Vorlesungsreihe
Entwicklung webbasierter Anwendungen

XML

Prof. Dr.-Ing. Thomas Wiedemann
email: wiedem@informatik.htw-dresden.de

HOCHSCHULE FÜR TECHNIK UND WIRTSCHAFT DRESDEN (FH)
Fachbereich Informatik/Mathematik

Gliederung

- Übersicht
- XML
- XML-Schema
- XSLT
- XHTML

- Hinweis: Aufgrund des sehr zahlreich und detailliert vor-liegenden Online-Materials stellen die folgenden Folien nur eine Übersicht und Zusammenfassung zum Thema dar.
- Details und konkrete Beispiele können den Quellen, dabei insbesondere der Dokumentation unter http://www.w3schools.com/xml/ entnommen werden.

Überblick zu HMTL und XML

- HTML und XML sind Datenbeschreibungssprachen auf Anwendungsebene (entspricht OSI-Schichten 6+7)
- Transport erfolgt über TCP im Client/Server-Modus (XML teilweise auch über UDP) oder aufsetzende Protokolle wie http
- Gemeinsame Wurzel ist SGML, eine sehr komplexe Dokumentenbeschreibungssprache aus den 80er Jahren
- HTML (Hypertext Markup Language) ist eine Beschreibungssprache für Text- und Multimediadarstellungen
- XML dient zur Definition neuer, anwendungsspezifischer Sprachen
- beide sind als Klartext-Format definiert
  - erfordern prinzipiell keine speziellen Editoren oder proprietären Programme
  - Verarbeitung erfolgt durch Interpretation (Parsen + Ausführung von zugeordneten Visualisierungsfunktionen)
  - vor allen der starke Leitungszuwachs der Prozessoren erlaubt eine derartige Echtzeitinterpretation ohne nennenswerte Verzögerungen beim Seitenaufbau

XML -Einführung

- XML - Extensible Markup Language (dt: Erweiterbare Auszeichnungssprache)
- ist wie HTML eine vereinfachte Untermenge von SGML (welche bereits 1986 für allgemeine Dokumentenbeschreibungen definiert wurde)
- 1998 erste Empfehlung (Recommendation) zu XML durch W3-Konsortium
- Aktuell: **XML 1.0 (Fifth Edition)** – W3C Recom. vom August 2008 oder **XML 1.1** von 2006 (vgl. W3C-XML-eite: http://www.w3.org/XML/
- Zitat W3C: „XML 1.1 updates XML so that it no longer depends on the specific Unicode version … You are encouraged to create or generate XML 1.0 documents if you do not need the new features in XML 1.1; XML Parsers are expected to understand both XML 1.0 and XML 1.1."
- Zitat W3C: Der "revolutionäre" Vorteil von XML liegt in der Bereitstellung eines Regelwerkes zur Definition beliebiger Datenbeschreibungssprachen
  - bei sehr hoher Flexibilität
  - und gleichzeitig guter Praktikabilität und Unterstützung durch Tools
- bereits existierende Ableitungen von XML sind
  - MathML: Mathematical Markup Language
  - SVG: Scalable Vector Graphics für 2D-Vektorgrafiken
  - XHTML ist eine Neudefinition von HTML mit den Regeln von XML (->HTML5!)
  - WML: Wireless Markup Language für Handy-Darstellungen

XML-Grundlagen

Allgemeine Regeln

- Daten werden hierarchisch mit Tags strukturiert
- Tags <tagname> </tagname> sind durch Anwender frei definierbar
- Tags können auch Attribute enthalten <Seite Nr="12">...</Seite>
- unterliegen den Regeln einer Document Typ Definition (DTD), welche extern oder am Anfang der Daten definiert
- beschrieben wird die logische Bedeutung der Daten, nicht die Formatierung

