



Aufbau von XML-Dokumenten

Prof. Dr.-Ing. Robert Tolksdorf
Freie Universität Berlin
Institut für Informatik
Netzbasierende Informationssysteme
tolk@ag-nbi.de



Aufbau von XML-Dokumenten

Heutige Vorlesung

- XML-Syntax
 - Elemente
 - Attribute
 - Deklaration
 - ...

- Namensräume

Wiederholung: Was ist XML?



- XML ist eine Methode, um strukturierte Daten in einer Textdatei darzustellen.



- XML sieht ein wenig wie HTML aus.



- XML ist Text, aber nicht zum Lesen.



- XML ist eine Familie von Techniken.



- XML ist neu, aber nicht so neu.



- XML überführt HTML in XHTML



- XML ist modular



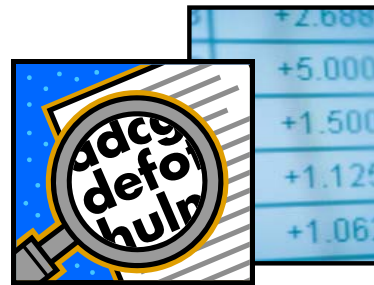
- XML ist lizenzfrei und plattformunabhängig



XML-Syntax: XML-Dokumente

Was ist ein XML-Dokument?

Inhalt: Text oder Daten



kodiert als

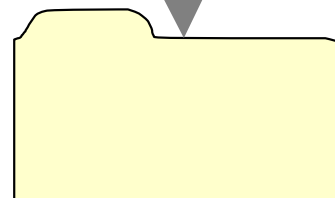
XML-Dokument

```
<name>
  <first>John</first>
  <last>Doe</last>
</name>
```

Objekt, das
Syntaxregeln von XML
entspricht (wohlgeformt
ist)

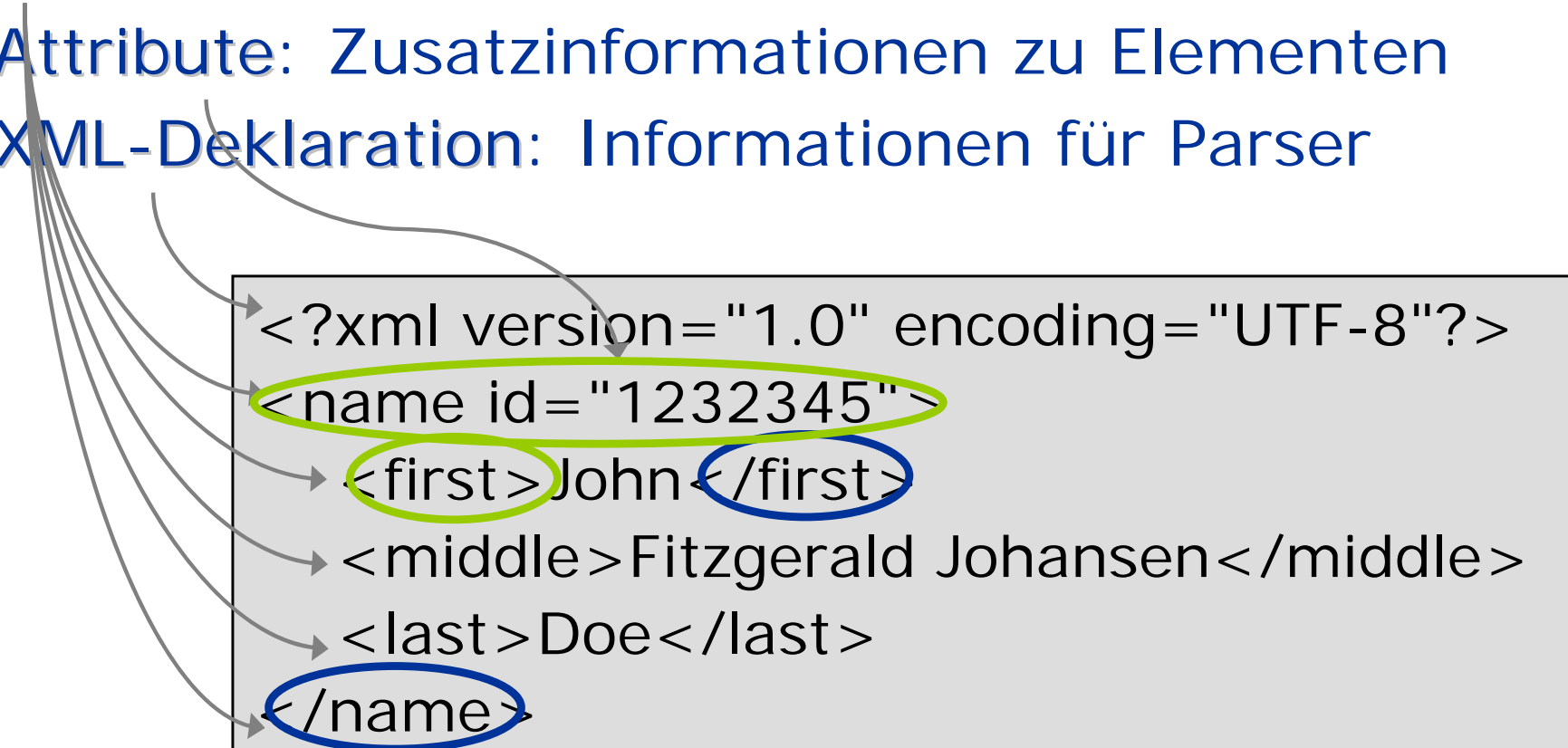
gespeichert in

XML-Datei



- **Elemente:** strukturieren das XML-Dokument
- **Attribute:** Zusatzinformationen zu Elementen
- **XML-Deklaration:** Informationen für Parser

```
<?xml version="1.0" encoding="UTF-8"?>  
<name id="1232345">  
  <first>John</first>  
  <middle>Fitzgerald Johansen</middle>  
  <last>Doe</last>  
</name>
```



