup Language (CML), User Interface Markup Language (UIML), ...

Was ist eine Markup-Sprache?

- Notation zur Anreicherung von Text mit logischer Struktur (Charles F. Goldfarb)
- SGML Tradition: *Tags* der Form `<einTag>` (1985)

Was ist XML?

- *Rahmen* zur Definition von Markup-Sprachen
- einheitliche Markup-Syntax, Unicode basiert
- selbstdefinierte Tags
- keine vordefinierte Semantik
- Version 1.0 (1998), Version 1.1 (2004)

Beispiel: Wohlgeformtes XML Dokument

```
<?xml version="1.0"?>
<Eltern>
  <Kind>
    Hier kommt der Inhalt.
  </Kind>
  <Leer warum="weiß nicht" />
</Eltern>
```

XML Deklaration: `<?xml version="1.0"?>`

Elemente: `Eltern`, `Kind`, `Leer`

Öffnende Tags: `<Eltern>`, `<Kind>`

Schließende Tags: `</Kind>`, `</Eltern>`

Leeres Element: `<Leer />`

Attribut: `warum="weiß nicht"`
(im öffnenden Tag, bzw. im Tag des leeren Elements)

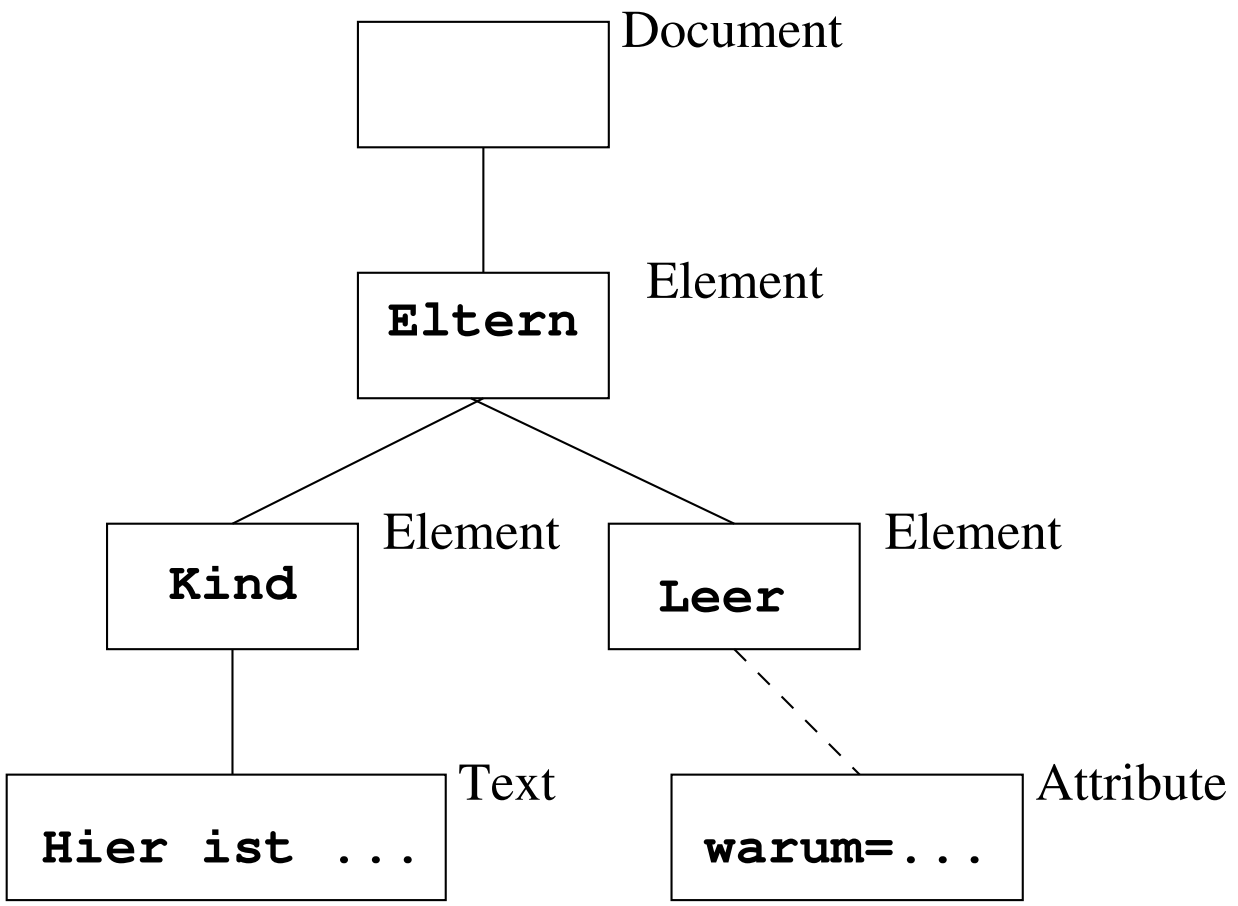
Wohlgeformtheit (well-formedness)