  <Seitentitel>Literaturangaben</Seitentitel>
- Spezielle Formatangaben werden in externen XSL (++) abgelegt und durch Hilfsprogramme in andere Ausgabeformate gewandelt (z.B. Postscript für Drucker oder auch HTML/CSS)

XML-Beispieldefinitionen

Beispiele (aus selfhtml) – man beachte die deutschen Tagnamen

```
<rechteck oben="100" links="185" breit="427" hoch="110">
   <hintergrund typ="verlauf" richtung="waagerecht" startfarbe="#0000FF"
    endfarbe="#FFFFFF">
    <inhalt typ="text" format="stil_6">
      Ein kleiner Text
    </inhalt>
   </hintergrund>
 </rechteck>

 <projekt sprache="perl" name="Performance-Test IC-Baustein TL410" typ="shell"
    stand="02-10-2001">
   <modul name="main" stand="02-10-2001" ablage="/usr/scripts/tl410/tl410.pl">
    <funktion name="datenversorgung" stand="14-09-2001">
     <beschreibung>
      versorgt den Speicherbaustein mit sinnvollen Anfangswerten aus Testreihe T3.
     </beschreibung>
    </funktion>
   </modul>
 </projekt>
```

Spezielle Regeln in Abweichung von HTML

- eine XML-Datei startet immer mit <?xml version="1.0"?> und muß mindestens ein Element enthalten
- optional sind weitere Attribute möglich

  <?xml version="1.0" encoding="ISO-8859-1" standalone="yes"?>
- XML unterscheidet Groß- und Kleinschreibung bei Tags
- Tagname dürfen nicht mit der Zeichenfolge xml und nicht mit Ziffern beginnen (und dürfen kein = oder Leerzeichen enthalten)
- Tags DÜRFEN NICHT INEINANDER GESCHACHTELT werden

  Tags <Seite><Abschnitt>text</SEITE><Abschnitt> ist nicht erlaubt
- es gibt keine alleinstehenden Tags, d.h. Elemente bestehen IMMER aus Tag und End-Tag, Attribute zwingend mit ""
- leere Elemente können als <tagname /> geschrieben werden

Falls diese Regeln eingehalten werden,
ist die XML-Datei wohlgeformt !

XML-Gültigkeit

- Wenn eine XML-Datei auch den Regeln einer XML-Document-Type-Definition (DTD) in allen Elementen entspricht so ist die Datei GÜLTIG !
- Eine DTD muß nicht zwingend existieren, kann jedoch helfen Fehler oder potentielle Inkorrektheiten zu vermeiden.
- Die DTD am Anfang der XML-Datei definiert die gültigen Elemente und deren Anordnung :

```
<?xml version="1.0"?>
<!DOCTYPE quelle [
  <!ELEMENT quelle (adresse, beschreibung)>
  <!ELEMENT adresse (#PCDATA)>
  <!ELEMENT beschreibung (#PCDATA)>
]>
<quelle>
  <adresse>http://www.willy-online.de/</adresse>
  <beschreibung>alles zum wichtigsten Teil am Manne</beschreibung>
</quelle>
```

DTD

XML-DTD

- DTD´s können selbst definiert, wieder verwendet oder modular aus mehreren, einzelnen DTD´s aufgebaut werden (siehe selfhmtl)
- sind analog zum Softwareentwurf zu entwickeln (auch wie DB-Entwurf)
- sind reine Textdateien (jedoch LEIDER kein XML-Format !! -> Nachteil !)
- **Allgemeine Form**

  <!ELEMENT Name (Inhalt)>
- Einfachste Form

  <!ELEMENT Name (#PCDATA) > - enthält nur Text (keine < >)

  PCDATA steht für PARSED Character Data
  - Nachteil : PCDATA unterscheidet (noch) keine Datentypen
- **Elemente als Kombination anderer Element**

