Kapitel WT:III (Fortsetzung)

III. Dokumentsprachen

- Auszeichnungssprachen
- HTML
- Cascading Stylesheets CSS
- XML-Grundlagen
- XML-Schema
- Die XSL-Familie
- APIs für XML-Dokumente

XML-Grundlagen

```
<!DOCTYPE html>

<html>
  <head>
    <meta charset="utf-8">
    <title>Person</title>
  </head>

  <body>
    <h3>Alan Turing</h3>
    <p>
      23. Juni 1912* <br>
      Mathematiker, Informatiker
    </p>
  </body>
</html>
```

XML-Grundlagen

```
<!DOCTYPE html>

<html>
  <head>
    <meta charset="utf-8">
    <title>Person</title>
  </head>

  <body>
    <h3>Alan Turing</h3>
    <p>
      23. Juni 1912* <br>
      Mathematiker, Informatiker
    </p>
  </body>
</html>
```

```
<?xml version="1.0" standalone="no" ?>
<!DOCTYPE person SYSTEM "person.dtd">

<person>
  <name>
    <vorname>Alan</vorname>
    <nachname>Turing</nachname>
  </name>

  <geburtstag>23. Juni 1912</geburtstag>

  <beruf>Mathematiker</beruf>
  <beruf>Informatiker</beruf>
</person>
```

XML-Grundlagen

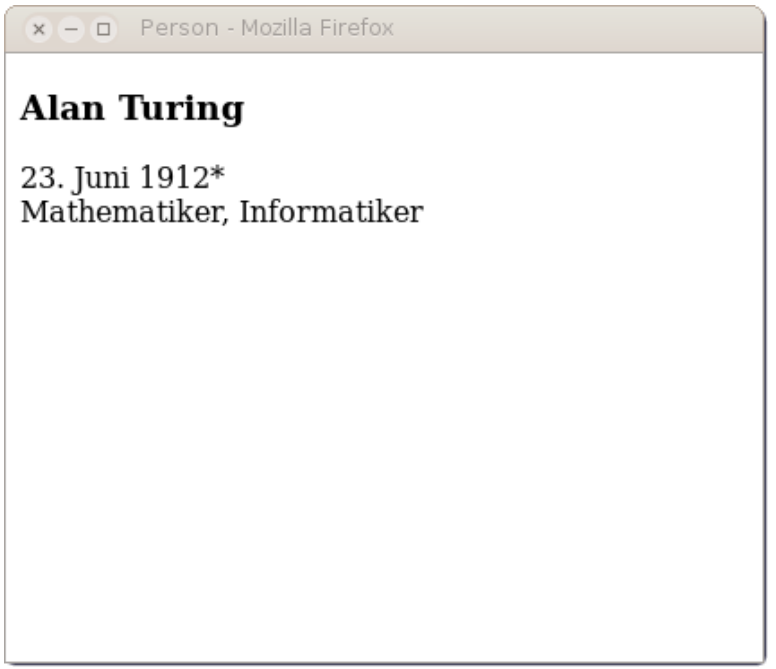
```
<!DOCTYPE html>

<html>
  <head>
    <meta charset="utf-8">
    <title>Person</title>
  </head>

  <body>
    <h3>Alan Turing</h3>
    <p>
      23. Juni 1912* <br>
      Mathematiker, Informatiker
    </p>
  </body>
</html>
```

```
<?xml version="1.0" standalone="no" ?>
<!DOCTYPE person SYSTEM "person.dtd">

<person>
  <name>
    <vorname>Alan</vorname>
    <nachname>Turing</nachname>
  </name>

  <geburtstag>23. Juni 1912</geburtstag>

  <beruf>Mathematiker</beruf>
  <beruf>Informatiker</beruf>
</person>
```

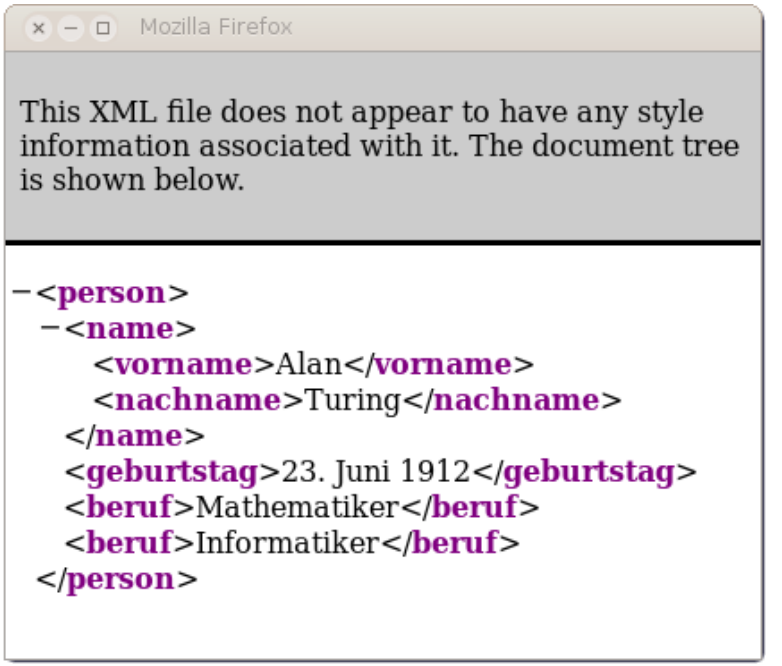
Keine operationale Semantik:

```
<?xml version="1.0" standalone="no" ?>
<!DOCTYPE person SYSTEM "person.dtd">

<person>
  <address>
    <zip>Alan</zip>
    <city>Turing</city>
  </address>

  <weight>23. Juni 1912</weight>

  <name>Mathematiker</name>
  <name>Informatiker</name>
</person>
```

XML-Grundlagen



[person.html]



[person.xml]

In XML können beliebige Elementtypen definiert und deklariert werden. Mittels einer DTD *(Document Type Definition)* lassen sich strukturelle Constraints (= Inhaltsmodelle) für die Verwendung von Elementinstanzen vorschreiben.

Bemerkungen:

- XML kompakt:
  1. Historie
  2. XML Dokumentenverarbeitung
  3. Aufbau XML-Dokument
  4. Weitere Regeln zur Syntax
  5. Wohlgeformtheit und Validität
  6. Document Type Definition, DTD
  7. Internationalisierung
  8. Namensräume
  9. XML Information Set

XML-Grundlagen [W3C status, reports, xml home]