- **Namensräume:** lösen Namenskonflikte auf und geben Elementen eine Bedeutung



XML-Syntax

Grundbausteine: XML-Elemente

- Beispiel:



- besteht aus:

- einem **Anfangs-Tag** (engl. *start tag*): hier `<first>`
- einem dazugehörigen **Ende-Tag** (engl. *end tag*): hier `</first>`
- einem **Inhalt**: hier „John“

- alles zusammen bildet ein **Element**

- haben einen **Namen**: hier „first“

1. **unstrukturierter Inhalt:**
 - einfacher Text ohne Kind-Elemente
2. **strukturierter Inhalt:**
 - Sequenz von $> \emptyset$ Kind-Elementen
3. **gemischter Inhalt:**
 - enthält Text mit mind. einem Kind-Element
4. **leerer Inhalt**

1. Unstrukturierter Inhalt

- Beispiel: `<first>John</first>`
- einfacher Text ohne Kind-Elemente
Kind-Element: Element, das im Inhalt eines Elementes vorkommt
- unstrukturierter Inhalt auch als *Parsed Character Data (PCDATA)* bezeichnet:
 - **character data**: einfache Zeichenkette
 - **parsed**: Zeichenkette wird vom Parser analysiert, um Ende-Tag zu identifizieren
 - Normalisierung: u.a. Zeilenumbruch (CR+LF) → #xA

Anmerkung: Auf den Folien schreibe ich der besseren Lesbarkeit wegen *Kind-Element* statt *Kindelement*!

- Reservierte Symbole < und & in PCDATA nicht erlaubt.
- Symbole wie >, /, (,), {, }, [,], % allerdings erlaubt
- statt < und & Entity References & bzw. < benutzen
- Entity References in XML:

& ⇒ &

< ⇒ <

> ⇒ >

' ⇒ '

" ⇒ "

- Unstrukturierten Inhalt mit vielen reservierten Symbolen besser als Character Data (CDATA) darstellen.

- Beispiel:

```
<formula>  
  <![CDATA[ X < Y & Y < Z ]]>  
</formula>
```

- Inhalt: String zwischen inneren Klammern []
hier: $X < Y \ \& \ Y < Z$
- XML-Parser sucht in CDATA lediglich `]]>`, analysiert den Inhalt aber ansonsten nicht.
- `"]]>` als Inhalt von CDATA nicht erlaubt

2. Strukturierter Inhalt

- Beispiel:

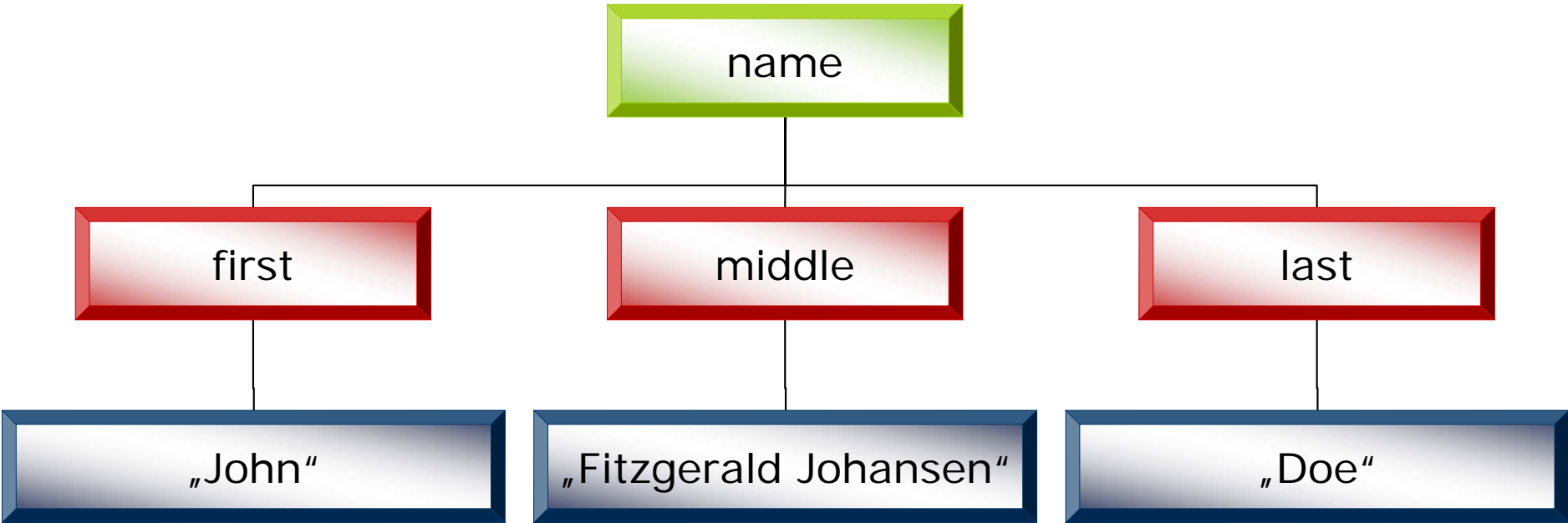
```
<name>  
    <first>John</first>  
    <last>Doe</last>  
</name>
```

- Sequenz von > 0 Kind-Elementen:
- hier: `<first>John</first>` und `<last>Doe</last>`
- kein Text vor, nach oder zwischen den Kind-Elementen
- Kind-Elemente immer geordnet:
Reihenfolge, so wie sie im XML-Dokument erscheinen
- Elemente können beliebig tief geschachtelt werden.



