
Aufbau von XML-Dokumenten

Heutige Vorlesung

- XML-Syntax
- Namensräume
- Semantik von XML-Elementen

Wiederholung: Was ist XML?



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



- XML sieht fast aus wie HTML, ist aber kein HTML.



- XML ist Text, aber nicht zum Lesen.



- XML ist eine Familie von Techniken.



- XML ist neu, aber nicht so neu.

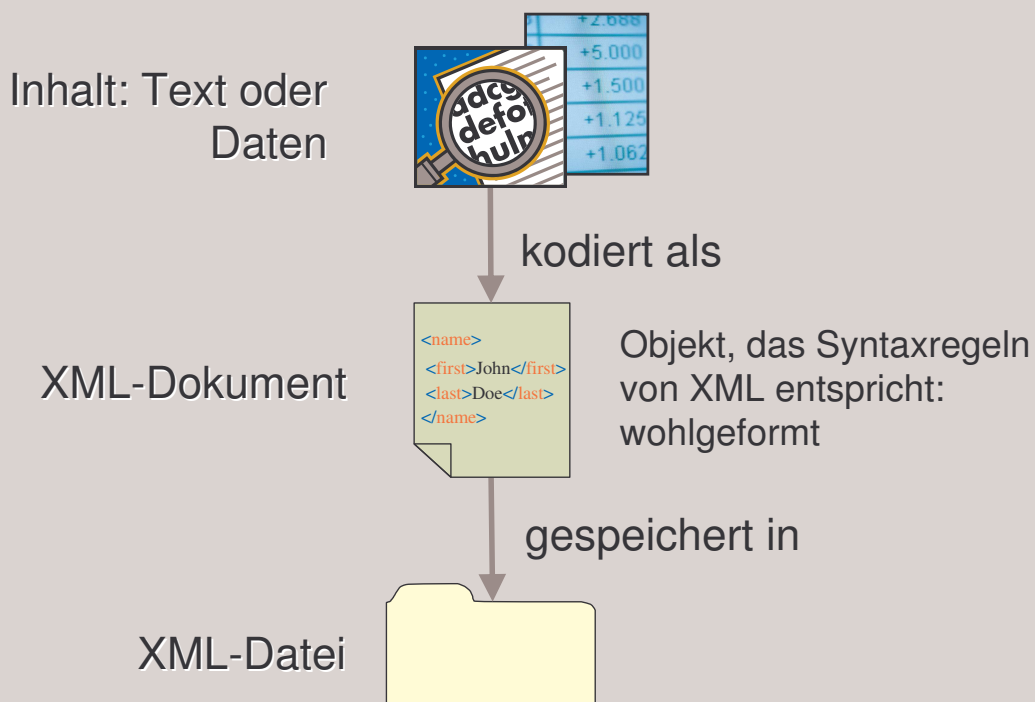


- XML ist lizenzfrei und plattformunabhängig.



XML-Syntax: Wohlgeformte XML-Dokumente

Was ist ein XML-Dokument?



Grundbausteine von XML

- **Elemente:** strukturieren das XML-Dokument
- **Attribute:** Zusatzinformationen zu einzelnen 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 Elemente eine bestimmte Bedeutung

- Beispiel: `<first>John</first>`
- besteht aus:
 - einem **Anfangs-Tag** (engl. *start tag*): hier `<first>`
 - einem dazugehörigen **Ende-Tag** (engl. *end tag*): hier `</first>`
 - einem **Inhalt**: hier „John“
- haben einen **Namen**: hier „first“
- alles zusammen bildet ein **Element**: `<first>John</first>`

vier verschiedene Arten von Inhalt

1. **unstrukturierter Inhalt**
2. **strukturierter Inhalt**
3. **gemischter Inhalt**
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.

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

PCDATA



- Reservierte Symbole < und & in PCDATA *nicht* erlaubt.
- Symbole >, / (,), [,], % etc. hingegen erlaubt.
- statt < und & ***Entity References*** & bzw. < benutzen
- Entity References in XML:
 - & ⇒ &
 - < ⇒ <
 - > ⇒ >
 - ' ⇒ '
 - " ⇒ "

- unstrukturierter Inhalt mit vielen reservierten Symbolen besser als sog. **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 ansonsten nicht.