Wohlgeformt heißt

- XML Deklaration vorhanden
- sämtliche Elemente haben öffnendes und schließendes Tag
  (oder `<Leer />`, falls kein Inhalt)
- korrekte Schachtelung der Tags!
- genau ein Wurzelelement vorhanden
- jeder Attributname höchstens einmal pro Element

$\Rightarrow$ Dokument ist durch einen Baum repräsentierbar

Baumdarstellung des Dokuments



Knotentypen in XML Dokumenten

(XPath Terminologie)

Node übergeordneter Knotentyp

Text enthält Text, keine Kinder

Element enthält andere Knoten als Kinder (insbesondere Text, Element), sowie Attribute (sind nicht Kinder). Kinder von Elementen sind geordnet

Attribut Name/Wert-Paar anhängend an einem Elementknoten

Comment zählt weder zum Inhalt noch zur Struktur, keine Kinder

Processing Instruction Anweisung an XML-verarbeitendes Programm

Document Wurzelknoten eines XML Dokuments; enthält genau ein Element, das Wurzelelement

- Knoten eines Dokuments sind durch *Dokumentenordnung* total geordnet
- gegeben durch Baumdurchlauf in preorder von links nach rechts
- entsprechend der Reihenfolge in XML Notation
- Baumterminologie
  - root
  - siblings
  - leaves
  - children
  - parent
  - descendants
  - ancestors

Gültigkeit eines XML Dokuments

- muss die logische Struktur definieren

$\rightarrow$ DTD (Document Type Definition)

- DTD ordnet jedem Elementnamen e einen regulären Ausdruck $R(e)$ über Elementnamen zu
  falls <e><$e_1 \ldots$><$e_2 \ldots$>$\ldots$<$e_n \ldots$></e> in gültigem Dokument vorkommt, so ist $e_1 e_2 \ldots e_n \in L(R(e))$.
  dh die Namen der Kindelemente von e müssen dem Ausdruck $R(e)$ entsprechen
- DTD ordnet jedem Elementnamen eine Menge von getypten Attributen zu
- DTD definiert Abkürzungen (*entities*), die vom XML-Parser expandiert werden

```
<?xml version="1.0"?>
<!DOCTYPE ELTERN [
  <!ELEMENT ELTERN (KIND*)>
  <!ELEMENT KIND (MARKE?,NAME+)>
  <!ELEMENT MARKE EMPTY>
  <!ELEMENT NAME (NACHNAME+,VORNAME+)*>
  <!ELEMENT NACHNAME (#PCDATA)>
  <!ELEMENT VORNAME (#PCDATA)>
  <!ATTLIST MARKE
            NUMMER ID #REQUIRED
            GELISTET CDATA #FIXED "ja"
            TYP (natürlich|adoptiert) "natürlich">
  <!ENTITY STATEMENT "Wohlgeformtes XML">
]>
<ELTERN>
 &STATEMENT;
 <KIND>
  <MARKE NUMMER="1" GELISTET="ja" TYP="natürlich" />
  <NAME>
   <NACHNAME>Flavius</NACHNAME>
   <VORNAME>Secundus</VORNAME>
  </NAME>
 </KIND>
</ELTERN>
```


8.1 Inhalte eines XML Dokuments

- Elemente (benannte Baumknoten)
- Attribute (Namen/Wert-Paare an Baumknoten)
- Referenzen
  - *character reference* `&#x21;`, `&#62;`
  - *entity reference* `<`, `>`, `&STATEMENT;`
  - *parameter-entity reference* `%ISOLat2;` (nur in DTD)
- `<!-- Kommentare -->`
- Verarbeitungsanweisungen (*processing instructions*) `<?`*name daten*`?>` werden an Anwendung *name* übergeben
  `<?xml:stylesheet type="text/css2" href="style.css"?>`
- CDATA (*character data*)

  `<![CDATA[` *uninterpretierte Daten*
  werden wörtlich eingefügt
  `]]>`

- Vorspann
  - `<?xml version="1.0" encoding="UTF-8"?>`
  - Spezifikation einer internen DTD