Baumstruktur von XML

```
<name>  
  <first>John</first>  
  <middle>Fitzgerald Johansen</middle>  
  <last>Doe</last>  
</name>
```



3. Gemischter Inhalt

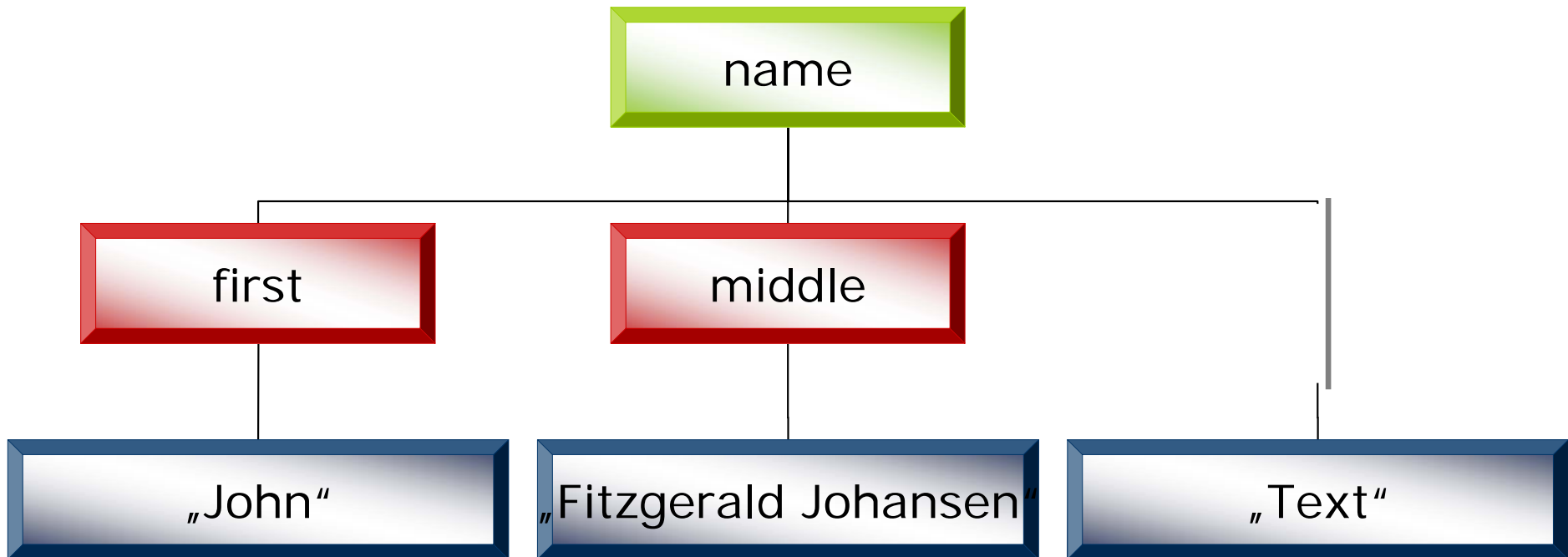
- Englisch: mixed content
- enthält Text mit mindestens einem Kind-Element
- Beispiel:

```
<section>  
Text  
<subsection> ... </subsection>  
Text  
</section>
```

- Wie unterscheidet ein Parser strukturierten und gemischten Inhalt, wenn Text = leerer String?
- Antwort: Nur mit zugehöriger DTD oder XML-Schema möglich!

Baumstruktur von XML

```
<name>  
  <first>John</first>  
  <middle>Fitzgerald Johansen  
  </middle>  
  Text  
</name>
```



4. Leerer Inhalt

- Beispiel:

```
<name>
  <first>John</first>
  <middle></middle>
  <last>Doe</last>
</name>
```

- weder Text noch Kind-Element
- `<middle></middle>` auch **leeres Element** genannt
- Abkürzung: **selbstschießendes Tag** (engl.: *self-closing tag*) `<middle/>` :

```
<name>
  <first>John</first>
  <middle/>
  <last>Doe</last>
</name>
```

Warum leere Elemente?

```
<name>  
  <first>John</first>  
  <last>Doe</last>  
</name>
```

VS.

```
<name>  
  <first>John</first>  
  <middle/>  
  <last>Doe</last>  
</name>
```

- leeres Element evtl. von DTD oder XML-Schema vorgeschrieben
- einfacher später mit Inhalt zu füllen
- leeres Element kann Attribute haben:
 <middle status="unknown"></middle> oder
 <middle status="unknown"/>



XML-Syntax

Grundbausteine: XML-Attribute

```
<name id="1232345" nickname="Shiny John">  
  <first>John</first>  
  <last>Doe</last>  
</name>
```

- **Attribut: Name-Wert-Paar**
 - name="wert" oder name='wert' aber ~~name="wert"~~
- **Attribut-Wert:**
 - immer PCDATA: keine Kind-Elemente, kein < und &
 - "we"rt" und 'we'rt' ebenfalls nicht erlaubt
 - Normalisierung: u.a. Zeilenumbruch → #xA
- **Beachte: Reihenfolge der Attribute belanglos**

Attribut statt Element

- Jedes Attribut auch als Kind-Element darstellbar:

```
<name id="12345">  
  <first>John</first>  
  <middle>Fitzgerald</middle>  
  <last>Doe</last>  
</name>
```



```
<name>  
  <id>12345</id>  
  <first>John</first>  
  <middle>Fitzgerald</middle>  
  <last>Doe</last>  
</name>
```

id als Attribut

id als Kind-Element

Element statt Attribut

- Jedes Kind-Element mit **unstrukturiertem** Inhalt auch als Attribut darstellbar:

```
<name>  
  <id>12345</id>  
  <first>John</first>  
  <middle>Fitzgerald</middle>  
  <last>Doe</last>  
</name>
```

**id, first, middle und last
als Kind-Elemente**



```
<name id="12345"  
  first="John"  
  middle="Fitzgerald"  
  last="Doe" />
```

**id, first, middle und last
als Attribute**

Resultat: leeres Element

Attribut oder Element?