  <!ELEMENT telefonnummer (vorwahlnummer, durchwahlnummer)>

  <!ELEMENT vorwahlnummer (#PCDATA)>

  <!ELEMENT durchwahlnummer (#PCDATA)>
- **Mögliche Wiederholungen** werden mit einem + (mindestens einmal) oder * (0 bis N-mal) angegeben

  <!ELEMENT Telefone (telefonnummer)* >

XML-DTD II

- Kombinierte Elemente können andere Elemente auch alternativ enthalten:

  <!ELEMENT adresse (anrede?, name, (postfach | wohnanschrift), plzort)>

  ? Steht für optionale Angabe | steht für alternative Angabe

- Beliebige Reihenfolgen der Elemente werden mit einem PCDATA innerhalb einer alternativen Liste angegeben

  <!ELEMENT text (#PCDATA | zynisch | lachend)*>

  <!ELEMENT zynisch (#PCDATA)>

  <!ELEMENT lachend (#PCDATA | augenzwinkernd)*>

  <!ELEMENT augenzwinkernd (#PCDATA)>

- gültig wäre dann :

  ... Sagt die Frau zum Mann: <lachend>Mann o Mann, <augenzwinkernd>du bist dein Geld wert!</augenzwinkernd></lachend>

  Antwortet der Mann: <zynisch>Ja, weil ich den Einkaufswagen schiebe und sich eine Mark darin befindet!</zynisch>

  Daraufhin die Frau: <lachend>Du hast es erfasst!</lachend> ...

XML-DTD III

Weitere, optionale Bestandteile von DTD´s

- Bedingte Abschnitte mit INCLUDE und / IGNORE :

```
<![INCLUDE[
<!ELEMENT Vorsitzender (#PCDATA)>
]]>
<![IGNORE[
<!ELEMENT Zweitvor (#PCDATA)>
]]>
```

- Kommentare : <!– Aufpassen ! -->

- **Entities für Textbausteine** (=Makros / Textbausteine – auch ü ist Entity)

```
<!ELEMENT text-mit-baustein (#PCDATA)>
<!ENTITY mfg "mit freundlichen Gr&#252;&#223;en" >
```

Anwendung durch Aufruf des Entitynamens mit &:

```
<text-mit-baustein> ich verbleibe &mfg; </text-mit-baustein>
```

=> Entities erlauben sehr mächtige Konstrukte zur effizienten, auch verteilten und modularen Strukturierung von XML-Dokumenten

- **Namespace-Räume zur Anwendung unterschiedlicher DTD´s in einem Dok.**

```
<buch xmlns="http://www.meinserver.de/XML/buch">
... <html xmlns="http://www.w3.org/TR/REC-html-40">
```

XML-Schema

- definiert vom W3C : http://www.w3.org/XML/Schema
- behebt Nachteile von DTD´s (kein XML, keine Typen)
- beschreiben in einer komplexen Schemasprache Datentypen, einzelne XML-Schema-Instanzen (Dokumente) und Gruppen solcher Instanzen.
- ein konkretes XML-Schema wird auch als eine XSD (XML-Schema-Definition) bezeichnet und hat als File üblicherweise die Dateiendung ".xsd".
- unterstützt im Gegensatz zu DTD´s auch Datentypen
  - atomare Datentypen: xsd:string, xsd:decimal, xsd:integer ,xsd:float , xsd:boolean, xsd:date, xsd:time
  - Listen und Unions bestehend aus atomaren Elementen
  - Komplexe, aus den vorherigen Typen zusammengesetzte Typen

Eine Beispieldefinition :

```
<xsd:simpleType name="monatInt">
  <xsd:restriction base="xsd:integer">
    <xsd:minInclusive value="1"/>
    <xsd:maxInclusive value="12"/>
  </xsd:restriction>
</xsd:simpleType>
<xsd:simpleType name="monate">
  <xsd:list itemType="monatInt"/>
</xsd:simpleType>
```