    ```
    <!DOCTYPE Name [
      Element-, Attribut-, Entitydeklarationen
    ]>
    ```

  - Spezifikation einer externen DTD

    ```
    <!DOCTYPE HTML3
      PUBLIC "-//IETF//DTD HTML Strict//EN//3.0"
             "html-3s.dtd"
             [ ggf. interne Deklarationen ]>
    ```

8.1.1 Elementdeklaration

⟨*elementdecl*⟩ ::= <!ELEMENT ⟨*Name*⟩ ⟨*contentspec*⟩>

⟨*contentspec*⟩ ::= EMPTY | ANY | ⟨*Mixed*⟩ | ⟨*children*⟩

- Produktion einer kf. Grammatik
- ⟨*Mixed*⟩: #PCDATA und Elemente vermischt
- ⟨*children*⟩: regulärer Ausdruck über Elementnamen; Operatoren , | ? + *

8.1.2 Attributdeklaration

$$\langle AttlistDecl \rangle \quad ::= \quad \texttt{<!ATTLIST } \langle Name \rangle \ \langle AttDef \rangle^* \texttt{>}$$
$$\langle AttDef \rangle \quad ::= \quad \langle Name \rangle \ \langle AttType \rangle \ \langle DefaultDecl \rangle$$

$\langle AttType \rangle$ **::=**

`CDATA`	String
`ID`	eindeutiger Schlüssel
`IDREF`, `IDREFS`	Verweis(e) auf Schlüssel
`ENTITY`, `ENTITIES`	Name(n) von Entities
`NMTOKEN`, `NMTOKENS`	ein oder mehrere Wörter
`(`*Nmtoken* `(|` *Nmtoken*$)^*$`)`	Aufzählungstyp

⟨*DefaultDecl*⟩ **::=**

	`#REQUIRED`	erforderlich
|	`#IMPLIED`	nicht erforderlich, kein Default
|	`"`⟨*AttValue*⟩`"`	Defaultwert
|	`#FIXED "`⟨*AttValue*⟩`"`	fester Wert

8.1.3 Entity Deklaration

$$
\begin{array}{lcl}
\langle \mathit{GEDecl} \rangle & ::= & \texttt{<!ENTITY } \langle \mathit{Name} \rangle \ \langle \mathit{EntityDef} \rangle \texttt{>} \\
\langle \mathit{PEDecl} \rangle & ::= & \texttt{<!ENTITY \%} \ \langle \mathit{Name} \rangle \ \langle \mathit{PEDef} \rangle \texttt{>} \\
\langle \mathit{EntityDef} \rangle & ::= & \langle \mathit{EntityValue} \rangle \mid (\langle \mathit{ExternalID} \rangle \ \langle \mathit{NDataDecl} \rangle ?) \\
\langle \mathit{PEDef} \rangle & ::= & \langle \mathit{EntityValue} \rangle \mid \langle \mathit{ExternalID} \rangle
\end{array}
$$

- $\langle \mathit{GEDecl} \rangle$ *general entity*
  Abkürzungen, Dokumentenbausteine (auch von Dateien $\langle \mathit{ExternalID} \rangle$)
  `<!ENTITY dd "Dagobert Duck">`
- $\langle \mathit{PEDecl} \rangle$ *parameter entity*
  Parameter zur Steuerung der DTD Definition
  `<!ENTITY % ISOLat2 "INCLUDE">`

8.1.4 Media Type für XML Dokumente

- `text/xml` eingeschränkt auf Zeichensatz mit 1-Byte/Zeichen
- `application/xml` mit voller Unicodeunterstützung

8.2 XML Namespaces

"Modulsystem für XML"
Vermeidung von Namenskonflikten bei Benutzung von

- mehreren Quellen für Elementdeklarationen oder
- mehreren Anwendungen mit dem gleichen Dokument

Definition: An XML namespace is a collection of names, identified by a URI reference RFC RFC2396, which are used in XML documents as element types and attribute names.