- Attribut kann nur einfachen String (PCDATA) als Wert haben, Element kann beliebig strukturiert sein
- `<![CDATA[...]]>` in Element-Inhalten erlaubt, nicht aber in Attribut-Werten
- Reihenfolge der Attribute belanglos, diejenige von Elementen nicht
- einheitliche Darstellung mit Elementen eleganter, Darstellung mit Attributen kompakter

Fazit: Attribute für einfache, unstrukturierte Zusatzinformationen (Metadaten) geeignet.

```
<name creation-date="21.05.2003">  
  <first>John</first>  
  <middle>Fitzgerald Johansen</middle>  
  <last>Doe</last>  
</name>
```

- Erstellungsdatum creation-date ist Zusatzinformation
 - falls noch andere Attribute vorhanden: Reihenfolge egal
- ⇒ Repräsentation als Attribut
- Nachteil: Datum "21.05.2003" unstrukturiert

Reservierte Attribute

■ `xml:lang`

- Sprache des Inhalts
- Beispiel: `<p xml:lang="de">Übung 1</p>`

■ `xml:space`

- Leerräume im Inhalt
- Beispiel: `<p xml:space="default">Übung 1</p>`



XML-Syntax

Grundbausteine: XML-Deklaration

```
<?xml version="1.0" encoding="UTF-8"?>
<name id="1232345">
  <first>John</first>
  <middle>Fitzgerald Johansen</middle>
  <last>Doe</last>
</name>
```

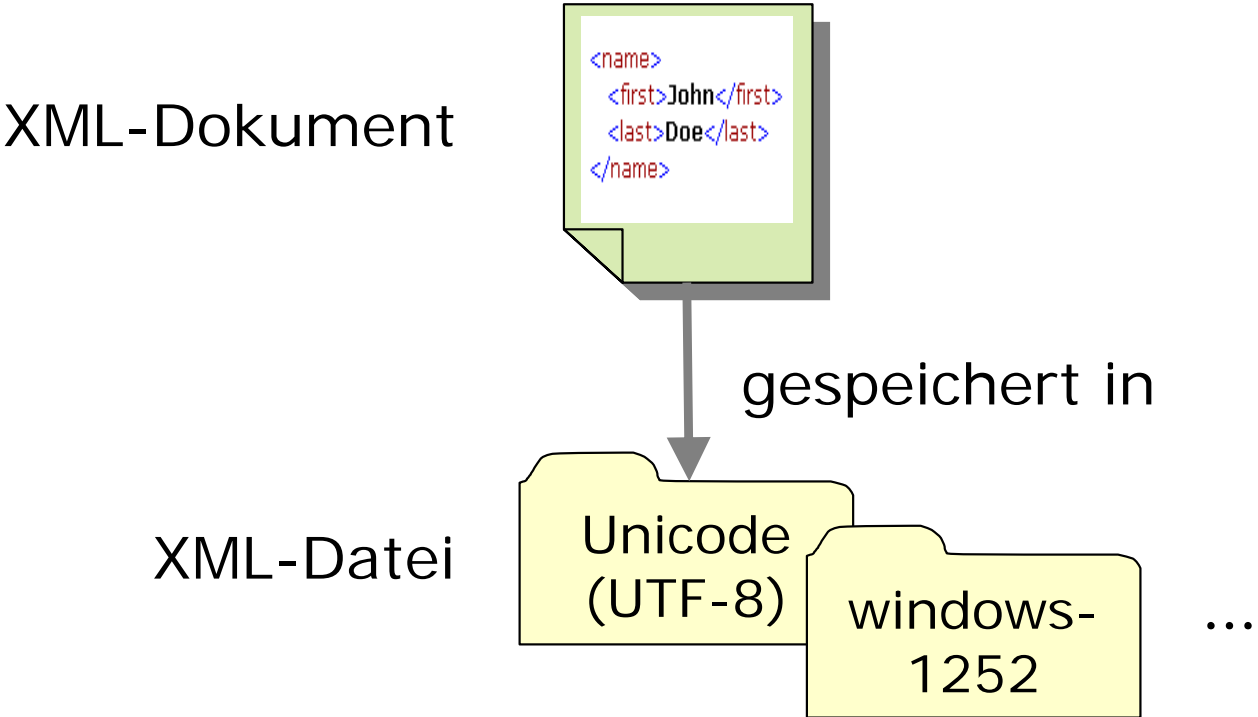
- enthält Informationen für Parser: z.B. verwendete XML-Version und Kodierung
- wenn vorhanden, dann immer am Anfang der Datei
- in XML 1.0 optional, in XML 1.1 obligatorisch

`<?xml version="1.0" encoding="UTF-8" ?>`

- **Attribut: version**
 - verwendete XML-Version: "1.0" oder "1.1"
 - obligatorisch
- **Attribut: encoding**
 - Kodierung der XML-Datei
 - Optional
- **Attribut: standalone**
 - Gibt an, ob es eine zugehörige DTD oder ein XML-Schema gibt ("no") oder nicht ("yes").

Beachte: immer in dieser Reihenfolge!

XML-Deklaration: Kodierung



XML-Deklaration: Kodierung

- XML-Parser
 - müssen intern mit Unicode (UTF-8 oder UTF-16) arbeiten
- Unicode
 - kann alle nationalen Zeichen darstellen: insgesamt ca. 65.000 Zeichen
- encoding-Attribut
 - Zeichenkodierung der XML-Datei
 - Fehlt das Attribut, dann wird Kodierung in Unicode angenommen.

Beachte: XML-Parser müssen nur Unicode verarbeiten können!



XML-Syntax

Andere Grundbausteine

Weitere Grundbausteine von XML

- **Kommentare**
 - `<!-- Kommentar -->`
 - `--` in Kommentaren nicht erlaubt
- **Prozessorinstruktionen**
 - Beispiel: `<?mysql SELECT * FROM PO?>`
 - selten benutzt



XML-Syntax

Wohlgeformte XML-Dokumente

1. Jedes Anfangs-Tag muss ein zugehöriges Ende-Tag haben.
2. Elemente dürfen sich nicht überlappen.
3. XML-Dokumente haben genau ein Wurzel-Element.
4. Element-Namensräume
5. XML beachtet Kleinschreibung
6. XML belässt White Space im Text.
7. Ein Element darf niemals zwei Attribute mit dem selben Namen haben.

Wie kann aus den Grundbausteinen ein wohlgeformtes XML-Dokument gebildet werden?