Eine Instanz davon :

```
<monate>
  1 2 3 4 5 6 7 8 9
</monate>
```

Übergang von DTD´s zu XML-Schema

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<!DOCTYPE artikel SYSTEM „adr.dtd">
<artikel xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:noNamespaceSchemaLocation="adr.xsd">
<adresse>  <aname>Lutz Mustermann</aname>  <plz>01069</plz>
<ort>Der Inhalt</ort>
</adresse>
```

```
<!ELEMENT adresse (aname, plz, ort)>
<!ELEMENT aname (#PCDATA)>
<!ELEMENT plz (#PCDATA)>
<!ELEMENT ort (#PCDATA)>
```

DTD-Datei: adr.dtd

Schema-Datei: adr.xsd

```
<?xml version="1.0“ encoding="ISO-8859-1"?>
<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema">
 <xs:element name="artikel">
  <xs:complexType>  <xs:sequence>
    <xs:element name=“aname“  type="xs:string"/>
    <xs:element name=“plz“  type="xs:string"/>
    <xs:element name=“ort“  type="xs:string"/>
  </xs:sequence>  </xs:complexType>  </xs:element> </xs:schema>
```

XML-Schema-Details

- unterscheidet Elemente und Attribute auch mit default/fixed Angaben

  <xs:element name="dateborn" type="xs:date“ default=“2000-1-1“ />

  <xs:attribute name="lang" type="xs:string" fixed="EN“ />
- Attribute sind per Default OPTIONAL ! -> mit use=„required“ als notwendig def.

  <xs:attribute name="lang" type="xs:string" use="required"/>
- weitergehende Einschränkungen des Auftreten und der Werte

  <xsd:element name=“typ" type="xsd:string" minOccurs="0" maxOccurs="1"/>
- Einschränkungen de Werte mittel der Schlüsselwörter
  - length, maxLength, minLength – zul. Länge Strings oder Liste
  - enumeration – mögliche (alternative) Werte
  - pattern – Beschränkung durch Angabe eines regulären Ausdrucks
  - minExclusive, minInclusive, maxExclusive, maxInclusive – Wertebereich
  - totalDigits, fractionDigits – Dezimalstellen Gesamtzahl /Nachkommast.
  - whiteSpace – Behandlung von Leerzeichen und Tabs
- Aufbau modularer Beschreibungen

  <include schemaLocation="http://www.example.com/schemata/harddisk.xsd"/>

  <include schemaLocation="http://www.example.com/schemata/ram.xsd"/>

XML-Schema-Beispiele

- Restriktionen

```
</xsd:simpleType>
<xsd:simpleType name="KörperTemp">
 <xsd:restriction base="xsd:decimal">
  <xsd:totalDigits value="3"/>
  <xsd:fractionDigits value="1"/>
  <xsd:minInclusive value="35.0"/>
  <xsd:maxInclusive value="42.5"/>
 </xsd:restriction>
</xsd:simpleType>
```

```
<xsd:simpleType name="plz">
  <xsd:restriction base="xsd:string">
   <xsd:pattern value="(D )?[0-9]{5}"/>
  </xsd:restriction> </xsd:simpleType>
```

```
<xsd:simpleType name="size">
 <xsd:restriction base="xsd:string">
  <xsd:enumeration value=“big"/>
  <xsd:enumeration value=“medium"/>
  <xsd:enumeration value=“small“/>
</xsd:restriction> </xsd:simpleType>
```

```
<xs:element name="password">
<xs:simpleType>
 <xs:restriction base="xs:string">
  <xs:pattern value="male|female"/>
 </xs:restriction></xs:simpleType>
</xs:element>

<xs:element name="password">
<xs:simpleType>
<xs:restriction base="xs:string">
<xs:pattern value="[a-zA-Z0-9]{8}"/>
</xs:restriction> </xs:simpleType>
</xs:element>