Historie: zentrale XML-Spezifikationen

2006 Extensible Markup Language (XML) 1.1. Recommendation. [W3C REC, status]

2004 XML-Schema Part 0: Primer. Recommendation. [W3C REC, status]

2012 XML-Schema (XSD) 1.1 Part 1: Structures. [W3C REC]

2012 XML-Schema (XSD) 1.1 Part 2: Datatypes. [W3C REC]

2017 XSL Transformations (XSLT) 3.0. Prop. Recommendation. [W3C REC, status]

2017 XML Path Language (XPath) 3.1. Recommendation. [W3C REC, status]

2017 XML Query Language (XQuery) 3.1. Recommendation. [W3C REC, status]

2012 XSL Formatting Objects (XSL-FO) 2.0. Working Draft. [W3C WD, status]

XML-Grundlagen

Historie: weitere bekannte XML-Spezifikationen

2014 Mathematical Markup Language (MathML) 3.0. [W3C REC, status, examples]

2018 Scalable Vector Graphics (SVG) 2. [W3C CR, status, home]

2014 Resource Description Framework (RDF) 1.1. [W3C REC, status]

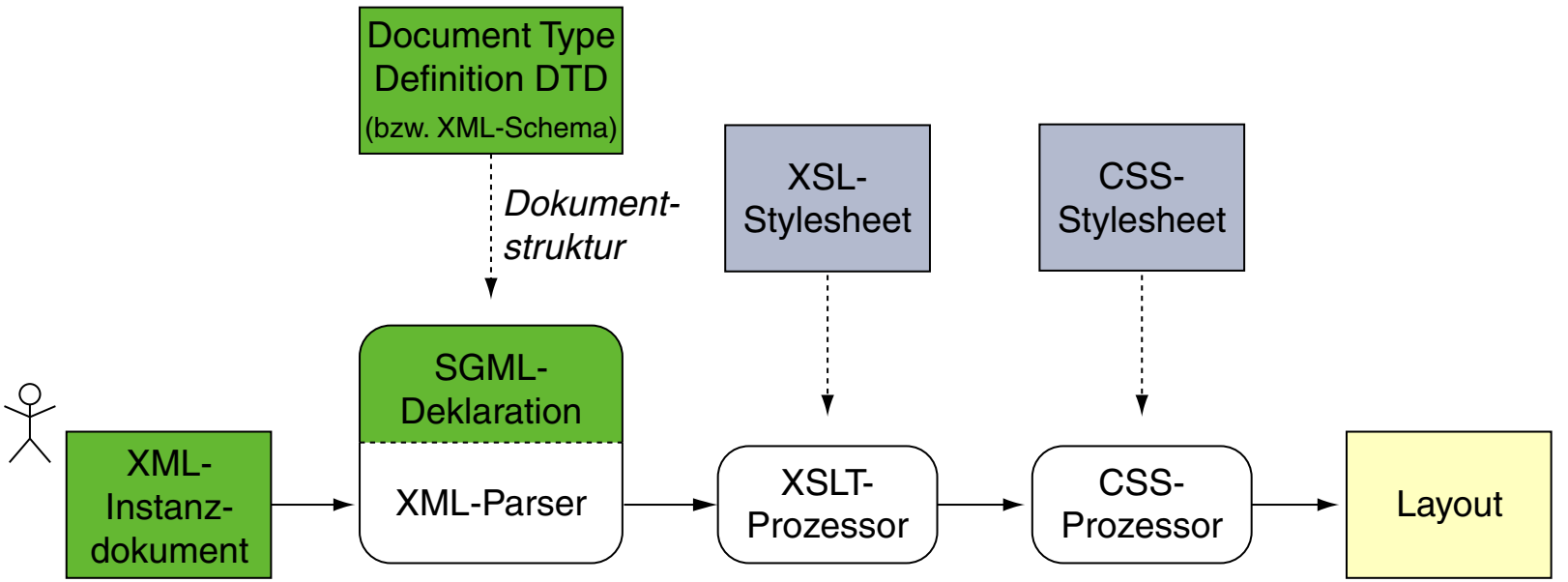
2007 Web Services Description Language (WSDL) 2.0. [W3C REC, status]

Bemerkungen:

- Viele Entwickler forderten eine erweiterbare Markup-Sprache, die weniger kompliziert als SGML ist. Tatsächlich ist die Spezifikation von XML deutlich einfacher und kürzer als die von SGML.
- XML-Dokumente werden nicht nur in Web-Anwendungen genutzt. Aufgrund seiner Flexibilität kann XML den Publishing-Anforderungen von Büchern, Zeitschriften, Katalogen, Postern etc. gerecht werden. Darüber hinaus hat XML als generelles Austauschformat große Verbreitung erlangt und stellt die wichtigste Sprache zur Kodierung von RDF-Graphen dar.
- Übersicht über die Ziele von XML : [W3C] (= sinnvolle Ziele für (Markup)Sprachen generell)
- Wiederholung: Der Standardisierungsprozess der W3C ist formalisiert und spiegelt sich in den verschiedenen Leveln der veröffentlichten Reports wider. [W3C reports]
- LaTeX ($\sim$ "Input-Format") versus MathML ($\sim$ "Output-Format") :
  - Input-Format: optimiert für die Eingabe / Formulierung durch Menschen
  - Output-Format: konzipiert für die Ausgabe auf unterschiedlichen Medien
  - Hintergrund: [Andrew Stacey] [stackexchange] [Wikipedia]

XML-Grundlagen

XML Dokumentenverarbeitung

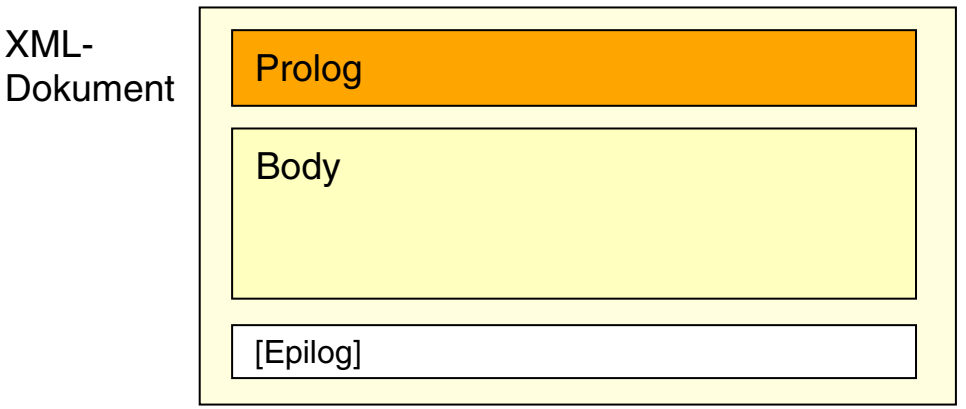


Vergleiche hierzu

- die SGML Dokumentenverarbeitung
- und die HTML Dokumentenverarbeitung.

XML-Grundlagen

Aufbau XML-Dokument




```
<?xml version="1.0"?>
<!DOCTYPE person ...>

<person>
  <name>
    <vorname>Alan</vorname>
    <nachname>...
  </name>
</person>
```

- Zum Prolog zählt alles vor dem Start des XML-Wurzelelementes; der Prolog enthält Meta-Informationen (= Informationen über das Dokument).
- Der Body besteht aus ineinander geschachtelten XML-Elementen.
- Der Epilog enthält Kommentare und Verarbeitungsanweisungen für das Dokument; der Epilog ist optional.
- Vergleiche hierzu die HTML-Dokumentstruktur.

XML-Grundlagen

Dokumenten-Prolog

1. XML-Deklaration.

```
<?xml version="1.0"?>
```

bzw.

```
<?xml version="1.0" encoding="UTF-8" standalone="no"?>
```

XML-Grundlagen

Dokumenten-Prolog

1. XML-Deklaration.

   ```
   <?xml version="1.0"?>
   ```
   bzw.
   ```
   <?xml version="1.0" encoding="UTF-8" standalone="no"?>
   ```

2a. DTD-Deklaration.

   ```
   <!DOCTYPE Wurzelelementname SYSTEM "URI">
   ```

XML-Grundlagen

Dokumenten-Prolog

1. XML-Deklaration.

   ```
   <?xml version="1.0"?>
   ```
   bzw.
   ```
   <?xml version="1.0" encoding="UTF-8" standalone="no"?>
   ```

2a. DTD-Deklaration.

   ```
   <!DOCTYPE Wurzelelementname SYSTEM "URI">
   ```

2b. Anstelle oder kombiniert mit der Referenz auf eine Datei lassen sich DTD-Befehle auch im Dokument einbinden. Man spricht von der externen bzw. der internen DTD-Teilmenge *(external subset, internal subset)*; zusammen bilden sie die vollständige DTD.

```
<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
<!DOCTYPE poem [
  <!ELEMENT poem (#PCDATA)>
  <!ENTITY author "William Shakespeare">
]>
<poem>
  Dieses Gedicht stammt von &author;
</poem>
```

XML-Grundlagen

Dokumenten-Prolog

1. XML-Deklaration.

   ```
   <?xml version="1.0"?>
   ```
   bzw.
   ```
   <?xml version="1.0" encoding="UTF-8" standalone="no"?>
   ```

2a. DTD-Deklaration.

   ```
   <!DOCTYPE Wurzelelementname SYSTEM "URI">
   ```

2b. Anstelle oder kombiniert mit der Referenz auf eine Datei lassen sich DTD-Befehle auch im Dokument einbinden. Man spricht von der externen bzw. der internen DTD-Teilmenge *(external subset, internal subset)*; zusammen bilden sie die vollständige DTD.

```
<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
<!DOCTYPE poem [
  <!ELEMENT poem (#PCDATA)>
  <!ENTITY author "William Shakespeare">
]>
<poem>
  Dieses Gedicht stammt von &author;
</poem>
```

3. Verarbeitungsanweisung für XML-Stylesheets.

   ```
   <?xml-stylesheet type="text/css" href="person.css"?>
   ```

Bemerkungen:

- `standalone="no"` bedeutet, dass der Rückgriff auf eine externe DTD erforderlich ist. `"no"` ist der Defaultwert für das `standalone`-Attribut.
- Die DTD-*Deklaration* enthält eine Referenz auf (= deklariert) den Typ des Dokuments mittels einer DTD *(Document Type Definition)*. Die DTD wiederum enthält die Definitionen und Deklarationen für Elemente und Attribute.
- Die Dateiendung einer DTD-Datei ist `.dtd`.
- Die Dokumenttyp-Deklaration kann entfallen.
- Bei einer weltweit bekannten DTD kann anstelle des Schlüsselwortes `SYSTEM` das Schlüsselwort `PUBLIC` zusammen mit dem Public-Identifier dieser DTD verwendet werden. Der Public-Identifier wiederum muss durch einen lokalen Katalog-Server auf eine URL abgebildet werden; als Fallback ist zusätzlich noch eine URI anzugeben. In der Praxis werden Public-Identifier kaum verwendet.