Regel 1: Anfangs- und Ende-Tags

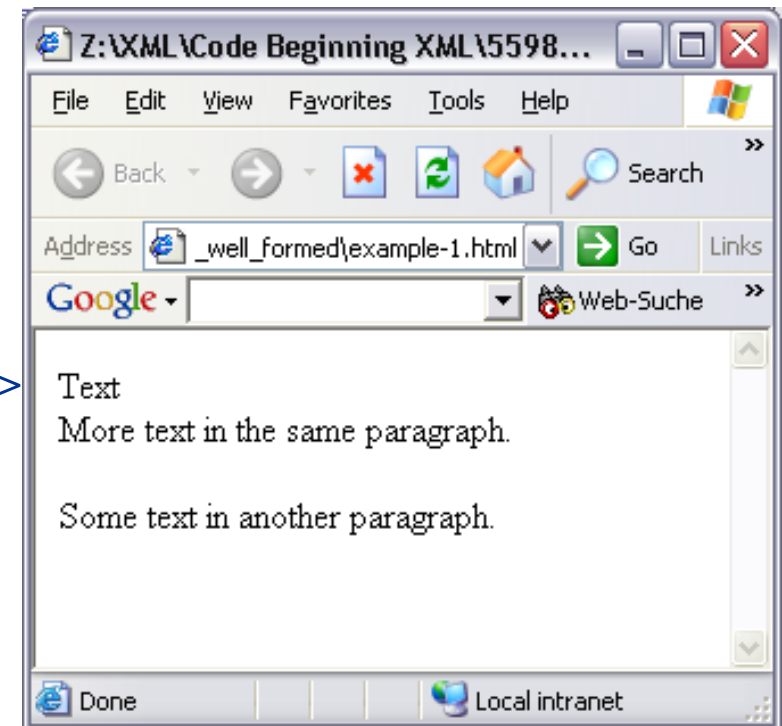
Jedes Anfangs-Tag muss zugehöriges Ende-Tag haben.

- In HTML gilt diese Regel nicht:

```
<HTML>
<BODY>
  <P>Text
  <BR>More text in the same paragraph.
  <P>Some text in another paragraph.</P>
</BODY>
</HTML>
```

- Wo endet das erste P-Element?

⇒ HTML mehrdeutig



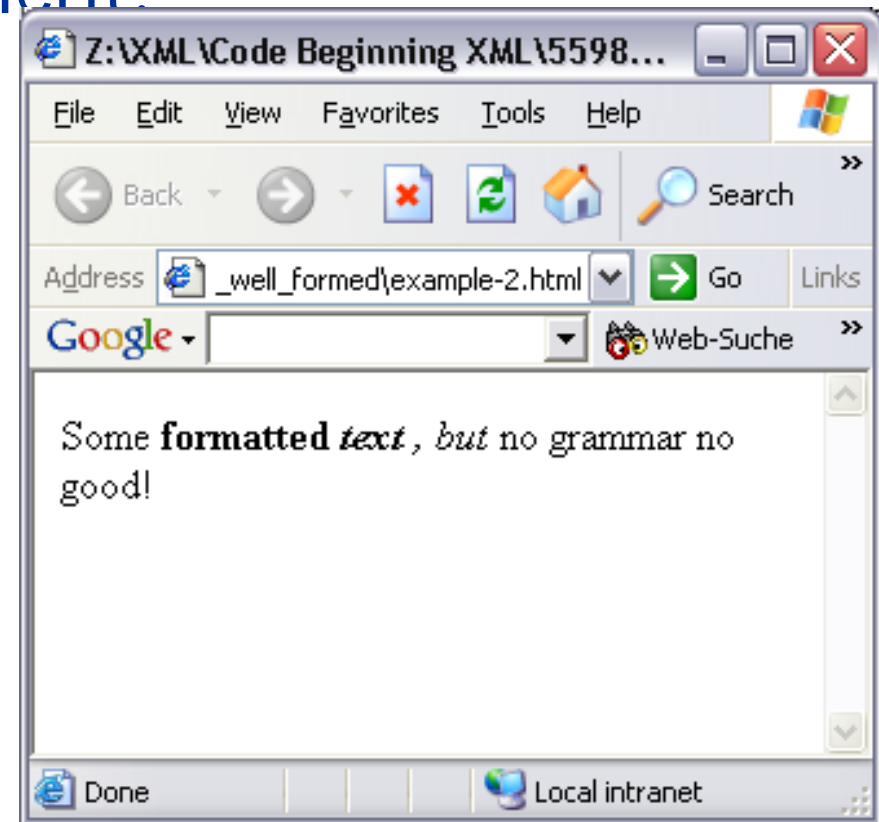
Regel 2: Elemente dürfen sich nicht überlappen

Elemente dürfen sich nicht überlappen.

- In HTML gilt diese Regel nicht:

```
<HTML>
<BODY>
<P>Some
<STRONG>formatted
<EM>text
</STRONG>, but
</EM>
no grammar no good!
</p>
</BODY>
</HTML>
```

⇒ HTML unstrukturiert



Regel 3: Wurzel-Element

Jedes XML-Dokument hat genau ein Wurzel-Element.

- Also z.B. statt zweier Wurzel-Elemente

```
<?xml version="1.0"?>
<name>John</name>
<name>Jane</name>
```

- zusätzliches Eltern-Element einführen:

```
<?xml version="1.0"?>
<names>
  <name>John</name>
  <name>Jane</name>
</names>
```

Regel 4: Namenskonventionen

Element- und Attribut-Namen:

- beginnen entweder mit Buchstaben oder `_` aber nie mit Zahlen:
z.B. `first`, `First` oder `_First`
- nach erstem Zeichen zusätzlich Zahlen sowie `-` und `.` erlaubt:
z.B. `_1st-name` oder `_1st.name`
- enthalten keine Leerzeichen
- enthalten kein Doppelpunkt
- beginnen nicht mit `xml`, unabhängig davon, ob die einzelnen Buchstaben groß- oder kleingeschrieben sind

Namenskonvention: Beispiele

- `<résumé>` ✓
- `<xml-tag>` nicht korrekt: beginnt mit „xml“
- `<123>` nicht korrekt: beginnt mit Zahl
- `<fun=xml>` nicht korrekt: enthält „=“
erlaubt wären: `_`, `-` und `.`
- `<first name>` nicht korrekt: enthält Leerzeichen

Regel 5: Groß- und Kleinschreibung

XML beachtet Groß- und Kleinschreibung.