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
- Elemente können beliebig tief geschachtelt werden.

3. Gemischter Inhalt (*mixed content*)



- enthält Text mit mind. einem Kind-Element
- Beispiel:

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

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 Element** `<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>
```

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

Grundbausteine von XML: Attribute



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

- Element kann eine beliebige Anzahl von Attributen haben.
- **Attribut:** Name-Wert-Paar der Form `name="wert"` oder `name='wert'`
- Wert *immer* vom Typ PCDATA: `<` und `&` *nicht* erlaubt
- ebenfalls `'`, `"` und CDATA *nicht* erlaubt
- Beachte: Reihenfolge der Attribute belanglos

Attribut statt Element



- Jedes Attribut auch als Kind-Element darstellbar:

```
<name id="1232345">
  <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>
```



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

id, first, middle und last
als Kind-Elemente

id, first, middle und last als
Attribute

Resultat: leeres Element

Attribut oder Element?



- Attribut kann nur einfachen String (PCDATA) als Wert haben, ein Element kann beliebig strukturiert sein
- CDATA in Element-Inhalt erlaubt, in Attribut-Wert *nicht* möglich
- Reihenfolge der Attribute belanglos, diejenige von Elementen nicht
- Einheitliche Darstellung mit Elementen eleganter, Darstellung mit Attributen kompakter

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

Beispiel



```
<name id="1232345" creation-date=" 21.05.2003 " >  
  <first>John</first>  
  <middle>Fitzgerald Johansen</middle>  
  <last>Doe</last>  
</name>
```

- Schlüssel des Datensatzes (id) und Erstellungsdatum (creation-date) sind Zusatzinformationen.
- Reihenfolge egal
- deshalb jeweils Repräsentation als Attribut
- Nachteil: Datum "21.05.2003" unstrukturiert

```
<?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
- muss *immer* am Anfang der Datei stehen

XML-Deklaration



Attribut version

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

- verwendete XML-Version
- mögliche Versionen: "1.0" und "1.1"
- obligatorisch

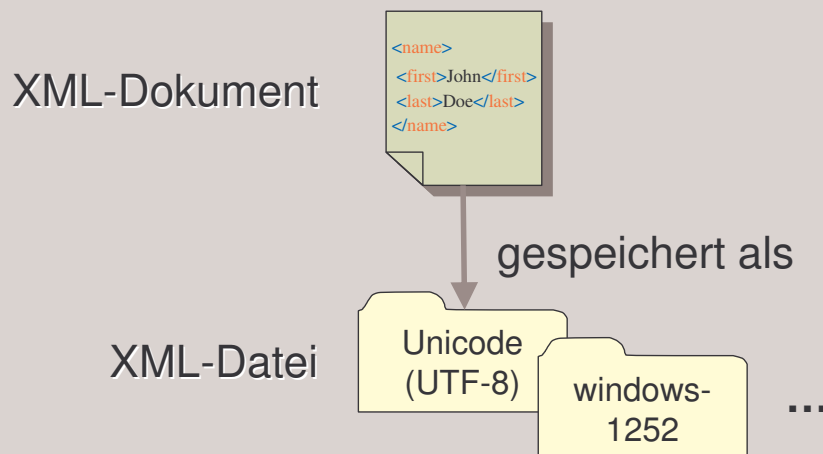
Attribut standalone

- Gibt an, ob es eine zugehörige DTD oder ein XML-Schema gibt ("no") oder nicht ("yes").
- optional

Attribut encoding

- Kodierung der XML-Datei
- optional

Beachte: Attribute immer in dieser Reihenfolge!



XML-Parser

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

- müssen gemäß XML-Spezifikation intern mit Unicode (UTF-8 oder UTF-16) arbeiten

Unicode

- kann *alle* nationalen Zeichen darstellen: insgesamt ca. 65.000 Zeichen

encoding-Attribut

- Zeichenkodierung der betreffenden XML-Datei
- Fehlt das Attribut, dann wird Kodierung in Unicode angenommen.
- Beachte: XML-Parser müssen gemäß XML-Spezifikation nur Unicode verarbeiten können!