- Qualifizierte Namen = $\langle Namespace\text{-}Präfix \rangle$: $\langle lokaler\ Teil \rangle$
- Präfix steht für URI, expandiert zu: $\langle URI \rangle$: $\langle lokaler\ Teil \rangle$
- Identität: String nach Expansion
- die URI ist beliebig, muß nicht funktional sein

Deklaration eines Namespace

- Deklaration ist jeweils gültig für das Eltern-Element des Attributs und sämtliche geschachtelten Elemente
- Attribut `xmlns` definiert Default-Namespace
  - unqualifizierte Elemente im Gültigkeitsbereich erhalten den Default-Namespace
  - Attribute müssen qualifiziert werden
- Attribut `xmlns:`⟨*Namespace-Präfix*⟩ bindet einen Namespace an das ⟨*Namespace-Präfix*⟩
- die `xmlns` Attribute können mehrfach auftreten
- Unqualifizierte Elemente außerhalb einer Default-Namespace Deklaration liegen im *leeren Namespace*, ebenso unqualifizierte Attribute

Beispiele

```
<x xmlns:edi='http://ecommerce.org/schema'>
   <!-- the "edi" prefix is bound to http://ecommerce.org/schema
        for the "x" element and contents -->
</x>
```

```
  <BOOKS>
    <bk:BOOK xmlns:bk="urn:BookLovers.org:BookInfo"
             xmlns:money="urn:Finance:Money">
      <bk:TITLE>A Suitable Boy</bk:TITLE>
      <bk:PRICE money:currency="US Dollar">22.95</bk:PRICE>
    </bk:BOOK>
  </BOOKS>
```

Beispiel: Default Namespace

```
  <BOOK xmlns="urn:BookLovers.org:BookInfo">
    <TITLE>A Suitable Boy</TITLE>
    <PRICE currency="US Dollar">22.95</PRICE>
  </BOOK>
```

8.3 XHTML

- Geschichte
  - HTML (Hypertext Markup Language)
  - Zusammenführung zweier Ideen
    - Hypertext
    - Markup-Sprachen (SGML)
  - ursprünglich logisches Markup
  - später: Einbau von Layout-Markup (HTML 4.01)
- Ziele von XHTML
  - Vereinfachung, daher Basis XML
  - Zurück zum logischen Markup, daher Einführung von Stylesheets
  - Transitionspfad (strict, transitional, frameset)
- Aktuelle Version: XHTML 1.0 `http://www.w3.org/TR/xhtml1`

8.3.1 Beispiel: XHTML Dokument

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
   "http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
<html>
  <head>
    <title>Internetprogrammierung, SS2005</title>
  </head>
  <body>
  <!-- siehe nächste Folien -->
  </body>
</html>
```


Beispiel: XHTML Inhalt

```
<h1>Spezialvorlesung Internetprogrammierung, SS2005</h1>

<h2><a name="sec:admin">Allgemeines</a></h2>

<div align="center">
  <table>
    <tbody>
      <tr><td></td><td>Vorlesung</td><td>Übung</td></tr>
      <tr>
        <td>Durchführung</td>
        <td>
          <a href="http://www.informatik.uni-freiburg.de/~thiemann/">
            Prof. Dr. Peter Thiemann</a>
        </td>
        <td>
          <a href="http://www.informatik.uni-freiburg.de/~neubauer/">
            Matthias Neubauer</a>
        </td>
      </tr>
    </tbody>
  </table>
</div>
```

Beispiel: XHTML Inhalt, Forts.

```
<h2><a name="id2">Vorlesung</a></h2>
<ul>
  <li>Erste Vorlesung: 11 April 2005</li>
  <li>Abschlussprüfung: mündlich, am 13.07.2005</li>
</ul>

<hr />
<address>
  <a href="http://www.informatik.uni-freiburg.de/~thiemann">
    <tt>Peter Thiemann</tt>
  </a>,
  Feb 28, 2005
</address>
```

8.3.2 Formulare in XHTML

- Dokumente mit Eingabemöglichkeiten (Textfelder, Bilder, Knöpfe, etc)
- Startpunkt für interaktive Webauftritte
- Anforderungen
  - Art der Eingabefelder definieren
    verschiedene XHTML Elemente
  - Eingabefelder benennen (String Attribut)
  - Wertebereich der Eingabefelder festlegen (String)
  - Ziel für die Eingabe festlegen (URI)
- Abstrakt ist ein XHTML Formular
  - Record von Strings + Instruktionen zum Versenden an den Server
  - repräsentiert durch Liste von Namen und Werten

Bauteile eines Formulars

Deklaration eines Formulars durch ein `<form>` Element

```
<!ELEMENT form (#PCDATA | %block; | %inline; | %misc;)*>
```

- fast alle XHTML Elemente innerhalb von `<form>` erlaubt
- keine geschachtelten Formulare
- zusätzlich Eingabeelemente innerhalb von Formularen