<xsd:element name=“teileliste">
 <xsd:complexType><xsd:sequence>
    <xsd:element name="pc" type="pc-
Typ" maxOccurs="unbounded"/>
  </xsd:sequence>
 </xsd:complexType></xsd:element>
```

Ausgabe von XML-Dokumenten mit CSS

XML-Dok. können (gegenwärtig) mit CSS oder XSL/XSLT ausgegeben werden

Die CSS-Ausgabe

- erfordert einen Browser, welcher die Kombination aus XML+CSS versteht **!**

  ⇒ **z.Z.:** 

⇒ **Vorteile**: läuft komplett im Browser (keine Serverfunktionen notwendig)

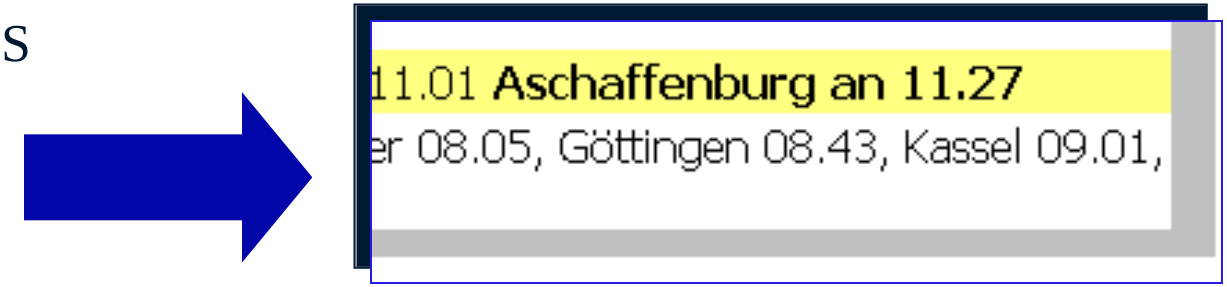
⇒ **Nachteile:** Probleme mit älteren Browsern (keine oder fehlerhafte Anzeige)

Bestandteile :

- DTD zum Dokument : <!ELEMENT route (#PCDATA | start | ziel)*>
- CSS-Stylesheet fahrplan.css ... ziel { font-weight:bold; color:0000E0; }
- XML-Daten <?xml-stylesheet type="text/css" href="fahrplan.css" ?>

  mit Verweis ... <ziel>Aschaffenburg an 11.27</ziel>

  zur DTD und zum CSS



Ausgabe von XML-Dokumenten mit CSS - II

Grenzen und Probleme bei der Darstellung mit CSS (gekürzt nach [selfhtml])

- CSS ist für Anwender sicher einfacher zu verstehen
- ältere, auch CSS-fähige Browser zeigen entweder gar nichts oder Text ohne jede Formatierung an. Damit Probleme bei allg, Anzeige im Web – dagegen in einem Intranet KEINE Probleme !
- Mit CSS in Version 1.0 lassen sich nur Elemente formatieren, keine Attribute. Erst ab CSS 2.0 besteht die Möglichkeit, die Verbindung zwischen einer solchen Attributangabe und einer entsprechenden Textformatierung herzustellen. Allerdings unerstützen nicht alle Browser CSS 2.0 .
- CSS bietet erst ab Version 2.0 die Möglichkeit, automatische Nummerierungs-Schemata auf Elemente anzuwenden, etwa auf Kapitelüberschriften (Kapitel 1, 1.1, 1.2, 1.2.1, 1.2, 1.3 usw.). In XSL/XSLT ist die Möglichkeit der Nummerierung von vorneherein enthalten.
- KEINE Filterung von Daten möglich ! In XSL/XSLT besteht eine solche Möglichkeit.
- Auch weitere Leistungsmerkmale von XSL/XSLT, wie etwa die automatische Sortierung von Daten vor der Ausgabe, ist mit CSS derzeit nicht möglich.

Ausgabe von XML-Dokumenten mit XSL/XSLT

XSL (*Extensible Stylesheet Language*, zu deutsch: erweiterbare Formatsprache) besteht aus folgenden Komponenten:

1. Komponente zur Formatierung von XML-Daten (oft *XSL-FO* - "XSL Formatting Objects" genannt) – seit 2013 veraltet (CSS3 Paged Media Module als Ersatz)!
2. Komponente zur Transformation von XML-Daten in andere XML-Daten. Abk. für Transformations-Komponente XSLT (*XSL Transformation)*
3. (zusätzlich Xpath als eine Subsprache innerhalb von XSLT. Grund: XSLT ist so mächtig ist, dass sie ein Eigenleben entwickelt hat.)

Hauptaufgaben von XSL/XSLT

- XSL als allgemeine Stylesheet-Sprache für XML-Daten
- Im Gegensatz zur Webausrichtung von CSS auf Print-Publikationen orientiert.
- Neben vielen Style-Eigenschaften auch Funktionen zur Steuerung des logischen Ablauf der Datenpräsentation steuern, wie z.B. automatische Kapitel-nummerierung, Filter, Sortierfunktionen, bedingungsabhängige Anweisungen oder Schleifenanweisungen, die eher an Programmiersprachen erinnern.
- Gegenwärtig meist Transformation von XML nach HTML : -> jeder Browser kann die Daten anzeigen kann. Dabei sind beliebige HTML-Konstrukte erlaubt - einschließlich JavaScript und CSS (auch gemischt !)

Ausgabe von XML-Dokumenten mit XSL/XSLT II

Notwendige XSL/XSLT-“Infrastruktur“

- **notwendig ist ein Parser, welcher das XML-Dokument analysiert (parst und dann entsprechend umformatiert ausgibt...)**

a.) Parser im Browser -> alle modernen Browser sollten heute XSLT-Standard-Transformationen ausführen (sehr alte Browser nicht)

b.) Parser im oder bei Server :

- CGI-Programme wie **Saxon** (auch für Tests gut)
- für Apache-Webserver :
  - XALAN - XSLT-Modul (in Java oder in C++)
  - PERL / PHP – Module (siehe entspr. VL / Prakt.)

Ausgabe von XML-Dokumenten mit XSL/XSLT III

Die wichtigsten XSLT-Strukturen

- **xsl-template** – definiert Formatierungsvorschrift auf Element „Produkt“ :