- Im Gegensatz zu HTML unterscheidet XML also z.B. zwischen `<P>` und `<p>`.

Dennoch möglichst nicht gleichzeitig
`<First>` und `<first>` verwenden!

Hinweis: eine Schreibweise im gesamten
Dokument verwenden z.B. `<FirstName>`

Regel 6: White Space

XML belässt White Space im Text.

- Beispiel:

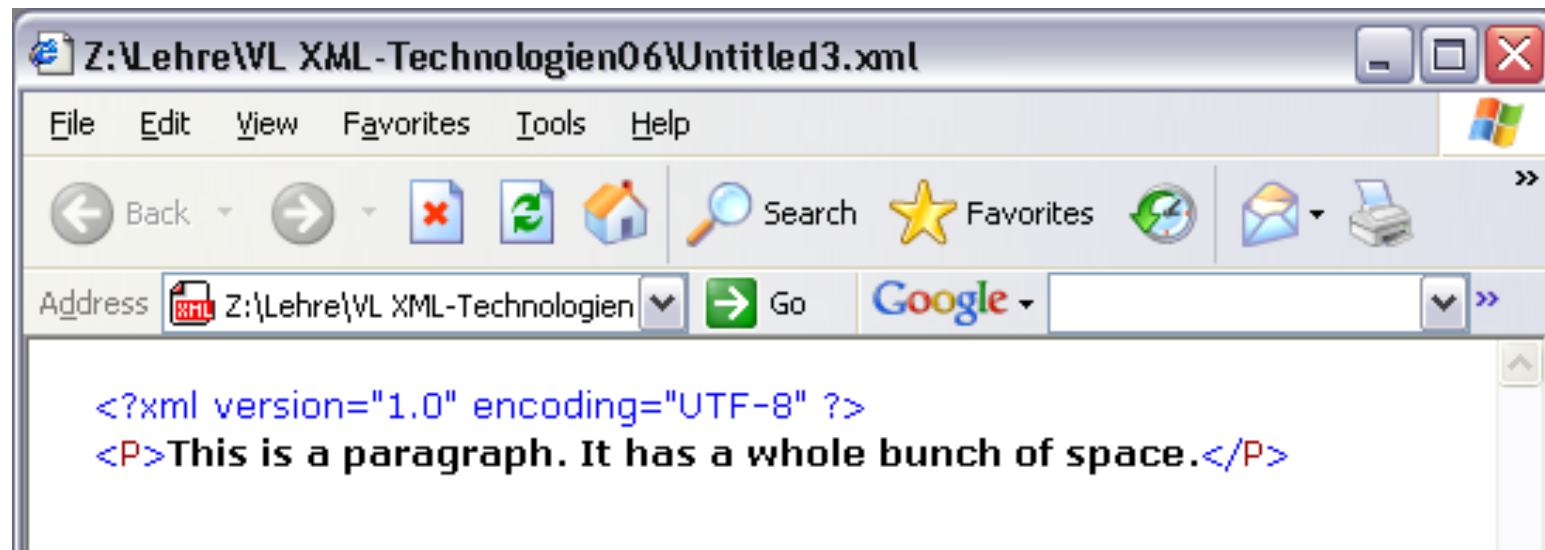
```
<?xml version="1.0" encoding="UTF-8" >  
<P>This is a paragraph.           It has a whole bunch  
  of space.</P>
```

Inhalt des P-Elementes:

```
This is a paragraph.           It has a whole  
  bunch  
  of space.
```

Regel 6: White Space

- Beachte: Von Browsern wird White Space allerdings nicht angezeigt:



- Grund:
 - XML-Dokumente werden zur Darstellung im Browser in HTML umgewandelt
 - HTML reduziert White Space auf ein Leerzeichen

	XML 1.0	XML 1.1
XML-Deklaration	optional	obligatorisch
Zeilenende (EoL)	LF, CR	LF, CR NEL (IBM-Mainframe)
Unicode in Namen	exklusive Strategie: alle Zeichen verboten, die nicht ausdrücklich erlaubt	inklusive Strategie: alle Zeichen erlaubt, die nicht ausdrücklich verboten

- ⇒ XML 1.1 interoperabel mit IBM-Mainframes
- ⇒ XML 1.1 aufwärtskompatibel bzgl. zukünftiger Unicode-Versionen
- ⇒ XML 1.1 nicht abwärtskompatibel!

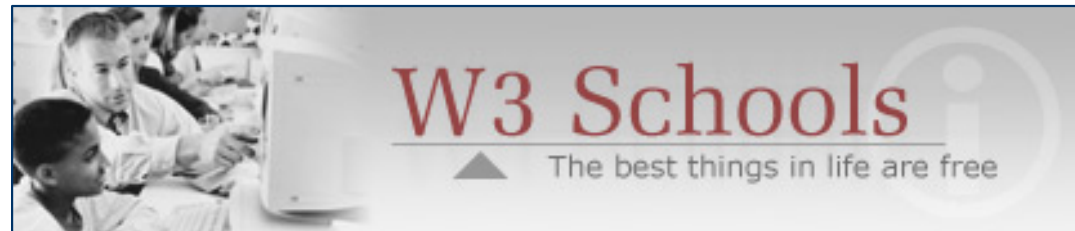
Viel Lärm um XML 1.1!