1. Jedes Anfangs-Tag muss ein zugehöriges Ende-Tag haben.
2. Elemente dürfen sich nicht überlappen.
3. XML-Dokumente müssen ein einziges Wurzel-Element haben.
4. Element-Namen müssen die Namenskonventionen von XML befolgen.
5. XML beachtet die Unterscheidung zwischen Groß- und Kleinschreibung.
6. XML belässt unsichtbare Zeichen 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

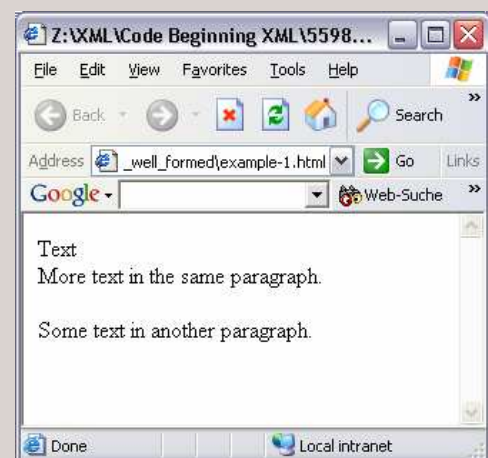
- Jedes Anfangs-Tag muss ein zugehöriges Ende-Tag haben.

- 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: Überlappung von Elementen



- 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-Elemente



- 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>
```

oder

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

Regel 4: Namenskonventionen



Element- und Attribut-Namen:

- beginnen entweder mit einem Buchstaben (Unicode) oder „_“:
z.B. first, First oder _First
- Nach dem ersten Zeichen zusätzlich Zahlen sowie „-“ und „.“ erlaubt:
z.B. _1st-name oder _1st.name
- enthalten keine Leerzeichen
- enthalten kein „:“
- beginnen nicht mit „xml“, unabhängig davon, ob einzelne Buchstaben groß- oder kleingeschrieben

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



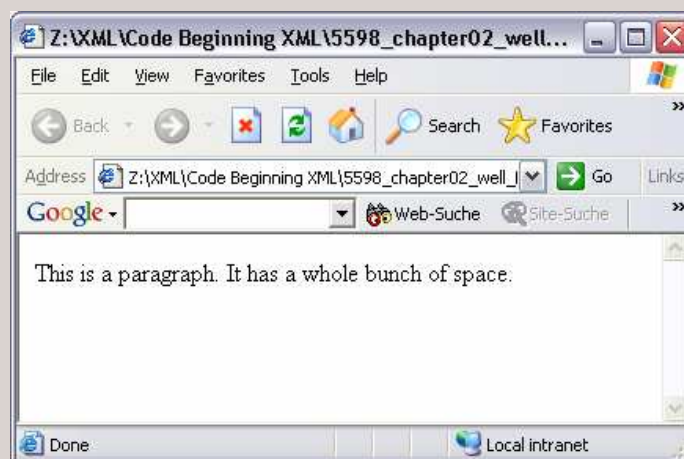
- XML beachtet grundsätzlich Groß- und Kleinschreibung.
- Im Gegensatz zu HTML unterscheidet XML also z.B. zwischen `<P>` und `<p>`.

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

Regel 6: Unsichtbare Zeichen



- Beispiel: `<P>`This is a paragraph. It has a whole bunch of space.`</P>`
- **HTML** reduziert unsichtbare Zeichen (Leerzeichen, CR, LF und Tab, engl. *white spaces*) auf ein Leerzeichen



Regel 6: Unsichtbare Zeichen



- XML belässt alle unsichtbaren Zeichen im Text.
- Beispiel: Der Inhalt von