```
<xsl:template match="produkt">
  <span style="color:blue"> <xsl:value-of select="." /></span>
</xsl:template>
```

- **xsl:value-of select** - setzt entsprechenden Tagvalue ein
- **xsl:for-each select** - führt Template mehrfach über alle Daten aus

```
<xsl:template match="/">  <html> <head> </head> <body>
<table border="1"> <tr> <td><b>Begriff</b></td> <td><b>Definition</b></td> </tr>
<xsl:for-each select="glossar/eintrag">
  <tr> <td valign="top"><xsl:value-of select="begriff" /></td>
  <td valign="top"><xsl:value-of select="definition" /></td>
    </tr> </xsl:for-each>
</table> </body> </html> </xsl:template> </xsl:stylesheet>
```

Unterstützung der Ausgabe mit XPATH

XPATH ist eine unterstützende Sprache für

- die eindeutige Adressierung von XML-Daten innerhalb komplexer mehrdeutiger Datenstrukturen (Details siehe [selfhtml] und w3c.org)

  `<xsl:value-of select="/child::adresse/child::vorname" />` (ausf. Notation)

  `<xsl:value-of select="/adresse/vorname" />` (kurze Notation)

- zusätzliche Hilfsfunktionen für alle Ebenen der XML-Transformation

  - arithmetische und Logische Operatoren für Berechnungen und Vergleiche
  - eine große Anzahl von Funktionen (z.B. sum(), round(), position(), not(),...)

- Definition logischer Ausdrücke zur Steuerung der Transformation und Auswertung komplexer Daten auf der Basis von XSL und XPATH-Ausdrücken :

  Bsp.: `<xsl:if test="position() != last()"> <xsl:text>, </xsl:text> </xsl:if>`

Unterstützung der Ausgabe mit XLINK und XPOINTER

XLINK ist eine unterstützende Sprache für

- die Referenzierung von Links in XML-Dokumente in Anlehnung an HTML
- Beispiel (Details siehe w3c.org)