- XML 1.0, 3rd Edition zeitgleich mit XML 1.1 veröffentlicht
 - 3rd Edition = 2nd Edition + Errata zur 2nd Edition
- Änderungen von XML 1.1 nur für IBM-Mainframes und Nicht-ASCII-Zeichen relevant
- XML 1.1 verlangt von Parsern, dass beide Versionen erkannt werden:
 - wenn keine XML-Deklaration oder explizit Version 1.0:
 - Wohlgeformtheit gemäß XML 1.0
 - in allen anderen Fällen:
 - Wohlgeformtheit gemäß XML 1.1
- Mittlerweile:
 - XML 1.0, 4rd Edition
 - XML 1.1, 2nd Edition

- beschreibt das XML-Datenmodell unabhängig von der konkreten XML-Syntax
- allerdings ziemlich umständlich:

“Rather than saying “‘foo’ has a ‘quantity’ of ‘3’” you have to say “the element information item with the [local name] property ‘foo’ has an attribute information item in its [attributes] property with the [local name] property ‘quantity’ and the [normalized value] property ‘3’.” (Nottingham, 2004)

⇒ <http://www.w3.org/TR/2004/REC-xml-infoset-20040204/>



- W3 Schools: kostenlose Online-Tutorials zu XML-Technologien

⇒ Online-Test

http://www.w3schools.com/xml/xml_quiz.asp

- XML-Dokumente werden normalerweise mit speziellen Editoren erstellt und modifiziert.
 - meistbenutzte XML-Editor: XMLSpy von Altova
 - steht in den PC-Pools zur Verfügung
 - Home Edition kostenlos
 - Enterprise Edition als vierwöchige Testlizenz kostenlos
- ➔ www.xmlspy.com





Namensräume



```
<course>
  <title>Semantic Web</title>
  <lecturers>
    <name>
      <title>Priv.-Doz. Dr. M.S.E.</title>
      <first>Steffen</first>
      <last>Staab</last>
    </name>
  </lecturers>
  <date>12/11/2004</date>
  <abstract>...</abstract>
</course>
```

- **Namenskonflikt:** gleicher Name, aber unterschiedliche Bedeutung
- z.B. Titel einer Veranstaltung vs. Titel einer Person
- in einem Dokument unterschiedliche Vokabularien

Auflösung durch Präfixe

```
<course:course>
  <course:title> Semantic Web </course:title>
  <course:lecturers>
    <pers:name>
      <pers:title> Priv.-Doz. Dr. M.S.E </pers:title>
      <pers:first> Steffen </pers:first>
      <pers:last> Staab </pers:last>
    </pers:name>
  </course:lecturers>
  <course:date> 12/11/2004 </course:date>
  <course:abstract> ... </course:abstract>
</course:course>
```

- Präfixe geben Kontext an: Aus welchem Bereich stammt der Name
 - z.B. pers:title vs. course:title
- ähnliches Vorgehen in Programmiersprachen:
 - z.B. java.applet.Applet

{
course:course
course:title course:abstract
course:lecturers
course:date
}

{
pers:name
pers:title pers:first
pers:last
}

Namensraum (namespace):

- alle Bezeichner mit identischen Anwendungskontext
- Namensräume müssen eindeutig identifizierbar sein.

Namensräume in XML

- WWW: Namensräume müssen global eindeutig sein.
- In XML wird Namensraum mit URI identifiziert.
- Zuerst wird Präfix bestimmter Namensraum zugeordnet, z.B.:



- Anschließend kann das Namensraum-Präfix einem Namen vorangestellt werden: z.B. `pers:title`
- Beachte: Wahl des Präfixes egal!

```
<course:course xmlns:course="http://www.w3.org/2004/course">
  <course:title>Semantic Web</course:title>
  <course:lecturers>
    <pers:name xmlns:pers="http://www.w3.org/2004/pers">
      <pers:title>Priv.-Doz. Dr. M.S.E</pers:title>
      <pers:first>Steffen</pers:first>
      <pers:last>Staab</pers:last>
    </pers:name>
  </course:lecturers>
  <course:date>12/11/2004</course:date>
  <course:abstract>...</course:abstract>
</course:course>
```

Standard-Namensraum

- **xmlns="URI"** statt xmlns:prefix="URI",
- Namensraum-Präfix kann weggelassen werden.
- Standard-Namensraum gilt für das Element, wo er definiert ist.
- Kind-Elemente erben Standard-Namensraum von ihrem Eltern-Element.
- Ausnahme: Standard-Namensraum wird überschrieben
- Beachte: Standardnamensräume gelten nicht für Attribute

```
<course:course xmlns:course="http://www.w3.org/2004/course">  
  <course:title>Semantic Web</course:title>  
  <course:lecturers>  
    <pers:name xmlns:pers="http://www.w3.org/2004/pers">  
      <pers:title>Priv.-Doz. Dr. M.S.E</pers:title>  
      <pers:first>Steffen</pers:first>  
      <pers:last>Staab</pers:last>  
    </pers:name>  
  </course:lecturers>  
  <course:date>12/11/2004</course:date>  
  <course:abstract>...</course:abstract>  
</course:course>
```

```
<course xmlns="http://www.w3.org/2004/course" >
  <title>Semantic Web</title>
  <lecturers>
    <pers:name xmlns:pers="http://www.w3.org/2004/pers" >
      <pers:title>Priv.-Doz. Dr. M.S.E</pers:title>
      <pers:first>Steffen</pers:first>
      <pers:last>Staab</pers:last>
    </pers:name>
  </lecturers>
  <date>12/11/2004</date>
  <abstract>...</abstract>
</course>
```

```
<course xmlns="http://www.w3.org/2004/course">  
  <title>Semantic Web</title>  
  <lecturers>  
    <name xmlns:pers="http://www.w3.org/2004/pers">  
      <title>Priv.-Doz. Dr. M.S.E</title>  
      <first>Steffen</first>  
      <last>Staab</last>  
    </name>  
  </lecturers>  
  <date>12/11/2004</date>  
  <abstract>...</abstract>  
</course>
```