```
<!ENTITY % inline.forms "input | select | textarea | label | button">
```

- jedes Eingabeelement definiert
  - einen Namen für die Eingabe und
  - einen Wert (String, ggf. durch die Eingabe)

```
<input name="f1" value="foo" />
<select name="f2"> options </select>
```

Versendeattribute von `<form>`

`action="`*`url`*`"` erforderlich

URL des Empfängerskripts; meist `http:...` oder `https:...`; `mailto:...` möglich, aber nicht Standard;

`method="`*`HTTP-Methode`*`"`

- `GET` (Voreinstellung)
- `POST`

`enctype` Kodierung (media type) zum Versenden der Formulardaten

- `"application/x-www-form-urlencoded"` (Standard)
- `"multipart/form-data"` (falls Dateieingabe)

Eingabeelemente

Jedes Eingabeelement erzeugt (mindestens) ein Paar

Name=*Wert*

das beim Einreichen des Formulars an die `action` URL geschickt wird

```
<!ELEMENT input EMPTY>
```

Attribute

`type=` `text` (Voreinstellung), `password`, `checkbox`, `radio`, `submit`, `reset`, `file`, `hidden`, `image`, `button`

`name="`*string*`"` erforderlich, jedes Eingabefeld muß einen eindeutigen Namen haben

`value="`*string*`"` voreingestellter Wert

Typen von Eingabefeldern

text Textfeld

password Textfeld ohne Echo

checkbox Kästchen zum Ankreuzen

radio Druckknopf; höchstens einer gedrückt

submit Knopf zum Abschicken des Formulars

reset Knopf zum Zurücksetzen des Formulars

file Dateieingabe

hidden erscheint nicht im Browser; wird mit verschickt

image Bild, die Koordinaten des Mausklicks werden verschickt

button einfacher Druckknopf

Menüs

Ausschnitt aus der DTD (XHTML 1.0)

```
<!ELEMENT select (optgroup|option)+  <!-- Auswahlmenü -->
<!ATTLIST select
  name      CDATA          #IMPLIED  <!-- Feldname -->
  size      %Number        #IMPLIED  <!-- Anzahl sichtbare Zeilen -->
  multiple  (multiple)     #IMPLIED  <!-- einfache / mehrfache Auswahl -->
  >

<!ELEMENT optgroup (option)+         <!-- Untermenü -->

<!ELEMENT option (#PCDATA)           <!-- Menüpunkt -->
<!ATTLIST option
  selected  (selected)     #IMPLIED  <!-- zu Beginn ausgewählt -->
  value     CDATA          #IMPLIED  <!-- Wert des Elements -->
  >
```

Standardverwendung

```
<select name="⟨Feldname⟩">
  <option value="⟨Wert⟩"> angezeigter Menüeintrag </option>
  ...
</select>
```

- Falls kein ⟨*angezeigter Menüeintrag*⟩ vorhanden, wird ⟨*Wert*⟩ angezeigt
- Übertragen wird der ausgewählte ⟨*Wert*⟩
- Falls mehrere Menüpunkte ausgewählt sind (`multiple`), wird für jeden angewählten Wert ein Paar ⟨*Feldname*⟩=⟨*Wert*⟩ übertragen.
- Bei `size=1` erscheint ein Popup-Menü; bei `size=`*Zahl* eine Auswahlliste mit *Zahl* Einträgen (ggf. mit Scrollbar).

Beispiel Formular

```
<form
  method="post"
  action="http://localhost/cgi-bin/TestCGI?wo-ist-das"
  name="myform">
  <br />Login: <input type="text" name="login" maxlength=80>
  <br />Password: <input type="password" name="password" maxlength=8>
  <br />Password forgotten: <input type="checkbox" name="checkbox-1">
  <br /><input type="submit" name="DO-ONE-ACTION" value="Do something">
  <br />
  <select name="menu-1">
    <option>first menu item</option>
    <option selected="selected">second menu item</option>
  </select>
</form>
```

Einreichung des Formulars

Eingabe von

- Login: `ich`
- Password: `PaSsWoRd`
- und Klick auf `Do something`

sendet an den Server

```
name=ich
password=PaSsWoRd
DO-ONE-ACTION=Do+something
menu-1=second+menu+item
```

Probleme mit Formularen

- nur Stringeingabe, nicht getypt
- nicht einmal `maxlength` Attribut überprüft
- Abhängigkeiten zwischen Eingabefeldern nicht spezifizierbar