```
<sites xmlns:xlink="http://www.w3.org/1999/xlink">
  <site xlink:type="simple" xlink:href="http://www.site.com">
    Text </site>
</sites>
```

XPOINTER ist eine zusätzliche Hilfssprache in XPATH

- zur Querrefenzierung von Dokumenten
- durch Einbinden von Marken in Dokumente

```
<daten    id="marke123">  datentext   </daten>
```

- können diese aus anderen Dokumenten referenziert werden :

```
<fact xlink:type="simple" xlink:href="http://..url.xml#marke123">
```

Einordnung vom XHTML

- Mit dem rasanten Siegeszug von XML seit 1998 entstand auch der Wunsch, die Präsentation mit HTML kompatibel zu XML zu gestalten.
- erste Version XHTML 1.0 (zur Abgrenzung von HTML neue Versionierung) seit dem Jahr 2000
- deckt in etwa Funktionsumfang von HTML 4.0 ab (in Details ggf. abweichend bei älteren Browsern)

Vorteile :

- XHTML ist syntaktisch kompatibel zu anderen XML-Sprachen
- damit Einbindung von oder in andere XML-Sprachen leicht möglich
- Verarbeitung von XHTML kann mit Standard-XML-Tools erfolgen, da gleiches **DOM** (*Document Object Model = Dokument-Objektmodell*).
- in der Regel Verarbeitung / Transformation von XHTMl durch **XSL**

Nachteile

- als Ersatz für HTML 4.0 sinnvoll, jedoch ggf. Probleme mit älteren Browsern
- **Achtung: ab Ende 2011 kann XHTML als „gestorben“ angesehen werden**
- **=> siehe VL1 zu HTML5**

XHTML - Bereitstellung

Den Zielapplikationen (Browser) muß die Verarbeitungsart der XHTML - Dateien mitgeteilt werden, davon hängt die Art der Verarbeitung ab :

a.) über den MIME-Type im http-Protokoll

- Mime = text/html ist definiert für normale HTML-Dokumente
  - bei einer Übertragung von XHTML mit diesem MIMETyp werden die Dokumente vom normalen HTML-Parser verarbeitet -> hohe Fehlertoleranz, kleinere Fehler (auch ungültige XML-Def.) werden akzeptiert.
  - **Für den Einsatz als HTML-Ersatz (noch) gebräuchlich !**
- MIME-Typ = text/xml führt zur Verarbeitung als XML-Format
  - falls keine Formatierungsvorschriften (XSLT /CSS) definiert sind, wird beim IE nur der XML-Baum dargestellt oder es erfolgt ein Download-Angebot
- MIME-Typ = application/xhtml+xml ist speziell für XHTML definiert
  - Browser sollte dieses Format explizit bei der Anforderung unterstützen, sollte langfristig verwendet werden, Fehler im XML-Format führen zum Abbruch !!!

B.) über die Dateinamenserweiterung

- Bei *.html erfolgt die Verarbeitung analog zu text/html
- Bei *.xml / *.xhtml erfolgt die Verarbeitung durch den XML-Parser.

XHTML - Dokumenttyp-Deklarationen

Durch verschiedene Dok.typ.Deklaration soll der Übergang erleichtert werden :

- Beispiel für HTML 4.01:

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

- 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">
```

- In Analogie zu HTML 4.0 gibt es auch in XHTML 1.0 es die drei Varianten Strict, Transitional und Frameset :

```
... "http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
... "http://www.w3.org/TR/xhtml1/DTD/xhtml1-frameset.dtd">
```