```
<P>This is a paragraph.           It has a whole bunch  
of space.</P>
```

ist also:

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

- Beachte: Von Browsern werden unsichtbare Zeichen allerdings *nicht* angezeigt.
- Grund: XML-Dokumente werden zur Darstellung im Browser in HTML umgewandelt.

Was gibt es sonst noch?



- Attribut: z.B. `xml:lang="fr"`
- Prozessorinstruktionen: z.B. `<?mySystem value="42"?>`
- Kommentare: z.B. `<!-- Kommentar -->`
- welche **Unicode**-Zeichen, wo erlaubt sind
- Namensräume: z.B. `<xhtml:p>...</xhtml:p>`

XML 1.0 vs. XML 1.1



	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 zu zukünftige Unicode-Versionen
- ⇒ XML 1.1 nicht abwärtskompatibel!

Viel Lärm um XML 1.1!



- **XML 1.0, 3rd Edition** zeitgleich mit XML 1.1 veröffentlicht!
- Änderungen in XML 1.1 nur für IBM-Mainframes und Nicht-ASCII-Zeichen relevant
- XML 1.1 verlangt von Parsern, dass beide Versionen erkannt und unterschiedlich behandelt werden:
- wenn keine XML-Deklaration oder explizit Version 1.0:
 - Wohlgeformtheit gemäß XML 1.0
- in allen anderen Fällen:
 - Wohlgeformtheit gemäß XML 1.1



- ⇒ http://www.w3schools.com/xml/xml_quiz.asp
- W3 Schools: kostenlose Online-Tutorials zu XML-Technologien

XML-Editoren

- XML-Dokumente werden normalerweise mit speziellen Editoren erstellt und modifiziert.
 - meistbenutzter XML-Editor ist XML Spy:
 - unterstützt XML-Schemata, XSLT (mit Debugger), Web-Services, Import und Export von Datenbank-Schemata
 - steht in den PC-Pools zur Verfügung
 - gibt es aber auch als kostenlose vierwöchige Testlizenz
- ➔ www.xmlspy.com



Namensräume

Namenskonflikte

```
<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: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
- Auf diese Weise werden auch Namenskonflikte in Programmiersprachen aufgelöst:
z.B. java.applet.Applet

Namensräume



{
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.

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

```
xmlns:pers="http://www.w3.org/2004/pers"
```

Namensraum-Präfix

Namensraum-Bezeichner (URI)

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

Beispiel



```
<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>
```

Exkurs: Uniform Resource Identifier (URI)



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

`http://www.w3.org/1999/xhtml`

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

Uniform Resource Identifier (URI)



- URI kann auch ein Namen einer Resource unabhängig von deren physischen Aufenthaltsort sein:

`urn:oasis:names:specification:docbook:dtd:xml:4.1.2`

`urn:isbn:1-861005-59-8`

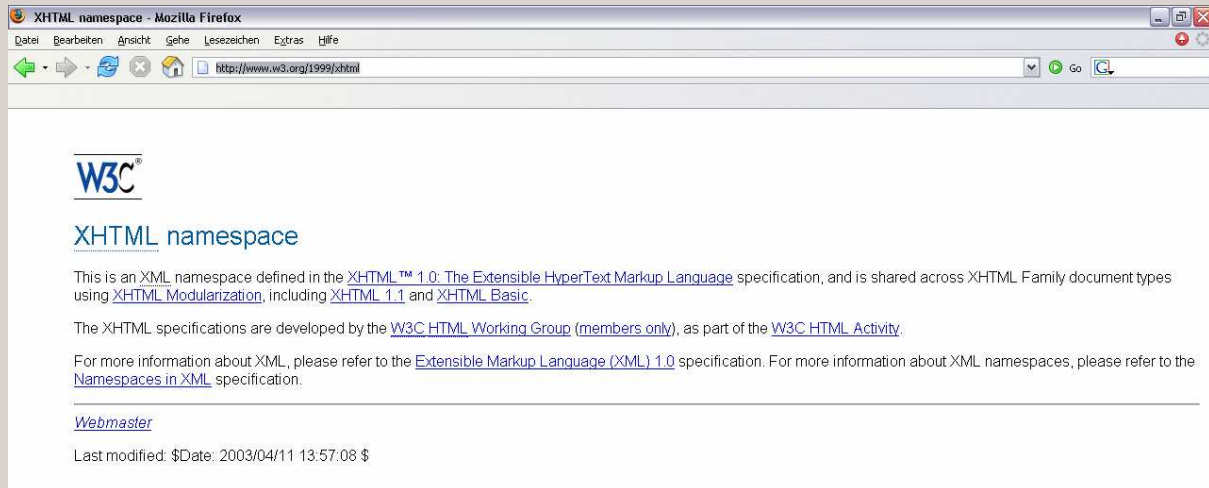
- `urn:oasis` und `urn:isbn` werden URI-Schemata (*URI schemes*) genannt.
- URI-Schemata können bei der IANA registriert werden:
 - genaue Festlegung der Syntax
 - Wer vergibt die dazugehörigen Namen?