XML-Grundlagen

Dokumenten-Body

Allgemeine Form einer XML-Elementinstanz:

```
<elementname {attribute}*> ... </elementname>
```

Tags müssen balanciert sein – Ausnahme sind *Empty-Element-Tags*:

```
<elementName/>    ≡    <elementName></elementName>
```

XML-Grundlagen

Dokumenten-Body

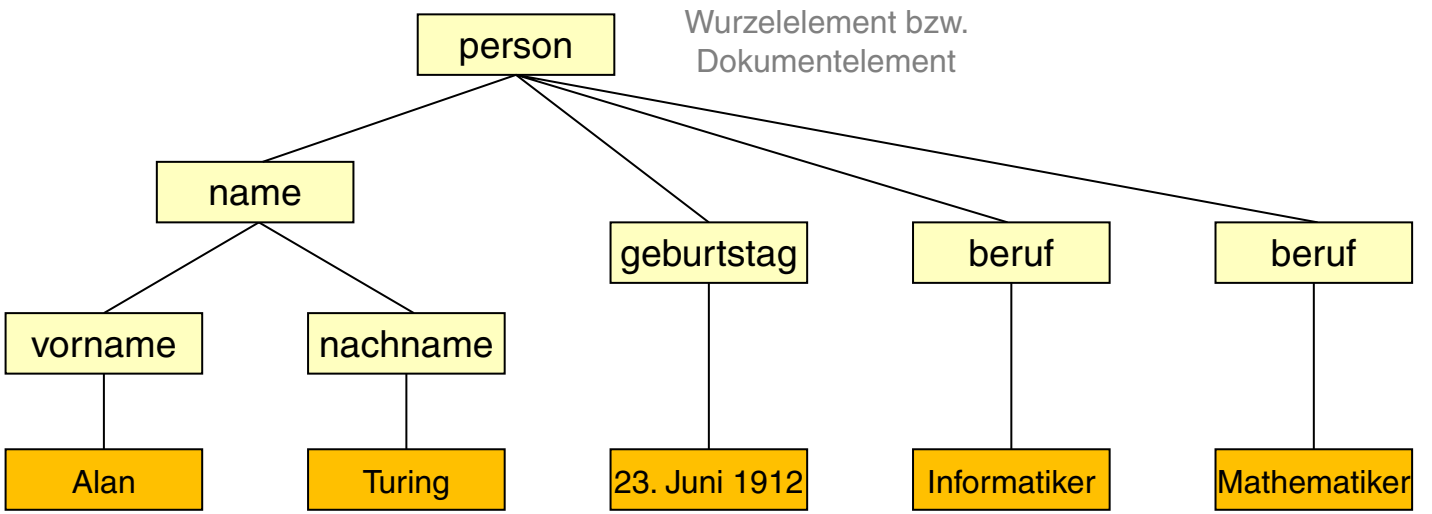
Allgemeine Form einer XML-Elementinstanz:

```
<elementname {attribute}*> ... </elementname>
```

Tags müssen balanciert sein – Ausnahme sind *Empty-Element-Tags*:

```
<elementName/>    ≡    <elementName></elementName>
```

Die Elementstruktur des Bodies entspricht einem Baum. Beispiel:



XML-Grundlagen

Attribute

Verwendung von Attributen wie in SGML [WT:III SGML] :

```
<person geboren="1912-06-23" gestorben="1954-06-07">
  Alan Turing
</person>
```

Frage des Stils: Elementmodellierung (a) oder Attributmodellierung (b)

(a)
```
<person>
  <name>
    <vorname>Alan</vorname>
    <nachname>Turing</nachname>
  </name>
  <beruf>Mathematiker</beruf>
  <beruf>Informatiker</beruf>
</person>
```

(b)
```
<person>
  <name vorname="Alan" nachname="Turing"/>
  <beruf wert="Mathematiker"/>
  <beruf wert="Informatiker"/>
</person>
```

Bemerkungen:

- Notiert man Empty-Element-Tags anstatt `<elementName ...>` als `<elementName ...></elementName>`, so darf kein Whitespace (Leerzeichen, Tabulatorzeichen, Zeilenvorschub) zwischen dem Start-Tag und dem Ende-Tag stehen.
- Zum Modellierungsstil: mit Hilfe von Attributen sollten nur Meta-Daten einer Elementinstanz spezifiziert werden. Deshalb ist im vorigen Beispiel die Elementmodellierung (a) vorzuziehen.
- Eine eindeutige Trennung zwischen Objektdaten und Meta-Daten ist nicht immer möglich.

XML-Grundlagen

Weitere Regeln zur Syntax

- XML-Namen dürfen aus beliebigen alphanumerischen, ideographischen sowie den drei Interpunktionszeichen „_“ , „–“ und „.“ bestehen.
- Entity-Referenzen wie in SGML: `&`*Entity-Name*`;` [WT:III SGML]
- Kommentare: `<!-- Dies ist ein Kommentar -->`

XML-Grundlagen

Weitere Regeln zur Syntax

- XML-Namen dürfen aus beliebigen alphanumerischen, ideographischen sowie den drei Interpunktionszeichen „_“ , „–“ und „.“ bestehen.
- Entity-Referenzen wie in SGML: `&`*Entity-Name*`;` [WT:III SGML]
- Kommentare: `<!-- Dies ist ein Kommentar -->`
- Die CDATA-Deklaration ermöglicht die literale Verwendung aller Zeichen:

```
<![CDATA[
  <svg xmlns="http://www.w3.org./2000/svg"
    width="12cm" height="10cm">
  <ellipse rx="110" ry="130" />
  ...
]]>
```

- Verarbeitungsanweisungen werden mit `<?` und `?>` eingeschlossen:

```
<?php
  mysql_connect("database.unc.edu", "clerk", "password");
  ...
?>
```

Bemerkungen:

- Ideographische Zeichen, auch Bildzeichen genannt, sind Zeichen, die eine unmittelbare Interpretation besitzen. Sie können sprachunabhängig als auch sprachspezifisch sein. Beispiele sind mathematische Zeichen, chinesische Schriftzeichen oder Logogramme.
- Kommentare und Verarbeitungsanweisungen sind Markup, aber keine Elementinstanzen. Sie dürfen überall im Dokument – jedoch nicht *in* einem Tag stehen.
- Das Schlüsselwort `#CDATA` bezeichnet den Datentyp *Character Data*. Abschnitte diesen Datentyps werden durch den Parser nicht analysiert. Innerhalb eines CDATA-Abschnitts wird nur die Zeichenkette „`]]>`“ als Markup interpretiert; sie markiert das CDATA-Ende-Tag. [w3schools]
- Das Schlüsselwort `#PCDATA` bezeichnet den Datentyp *Parsed Character Data*. Abschnitte diesen Datentyps werden durch den Parser analysiert. Innerhalb eines PCDATA-Abschnitts müssen deshalb Zeichen, die Bestandteil der HTML-Markup-Syntax sind (<, >, etc.) maskiert werden, falls sie nicht als Markup interpretiert werden sollen.
  Es sind alle Arten von Zeichen, Entity-Referenzen, CDATA-Abschnitte, Kommentare und Verarbeitungsanweisungen zugelassen, jedoch keine Elementinstanzen. Bei PCDATA handelt es sich üblicherweise um Text, der zwischen dem Anfang- und dem Ende-Tag einer Elementinstanz notiert wird. [w3schools]

XML-Grundlagen

Wohlgeformtheit und Validität

XML-Dokumente müssen wohlgeformt sein:

1. balancierte und geschachtelte (d.h. unverschränkte) Tags
2. genau ein Wurzelelement
3. Attributnamen eindeutig pro Element, Wertzuweisungen müssen in Anführungszeichen stehen
4. keine Kommentare und Verarbeitungsanweisungen in Tags

(es gibt noch mehr)

XML-Grundlagen

Wohlgeformtheit und Validität

XML-Dokumente müssen wohlgeformt sein:

1. balancierte und geschachtelte (d.h. unverschränkte) Tags
2. genau ein Wurzelelement
3. Attributnamen eindeutig pro Element, Wertzuweisungen müssen in Anführungszeichen stehen
4. keine Kommentare und Verarbeitungsanweisungen in Tags

(es gibt noch mehr)

XML-Dokumente können valide (gültig) sein. Das heißt,

1. das Dokument ist wohlgeformt,
2. das Dokument referenziert eine DTD (bzw. ein XML-Schema), und
3. der Dokumenteninhalt ist konform mit der DTD (bzw. dem XML-Schema).