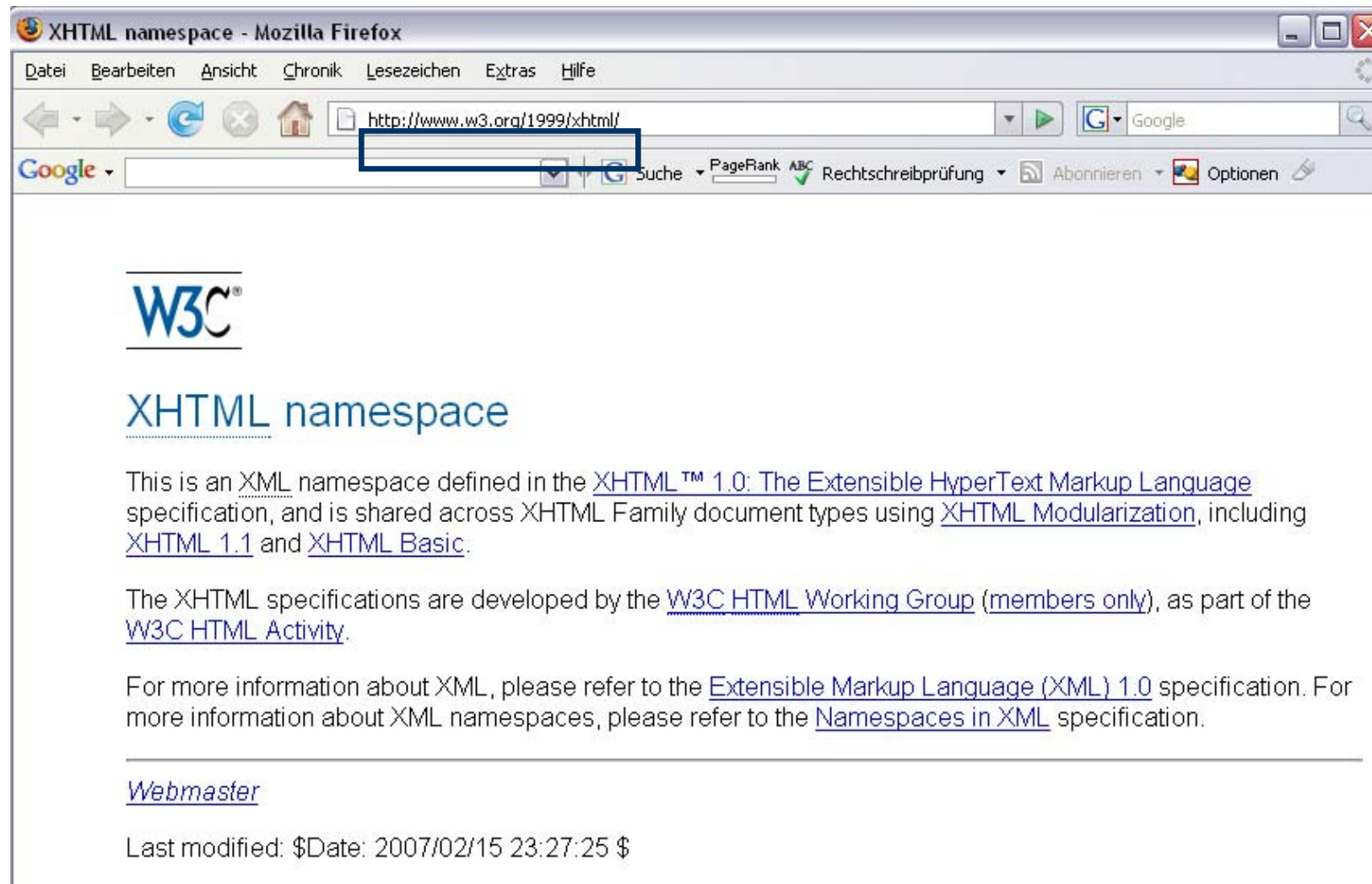
```
<course xmlns="http://www.w3.org/2004/course" >
  <title>Semantic Web</title>
  <lecturers>
    <name xmlns="http://www.w3.org/2004/pers" >
      <title>Priv.-Doz. Dr. M.S.E</title>
      <first>Steffen</first>
      <last>Staab</last>
    </name>
  </lecturers>
  <date>12/11/2004</date>
  <abstract>...</abstract>
</course>
```

- eindeutige Bezeichner für Ressourcen im WWW
- URI kann den physischen Aufenthaltsort einer Ressource beschreiben

<http://www.w3.org/1999/xhtml>

- Solche URIs werden auch Uniform Resource Locators (URLs) genannt.

- Beispiel: <http://www.w3.org/1999/xhtml> bezeichnet den Namensraum für XHTML



URIs als Namensraum-Bezeichner

- URI kann (muss aber nicht) Beschreibung des Namensraumes enthalten:
 - z.B. XML-Schema oder Spezifikationen
- URI muss nicht einmal existieren!
- Allerdings ist nur bei existierenden URIs Eindeutigkeit sichergestellt.

```
<Book xmlns="http://www.books-ns.org" >  
  ...  
</Book >
```

- <http://www.book-ns.org> existiert (noch) nicht
 - keine Fehlermeldung, keine Warnung von XML-Parser oder XML-Editor
 - dennoch Eindeutigkeit nicht sichergestellt:
jemand anderes kann gleiche URL für anderen Namensraum verwenden
- ⇒ neue Namensräume nur mit URIs bezeichnen, die man selbst besitzt

Qualified vs. Unqualified

- Element- oder Attribut-Name heißt namensraumeingeschränkt (qualified), wenn er einem Namensraum zugeordnet ist.
- Einschränkung vom Element-Namensraum:
 1. Standard-Namensraum festlegen
 2. Namensraum-Präfix voranstellen
- Einschränkung vom Attribut-Namensraum:
 1. Namensraum-Präfix voranstellen

Was bedeutet `<p>...</p>`?

- **HTML:**
 - Bedeutung festgelegt (p = Absatz)
- **XML:**
 - Bedeutung offen
 - kann aber mit **Namensraum** festgelegt werden
 - Beispiel: p stammt aus dem Namensraum für XHTML.

xhtml Abkürzung für Namensraum

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

Namensraum:
u.a. p = Absatz

Und das war es schon?

- Ja!
- Syntax wohlgeformter XML-Dokumente (fast) vollständig vorgestellt
- XML-Syntax also sehr einfach
- gleichzeitig ist XML beliebig erweiterbar
- Und das ist genau die Stärke von XML: einfach und flexibel!

- ☑ XML-Syntax
- ☑ Namensräume

- Definition von XML-Sprachen mit DTDs und XML-Schema

- 1. Übungsblatt