URIs als Namensraum-Bezeichner



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



URIs als Namensraum-Bezeichner



- Namensraum-URI *kann* (muss aber nicht) Beschreibung des Namensraumes enthalten.
- Namensraum-URI muss nicht einmal existieren!

URLs als Namensraum-Bezeichner



- Beispiel: `xmlns:myns="http://www.book-ns.org"`
- `http://www.book-ns.org` existiert (noch) nicht
- dennoch *kein* Fehler, *keine* Warnung, falls als Namensraum-URL verwendet!
- Eindeutigkeit dann jedoch *nicht* sicher gestellt:
- Jemand anderes kann *gleiche* Namensraum-URL für *anderen* Namensraum verwenden.
- Für eigene Namensräume also am besten nur URLs verwenden, die man selbst besitzt.

Standard-Namensraum



```
<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>
```

- es kann auch ein **Standard-Namensraum** definiert werden: `xmlns="URI"` statt `xmlns:prefix="URI"`
- Namensraum-Präfix kann dann weggelassen werden.

Standard-Namensräume überschreiben



```
<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>
```

1. Standard-Namensraum

2. Standard-Namensraum

Gültigkeit von Standard-Namensräumen



```
<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>
```

- Standard-Namensraum gilt für das Element, wo er definiert ist.
- Kind-Elemente erben Standard-Namensraum von ihrem Eltern-Element.
- Ausnahme: Standard-Namensraum wird überschrieben

Qualified vs. Unqualified



- Element-Name heißt **namensraumeingeschränkt** (*qualified*), wenn er einem Namensraum zugeordnet ist.
- Zwei Möglichkeiten, diese Zuordnung vorzunehmen:
 1. Standard-Namensraum festlegen
 2. Namensraum-Präfix voranstellen

Beispiel 1



```
<?xml version="1.0"?>
<BookStore xmlns="http://www.books.org">
  <Book>
    <Title>My Life and Times</Title>
    <Author>Paul McCartney</Author>
    <Date>July, 1998</Date>
    <ISBN>94303-12021-43892</ISBN>
    <Publisher>McMillin Publishing</Publisher>
  </Book>
</BookStore>
```

- alle Element-Namen (einschl. BookStore!) dem Standard-Namensraum zugeordnet
- alle Element-Namen daher namensraumeingeschränkt (*qualified*)

Beispiel 2



```
<?xml version="1.0"?>
<bk:BookStore xmlns:bk="http://www.books.org">
  <bk:Book>
    <bk:Title>My Life and Times</bk:Title>
    <bk:Author>Paul McCartney</bk:Author>
    <bk:Date>July, 1998</bk:Date>
    <bk:ISBN>94303-12021-43892</bk:ISBN>
    <bk:Publisher>McMillin Publishing</bk:Publisher>
  </bk:Book>
</bk:BookStore>
```

- alle Element-Namen haben Namensraum-Präfix
- alle Element-Namen daher namensraumeingschränkt (*qualified*).

Beispiel 3



```
<?xml version="1.0"?>
<bk:BookStore xmlns:bk="http://www.books.org">
  <Book>
    <Title>My Life and Times</Title>
    <Author>Paul McCartney</Author>
    <Date>July, 1998</Date>
    <ISBN>94303-12021-43892</ISBN>
    <Publisher>McMillin Publishing</Publisher>
  </Book>
</bk:BookStore>
```

- hier kein Standard-Namensraum festgelegt
- bk:Bookstore: namensraumeingeschränkt (*qualified*)
- alle anderen Element-Namen: *ohne* Namensraum-Präfix, daher nicht namensraumeingeschränkt (*unqualified*)

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 ist 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
- Ausnahmen: Prozessorinstruktionen <?.....?> und erlaubte Unicode-Zeichen
- XML-Syntax also sehr einfach
- gleichzeitig ist XML beliebig erweiterbar
- Und das ist genau die Stärke von XML: einfach und flexibel!

Wie geht es weiter?



- ☑ XML-Syntax
- ☑ Namensräume
- ☑ Semantik von XML-Elementen
 - Definition von XML-Sprachen mit DTDs und XML-Schema