[vgl. XML-Schema]

XML-Grundlagen

Document Type Definition, DTD

XML-Dokument:

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

<person>
  <name>
    <vorname>Alan</vorname>
    <nachname>Turing</nachname>
  </name>
  <geburtstag>23. Juni 1912</geburtstag>
  <beruf>Mathematiker</beruf>
  <beruf>Informatiker</beruf>
</person>
```

Zugehörige Document Type Definition [BNF]:

```
<!ELEMENT person (name, geburtstag, beruf+)>
<!ELEMENT name (vorname, nachname)>
<!ELEMENT vorname (#PCDATA)>
<!ELEMENT nachname (#PCDATA)>
<!ELEMENT geburtstag (#PCDATA)>
<!ELEMENT beruf (#PCDATA)>
```

XML-Grundlagen

Document Type Definition, DTD

XML-Dokument:

```
<?xml version="1.0"?>
<!DOCTYPE person SYSTEM "person.dtd">
<person>
  <name>
    <vorname>Alan</vorname>
    <nachname>Turing</nachname>
  </name>
  <geburtstag>23. Juni 1912</geburtstag>
  <beruf>Mathematiker</beruf>
  <beruf>Informatiker</beruf>
</person>
```

Zugehörige Document Type Definition [BNF]:

```
<!ELEMENT person (name, geburtstag, beruf+)>
<!ELEMENT name (vorname, nachname)>
<!ELEMENT vorname (#PCDATA)>
<!ELEMENT nachname (#PCDATA)>
<!ELEMENT geburtstag (#PCDATA)>
<!ELEMENT beruf (#PCDATA)>
```

XML-Grundlagen

Document Type Definition, DTD

XML-Dokument:

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

<person>
  <name>
    <vorname>Alan</vorname>
    <nachname>Turing</nachname>
  </name>
  <geburtstag>23. Juni 1912</geburtstag>
  <beruf>Mathematiker</beruf>
  <beruf>Informatiker</beruf>
</person>
```

Zugehörige Document Type Definition [BNF]:

```
<!ELEMENT person (name, geburtstag, beruf+)>
<!ELEMENT name (vorname, nachname)>
<!ELEMENT vorname (#PCDATA)>
<!ELEMENT nachname (#PCDATA)>
<!ELEMENT geburtstag (#PCDATA)>
<!ELEMENT beruf (#PCDATA)>
```

XML-Grundlagen

Document Type Definition, DTD

XML-Dokument:

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

<person>
  <name>
    <vorname>Alan</vorname>
    <nachname>Turing</nachname>
  </name>
  <geburtstag>23. Juni 1912</geburtstag>
  <beruf>Mathematiker</beruf>
  <beruf>Informatiker</beruf>
</person>
```

Zugehörige Document Type Definition [BNF]:

```
<!ELEMENT person (name, geburtstag, beruf+)>
<!ELEMENT name (vorname, nachname)>
<!ELEMENT vorname (#PCDATA)>
<!ELEMENT nachname (#PCDATA)>
<!ELEMENT geburtstag (#PCDATA)>
<!ELEMENT beruf (#PCDATA)>
```

XML-Grundlagen

Document Type Definition, DTD (Fortsetzung) [WT:III SGML]

Die DTD definiert:

1. Art und Aufbau von Elementtypen für eine Klasse von Dokumenten
   = die Inhaltsmodelle der Elementtypen
2. die in Elementinstanzen verwendbaren Attribute und ihre Datentypen
3. verschiedene Arten von Textkonstanten, sogenannte *Entities*

Closed-World-Semantik: Was in der DTD nicht deklariert ist, ist verboten.

XML-Grundlagen

Document Type Definition, DTD (Fortsetzung) [WT:III SGML]

Die DTD definiert:

1. Art und Aufbau von Elementtypen für eine Klasse von Dokumenten
   = die Inhaltsmodelle der Elementtypen
2. die in Elementinstanzen verwendbaren Attribute und ihre Datentypen
3. verschiedene Arten von Textkonstanten, sogenannte *Entities*

Closed-World-Semantik: Was in der DTD nicht deklariert ist, ist verboten.

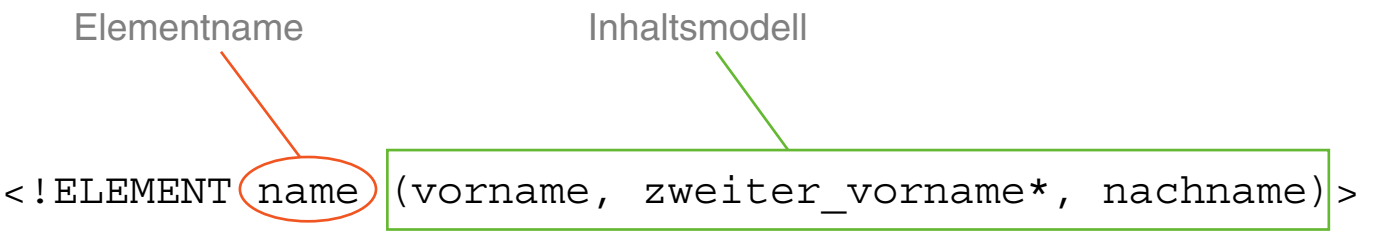
Syntax der Sätze in der DTD (Ausschnitt) [Beispiel]:

1. `<!ELEMENT` *Elementname Inhaltsmodell* `>`
2. `<!ATTLIST` *Elementname* `{` *Attributname Attributtyp Default-Wert* `}*` `>`
3. `<!ENTITY` *Entity-Name Zeichenkette* `>`

XML-Grundlagen

Document Type Definition, DTD (Fortsetzung)

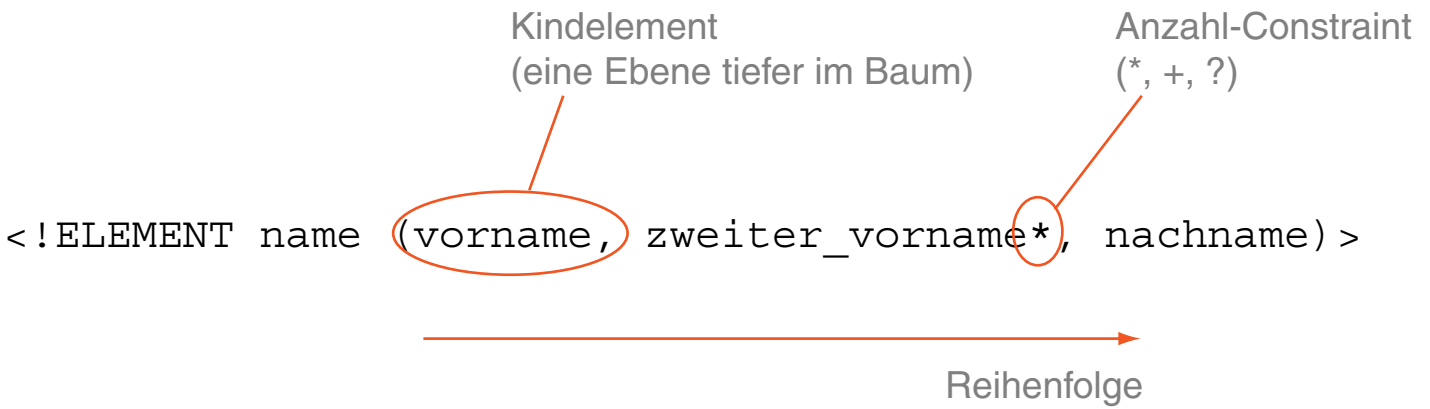
Definition eines Elementtyps in einer DTD:



XML-Grundlagen

Document Type Definition, DTD (Fortsetzung)

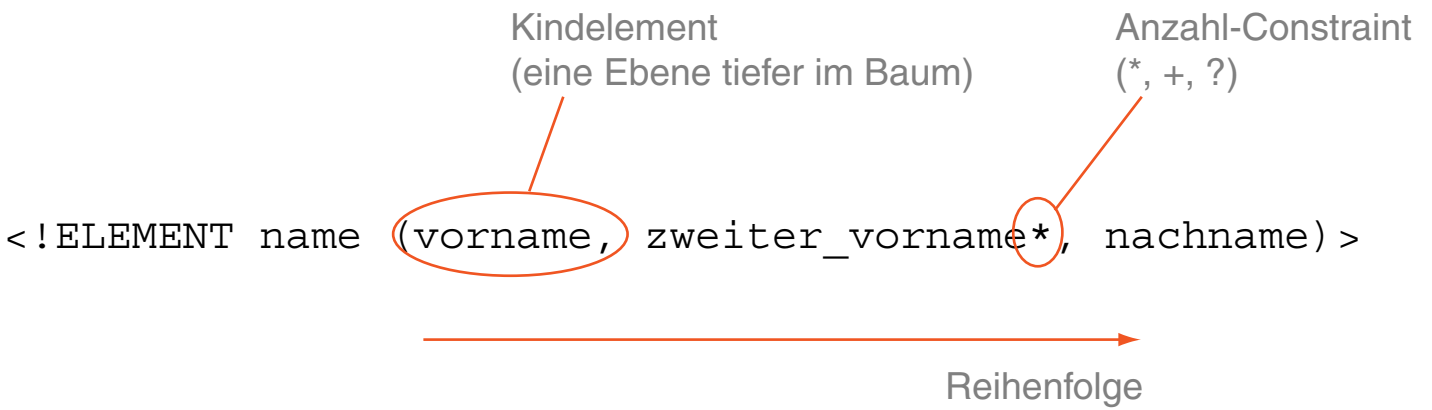
Definition eines Elementtyps in einer DTD:



XML-Grundlagen

Document Type Definition, DTD (Fortsetzung)

Definition eines Elementtyps in einer DTD:



gültige Elementinstanz:

```
<name>
  <vorname>Alan</vorname>
  <nachname>Turing</nachname>
</name>
```

ungültige Elementinstanzen:

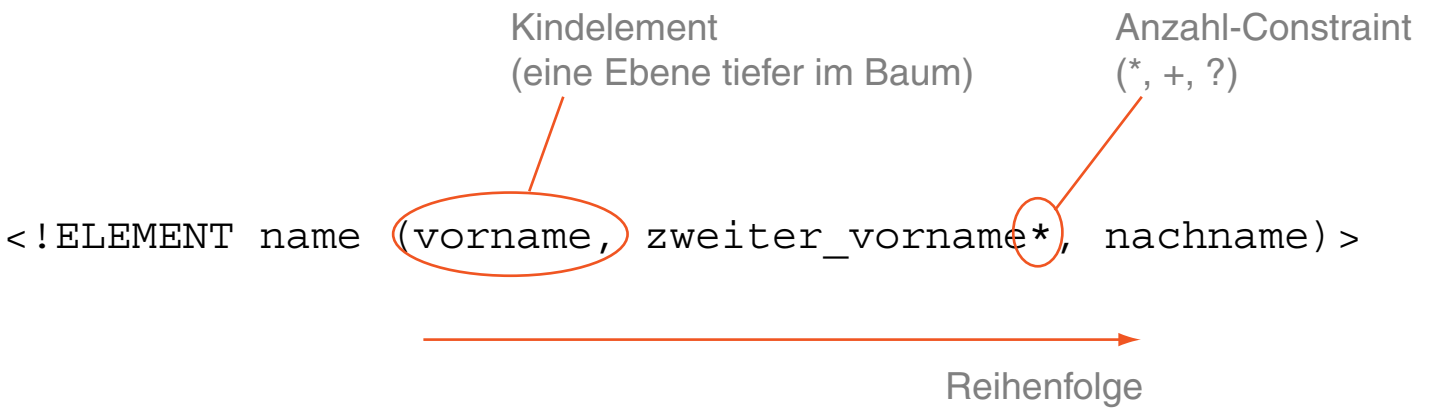
```
<name>
  <nachname>Turing</nachname>
  <vorname>Alan</vorname>
</name>
```

```
<name>
  <nachname>Turing</nachname>
</name>
```

XML-Grundlagen

Document Type Definition, DTD (Fortsetzung)

Definition eines Elementtyps in einer DTD:



gültige Elementinstanz:

```
<name>
  <vorname>Alan</vorname>
  <nachname>Turing</nachname>
</name>
```

ungültige Elementinstanzen:

```
<name>
  <nachname>Turing</nachname>
  <vorname>Alan</vorname>
</name>
```

```
<name>
  <nachname>Turing</nachname>
</name>
```

XML-Grundlagen

Document Type Definition, DTD (Fortsetzung)

Definition von Alternativen:

```
<!ELEMENT ziffer (null | eins | zwei | drei | vier | fünf)>
```

Klammerung und Kombination der syntaktischen Einheiten:

```
<!ELEMENT kreis (zentrum, (radius | durchmesser))>
```

XML-Grundlagen

Document Type Definition, DTD (Fortsetzung)

Definition von Alternativen:

```
<!ELEMENT ziffer (null | eins | zwei | drei | vier | fünf)>
```

Klammerung und Kombination der syntaktischen Einheiten:

```
<!ELEMENT kreis (zentrum, (radius | durchmesser))>
```

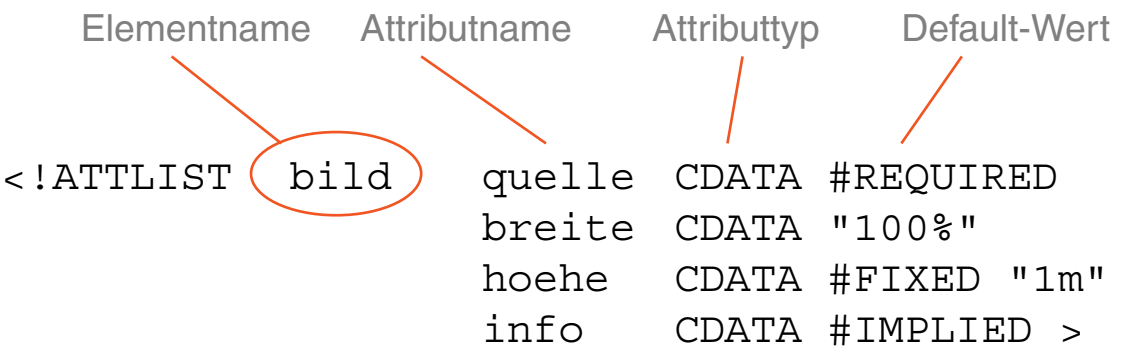
Wichtige Inhaltsmodelle für Elementtypen:

Inhaltsmodell	**Typischer Aufbau**
einfacher Inhalt	`<!ELEMENT` *Elementname* `(#PCDATA)>`
explizite Kindelemente	`<!ELEMENT` *Elementname* `(`*Elementname*`, . . . ,` *Elementname*`)>`
gemischter Inhalt	`<!ELEMENT` *Elementname* `(#PCDATA |` *Elementname*`)*>`
beliebiger Inhalt	`<!ELEMENT` *Elementname* `ANY>`
leerer Inhalt	`<!ELEMENT` *Elementname* `EMPTY>`

XML-Grundlagen

Document Type Definition, DTD (Fortsetzung)

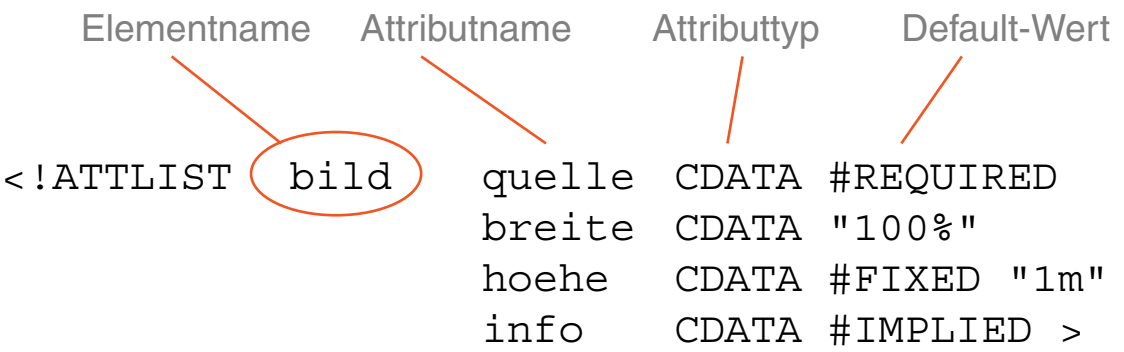
Deklaration der erlaubten Attribute für einen Elementtyp:



XML-Grundlagen

Document Type Definition, DTD (Fortsetzung)

Deklaration der erlaubten Attribute für einen Elementtyp:



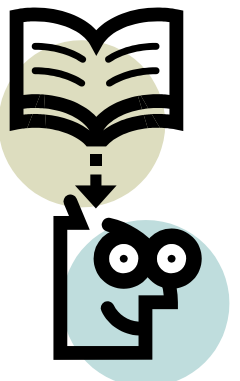
Default-Wert für Attribut	Semantik
`#REQUIRED`	das Attribut ist obligatorisch
Wert	Default, falls kein Attributwert im Dokument angegeben ist
`#FIXED` *Wert*	der Attributwert ist fest und kann angegeben sein
`#IMPLIED`	das Attribut ist optional

Bemerkungen:

- Attributwerte dürfen keine Elemente sein oder enthalten.
- In XML gibt es 10 Attributtypen, u.a.:
  - `CDATA` (Zeichenkette)
  - `ID` (eindeutiger Name)
  - `IDREF` (Verweis auf ein ID-Attribut)
  - `NMTOKEN` (Symbol)
  - `ENTITY`

XML-Grundlagen

Quiz [w3schools]



XML-Grundlagen

Internationalisierung

XML-Grundlagen

Internationalisierung (Fortsetzung)

Unterscheidung folgender Konzepte [W3C]:

1. abstraktes Zeichen *(Abstract character, Character)* [unicode]
   Element zur Konstruktion oder Repräsentation textueller Daten.
2. Zeichendarstellung *(Glyph, Glyph image, Character shape)* [unicode 1, 2]
3. Zeichenvorrat, Zeichensatz *(Character repertoire, Character set)* [unicode 1, 2]
   Menge von abstrakten Zeichen (1) zur Repräsentation von Text.

XML-Grundlagen

Internationalisierung (Fortsetzung)

Unterscheidung folgender Konzepte [W3C]:

1. abstraktes Zeichen *(Abstract character, Character)* [unicode]
   Element zur Konstruktion oder Repräsentation textueller Daten.

2. Zeichendarstellung *(Glyph, Glyph image, Character shape)* [unicode 1, 2]

3. Zeichenvorrat, Zeichensatz *(Character repertoire, Character set)* [unicode 1, 2]
   Menge von abstrakten Zeichen (1) zur Repräsentation von Text.

4. Code-Raum *(Codespace)* [unicode]
   Menge von Zahlen, die abstrakten Zeichen zugeordnet werden können. Zahlen des Code-Raums heißen Zeichen-Codes *(Code points, Character codes, Character numbers)*.

5. Code-Tabelle, codierter Zeichensatz *(Coded character set, Charset)* [unicode]
   Abbildung eines Zeichenvorrats bzw. Zeichensatzes (3) auf Code-Points (4).

6. Codierungsformat *(Encoding form, Encoding scheme, Encoding)* [unicode 1, 2]
   Format der Byte-Repräsentation eines Code-Points (4).

Bemerkungen:

- Fast alle Zeichen, die irgendwo auf der Welt benutzt werden, kann man bei www.unicode.org finden.
- Unterschied zwischen abstrakten Zeichen und Zeichendarstellung:
  “The difference between identifying a code point and rendering it on screen or paper is crucial to understanding. The character identified by a code point is an abstract entity, such as «LATIN CHARACTER CAPITAL A» or «BENGALI DIGIT 5». The mark made on screen or paper – called a glyph – is a visual representation of the character.
  The Unicode Standard does not define glyph images. The standard defines how characters are interpreted, not how glyphs are rendered. The software or hardware-rendering engine of a computer is responsible for the appearance of the characters on the screen. The Unicode Standard does not specify the size, shape, nor style of on-screen characters.” [unicode]

  “In Unicode, the letter A is a platonic ideal. It’s just floating in heaven.” [www.joelonsoftware.com]
- Das Konzept des Zeichenvorrats (3) ist an das Konzept einer Sprache geknüpft und folglich unabhängig von einem Computer. Ein Beispiel für einen Zeichenvorrat (3) ist Latein 1 *(Latin 1)*: es ist die Menge der Zeichen, die in ISO/IEC 8859-1 aufgeführt sind und die zum Schreiben der Sprachen Westeuropas benötigt werden.

XML-Grundlagen

Internationalisierung (Fortsetzung)

Ein Zeichenvorrat (3) für eine bestimmte Sprache heißt Alphabet oder Schrift.

Beispiele: lateinische Schrift, chinesische Schrift.

Anspruch von Unicode:

> *"The Unicode Standard provides the capacity to encode all of the characters used for the written languages of the world. To keep character coding simple and efficient, the Unicode Standard assigns each character a unique numeric value and name."* [unicode]

Sprache	(4) Code-Raum	(5) **Code-Tabelle** bzw. **Charset**	(6) **Codierungsformat** bzw. **Encoding**	Code-Einheit	Code-Länge
Englisch	0-7F	US-ASCII	*kanonisch*	1 Byte	1 Byte
Westeuropa	0-FF	ISO 8859-1	*kanonisch*	1 Byte	1 Byte
Chinesisch	0-FFFF	Big 5	*kanonisch*	2 Byte	2 Byte
Weltweit	0-10FFFF	Unicode 12.0	UTF-8 UTF-16	1 Byte 2 Byte	1-4 Byte 2-4 Byte

XML-Grundlagen

Internationalisierung (Fortsetzung)

Ein Zeichenvorrat (3) für eine bestimmte Sprache heißt Alphabet oder Schrift.

Beispiele: lateinische Schrift, chinesische Schrift.

Anspruch von Unicode:

> *"The Unicode Standard provides the capacity to encode all of the characters used for the written languages of the world. To keep character coding simple and efficient, the Unicode Standard assigns each character a unique numeric value and name."* [unicode]

Sprache	(4) Code-Raum	(5) **Code-Tabelle** bzw. **Charset**	(6) **Codierungsformat** bzw. **Encoding**	Code-Einheit	Code-Länge
Englisch	0-7F	US-ASCII	*kanonisch*	1 Byte	1 Byte
Westeuropa	0-FF	ISO 8859-1	*kanonisch*	1 Byte	1 Byte
Chinesisch	0-FFFF	Big 5	*kanonisch*	2 Byte	2 Byte
Weltweit	0-10FFFF	Unicode 12.0	UTF-8 UTF-16	1 Byte 2 Byte	1-4 Byte 2-4 Byte

Bemerkungen:

- Viele Code-Tabellen liegen in nur *einem* Codierungsformat / Encoding (6) vor, das in der Tabelle hier als kanonisch bezeichnet ist. In der Praxis wird oft (aber fälschlicherweise) der Name der Code-Tabelle / Charset (5) für das Codierungsformat / Encoding (6) verwendet.
- Beachte:
  - In der Meta-Information von HTML-Dokumenten verwendet man das Attribut `charset`, um das Codierungsformat / Encoding (6) (`UTF-8`, `UTF-16`, etc.) zu deklarieren – und nicht etwa die Code-Tabelle / Charset (5) (`Unicode`, etc.) [W3C]
  - In der Meta-Information von XML-Dokumenten verwendet man (richtigerweise) das Attribut `encoding`, um das Codierungsformat / Encoding zu deklarieren. [W3C]
- UTF-8 und UTF-16 verwenden variable Code-Längen. UTF steht für Unicode Transformation Format. [unicode]
- Damit ein XML-Parser ein Dokument lesen kann, muss er die verwendete Code-Tabelle / Charset (5) und dessen Codierungsformat / Encoding (6) kennen. [W3C]

XML-Grundlagen

Internationalisierung (Fortsetzung)

Die Code-Tabellen US-ASCII und GER-ASCII: druckbare Zeichen erhalten die Zeichen-Codes von 32 bis 126, Steuerzeichen von 0 bis 31.

US-ASCII.

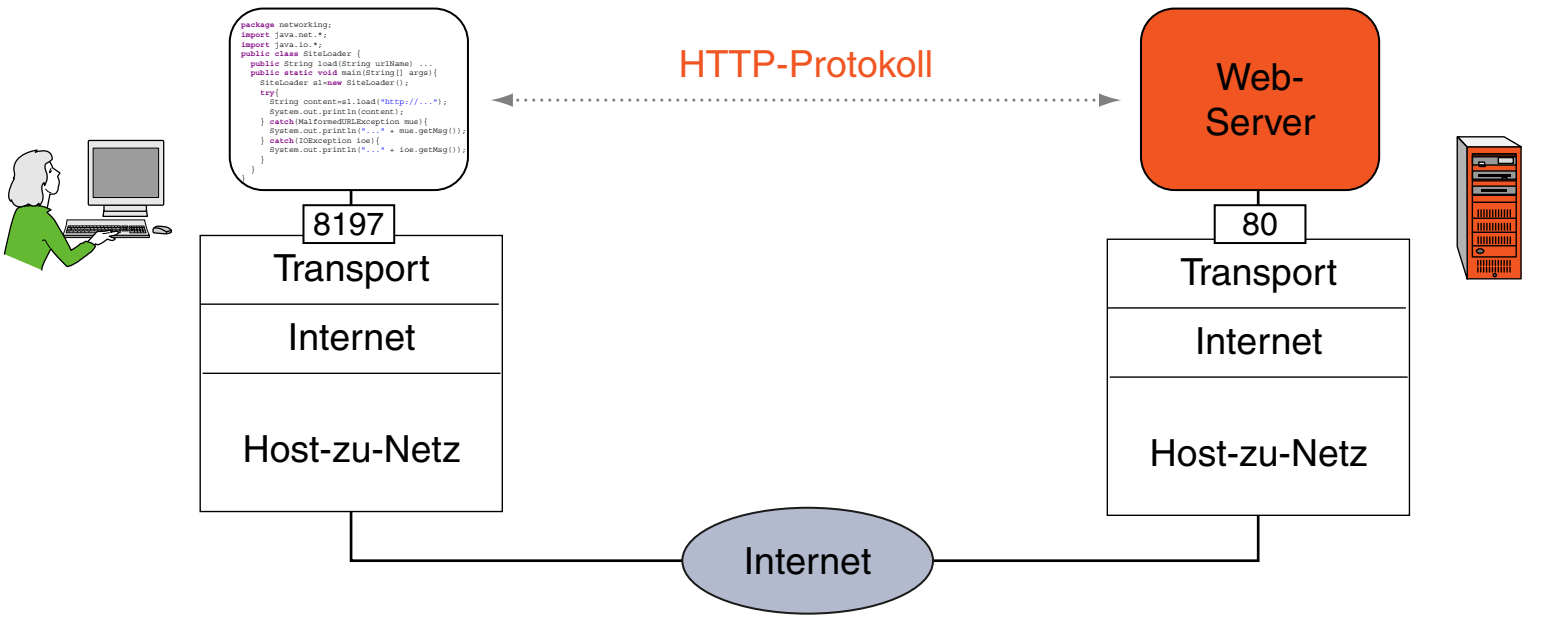
20	21 !	22 "	23 #	24 $	25 %	26 &	27 '	28 (	29)	2A *	2B +	2C ,	2D -	2E .	2F /
30 0	31 1	32 2	33 3	34 4	35 5	36 6	37 7	38 8	39 9	3A :	3B ;	3C <	3D =	3E >	3F ?
40 @	41 A	42 B	43 C	44 D	45 E	46 F	47 G	48 H	49 I	4A J	4B K	4C L	4D M	4E N	4F O
50 P	51 Q	52 R	53 S	54 T	55 U	56 V	57 W	58 X	59 Y	5A Z	5B [	5C \	5D]	5E ^	5F _
60 `	61 a	62 b	63 c	64 d	65 e	66 f	67 g	68 h	69 i	6A j	6B k	6C l	6D m	6E n	6F o
70 p	71 q	72 r	73 s	74 t	75 u	76 v	77 w	78 x	79 y	7A z	7B {	7C |	7D }	7E ~	

GER-ASCII.

20	21 !	22 "	23 #	24 $	25 %	26 &	27 '	28 (	29)	2A *	2B +	2C ,	2D -	2E .	2F /
30 0	31 1	32 2	33 3	34 4	35 5	36 6	37 7	38 8	39 9	3A :	3B ;	3C <	3D =	3E >	3F ?
40 §	41 A	42 B	43 C	44 D	45 E	46 F	47 G	48 H	49 I	4A J	4B K	4C L	4D M	4E N	4F O
50 P	51 Q	52 R	53 S	54 T	55 U	56 V	57 W	58 X	59 Y	5A Z	5B Ä	5C Ö	5D Ü	5E ^	5F _
60 `	61 a	62 b	63 c	64 d	65 e	66 f	67 g	68 h	69 i	6A j	6B k	6C l	6D m	6E n	6F o
70 p	71 q	72 r	73 s	74 t	75 u	76 v	77 w	78 x	79 y	7A z	7B ä	7C ö	7D ü	7E ß	

XML-Grundlagen

Internationalisierung: HTTP-Kommunikation mit Java [WT:II SiteLoader]



XML-Grundlagen

Internationalisierung: HTTP-Kommunikation mit Java [WT:II SiteLoader]