Wesentliche Unterschiede zu HTML 4.0

- **Notwendige Namensraumangabe im HTML-Wurzelelement**

  `<html xmlns="http://www.w3.org/1999/xhtml">`
- **Strengere Einhaltung der Grundstruktur - die Elemente html, head und body dürfen nicht weggelassen werden !**
- **Groß- und Kleinschreibung genau beachten bzw. unterscheiden**
- **Es sind keine leeren Elemente zugelassen :**

  Html: `<p>Text mit<br>2. Zeile</p><p><img src="bild.png" alt="Bild">`

  XHTML: `<p>Text mit<br />2. Zeile</p><img src="bild.png" alt="Bild" />`
- **In XHTML müssen Attribute immer in "" geschrieben werden**

  HTML: `<a name=anker>...</a>` -> `<a name="anker">Anker</a>`
- **Attribute ohne Wertzuweisung in XHTML mit Default-Zuweisung versehen :**

  `<td nowrap>Inhalt</td>` -> `<td nowrap="nowrap">Inhalt</td>`

- Keine unnötigen Leerzeichen und Zeilenumbrüche in Zuweisungen
- **Universalattribut lang für html und xhtml doppelt vermerken :**

  `<html xmlns="http://www.w3.org/1999/xhtml" lang="de" xml:lang="de">`
  `<!-- Inhalt der Datei --> </html>`

Aktuelle Entwicklungen im XML-Bereich

- im Vergleich zu anderen Formaten sehr große Dateien und aufwändig zu parsen
- einer der Vorteile von XML die Selbstbeschreibungsfähigkeit von XML, ist gleichzeitig auch ein Performancenachteil
- Entwicklung von Lösungsalternativen -> z.B. **Binary XML**
  - Generelle Ziele: kleinere Datenmenge und schnelleres Parsen
  - (leider) noch kein führender Standard
  - Optionen (vgl. https://en.wikipedia.org/wiki/Binary_XML) :
    - ganz einfach: Gzip von XML-Dateien (kleiner, aber langsamer)
    - **BiM** (Binary MPEG format for XML (auch MPEG7) zur Beschreibung von multimedialen Daten mittels XML-Daten
    - **WBXML** (WAP Binary XML) mit Fokus auf Mobilfunkübertragung (-> SyncML / MS Exchange ActiveSync-Protokoll)
    - **EXI** (Efficient XML Interchange) vom W3C in 2014 als Recom veröffentlicht (Entwicklung seit 2005) -> www.w3.org/XML/EXI/ mit Lauflängenkodierung auf der Basis einer lexikalischen Analyse => verschiedene Referenzimplementierungen (auch Open Source)

Zusammenfassung / Bewertung - XML -

- **Im Ergebnis einer gewissen Hype-Phase Ende der 90iger / Anfang der 200er Jahre wurde sehr viele IT-Datenformate auf XML umgestellt**
  - **vgl. Microsoft-Dokumentenformate DOCX etc.**
  - **Konfigurationsdateien in Java und anderen IDE´s meist auch im XML-Format**

- **Aktuell gewisse Ernüchterung durch einige Nachteile von XML**
  - **Im Vergleich zu anderen Formaten sehr große Dateien und aufwändig zu parsen**
  - **einer der Vorteile von XML die Selbstbeschreibungsfähigkeit von XML, ist gleichzeitig auch ein Performancenachteil**
  - **Entwicklung von Lösungsalternativen (z.B. Binary XML) war bisher noch nicht sehr erfolgreich**
  - **kleinere Konkurrenzformate wie JSON sind effektiver und schneller**

These: Für stark hierarchische Datenstrukturen ist XML weiterhin die beste Lösung, für kleinere oder komplexere Aufgaben sind andere Formate ggf. sinnvoller.