```
public static String load(String urlString) throws IOException {
    URL url = new URL(urlString);
    HttpURLConnection con = (HttpURLConnection) url.openConnection();

    String contentType = con.getContentType();
    System.out.println("Content-Type: " + contentType);
    String encoding = extractCharset(contentType, "utf-8");
    System.out.println("Charset encoding: " + encoding);

    InputStream in = con.getInputStream();
    BufferedReader br =new BufferedReader(new InputStreamReader(in, encoding));
    String curline;
    StringBuilder content = new StringBuilder();

    while ((curline = br.readLine()) != null) {
        content.append(curline + '\n');
    }

    br.close();
    con.disconnect();
    return content.toString();
 }
```

XML-Grundlagen

Internationalisierung: HTTP-Kommunikation mit Java [WT:II SiteLoader]

```
public static String load(String urlString) throws IOException {
    URL url = new URL(urlString);
    HttpURLConnection con = (HttpURLConnection) url.openConnection();

    String contentType = con.getContentType();
    System.out.println("Content-Type: " + contentType);
    String encoding = extractCharset(contentType, "utf-8");
    System.out.println("Charset encoding: " + encoding);

    InputStream in = con.getInputStream();
    BufferedReader br =new BufferedReader(new InputStreamReader(in, encoding));
    String curline;
    StringBuilder content = new StringBuilder();

    while ((curline = br.readLine()) != null) {
        content.append(curline + '\n');
    }

    br.close();
    con.disconnect();
    return content.toString();
}
```

XML-Grundlagen

Internationalisierung: HTTP-Kommunikation mit Java [WT:II SiteLoader]

```
public static String extractCharset
    (String contentType, String defaultCharset) {
    // Extracts the charset value from the Content-Type header.
    // If no charset value is specified, the given default is returned.
    // Example. Content-Type: "text/html; charset=UTF-8"
    // Background: The *charset* value corresponds to the *encoding* if
    // for the charset only a single (canonical) encoding exists.

    String charset = defaultCharset;
    for (String param : contentType.replace(" ","").split(";")) {
        if (param.toLowerCase().startsWith("charset=")) {
            charset = param.split("=")[1];
            break;
        }
    }
    return charset;
}
```

XML-Grundlagen

Internationalisierung: HTTP-Kommunikation mit Java [WT:II SiteLoader]

```
package documentlanguages.webcrawler;
import java.net.*;
import java.io.*;

public class SiteLoader2 {

    public static String load(String urlString) ...
    public static String extractCharset(String contentType, ...

    public static void main(String[] args){
        try{
            String content = SiteLoader2.load("http://www.heise.de");
            System.out.println(content);
        }
        catch(MalformedURLException e) {
            System.out.println("MalformedURLException:" + e.getMessage());
        }
        catch(IOException e) {
            System.out.println("IOException:" + e.getMessage());
        }
    }
}
```

XML-Grundlagen

Internationalisierung: HTTP-Kommunikation mit Java [WT:II Response-Message]

```
[stein@webis bin]$ java documentlanguages.webcrawler.SiteLoader2 | less
Content-Type: text/html; charset=utf-8
Charset encoding: utf-8

<!DOCTYPE html>

<html lang="de">
<head>
<title>heise online - IT-News, Nachrichten und Hintergründe</title>
<meta name="description" content="News und Foren zu Computer, IT, Wissenschaft, ...
<meta name="keywords" content="heise online, c't, iX, Technology Review, ...
<meta name="publisher" content="Heise Zeitschriften Verlag" />
<meta name="viewport" content="width=1175" />
<link rel="home" type="text/html" title="Startseite" href="/" />
<link rel="copyright" title="Copyright" href="/impressum.html" />
<meta http-equiv="PICS-Label" content="(PICS-1.1 &quot;http://www.rsac.org/...
<meta name="generator" content="InterRed V14.0, http://www.interred.de/, ...
<script type="text/javascript" src="/js/jquery/jquery-1.7.1.min.js"></script>
<script type="text/javascript" src="/js/ho/link_inline_images.min.js"></script>
<script type="text/javascript" src="/js/ho/bilderstrecke-1.1.min.js"></script>
...
</head>

<body>
...
```

Bemerkungen:

- Neben dem Entity-Header „Content-Type“ gibt es auch den Entity-Header „Content-Encoding“, der allerdings dafür verwendet wird etwaige Kompremierungen des Entity-Body anzugeben (z.B.: „zip“).
- Beachte, dass in dem Beispiel nicht das Encoding-Attribut, sondern das Charset-Attribut abgefragt wird. Das macht dann keinen Unterschied, wenn das Encoding kanonisch, also eindeutig ist, wie z.B. für den Charset `ISO 8859-1`.

XML-Grundlagen

Namensräume

Namensräume *(Namespaces)* in XML haben folgende Aufgaben:

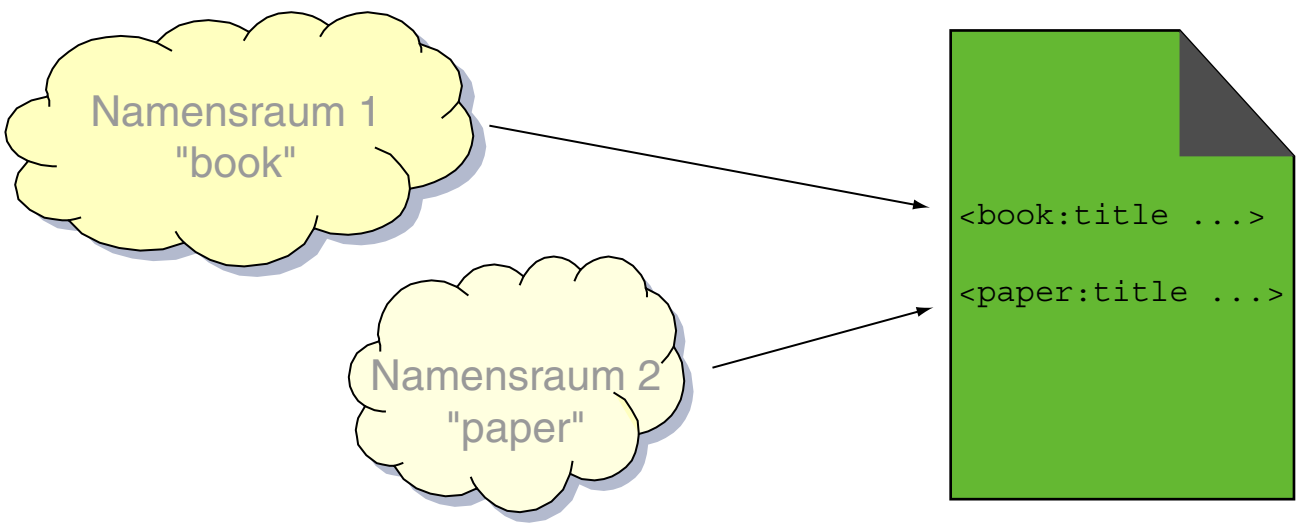
- Unterscheidung von Elementen und Attributen, die den gleichen Namen haben, aber in verschiedenen Zusammenhängen verwendet werden.
  Stichwort: Vermeidung von Namenskonflikten
- Logische Gruppierung aller Elemente und Attribute einer Anwendung.

XML-Grundlagen

Namensräume

Namensräume *(Namespaces)* in XML haben folgende Aufgaben:

- Unterscheidung von Elementen und Attributen, die den gleichen Namen haben, aber in verschiedenen Zusammenhängen verwendet werden.
  Stichwort: Vermeidung von Namenskonflikten
- Logische Gruppierung aller Elemente und Attribute einer Anwendung.



Namensräume sind Bezeichner – sie definieren keine Umgebung *(Scope)*.

XML-Grundlagen

Namensräume (Fortsetzung)

Namensräume *(Namespaces)* in XML haben folgende Aufgaben:

- Unterscheidung von Elementen und Attributen, die den gleichen Namen haben, aber in verschiedenen Zusammenhängen verwendet werden.
  Stichwort: Vermeidung von Namenskonflikten
- Logische Gruppierung aller Elemente und Attribute einer Anwendung.

Verwendung von Namensräumen in zwei Schritten:

1. Deklaration des Namensraums durch Bindung einer URI an einen Präfix mit `xmlns`.

   ```
   xmlns:book="https://www.books.com"
   ```

2. Qualifizierung des Vokabulars.

book:`title`	qualifizierender Name
book:	Präfix
`title`	lokaler Teil

Jede URI ist als Namensraum verwendbar.

XML-Grundlagen

Namensräume (Fortsetzung)

Gültigkeit der Namensraumdeklaration:

- Innerhalb des Elements (einschließlich), in dem die URI gebunden wird.

```
<book:Buch xmlns:book="https://www.books.com">
  <book:Titel>Heuristics</book:Titel>
  <Autor>Judea Pearl</Autor>
</book:Buch>
```

- Überschreiben der Deklaration innerhalb der Elementhierarchie möglich.

XML-Grundlagen

Namensräume (Fortsetzung)

Gültigkeit der Namensraumdeklaration:

- Innerhalb des Elements (einschließlich), in dem die URI gebunden wird.

```
<book:Buch xmlns:book="https://www.books.com">
  <book:Titel>Heuristics</book:Titel>
  <Autor>Judea Pearl</Autor>
</book:Buch>
```

- Überschreiben der Deklaration innerhalb der Elementhierarchie möglich.

Konzept des Default-Namensraums ermöglicht Qualifizierung ohne Präfix:

- Deklaration durch Bindung einer URI an den leeren Präfix.

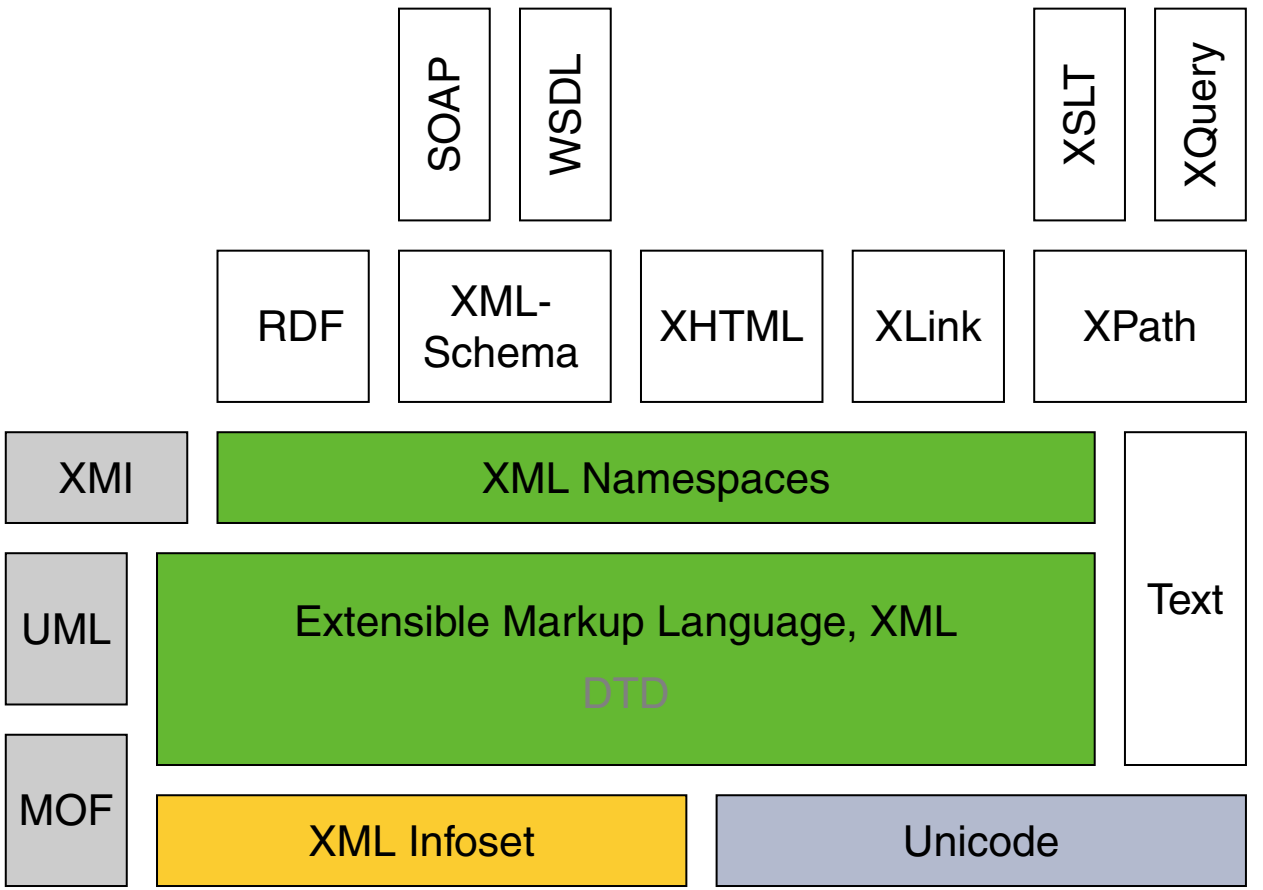
```
<Buch xmlns="https://www.books.com">
  <Titel>Heuristics</Titel>
  <Autor>Judea Pearl</Autor>
</Buch>
```

Bemerkungen:

- Man bezeichnet Elementnamen, Attributnamen etc. die zu einem Namensraum gehören, als „qualifiziert“. Diese Qualifizierung kann explizit über ein Präfix, oder implizit über die Deklaration eines Default-Namensraums geschehen.
- Nicht qualifizierte Namen gehören zu dem anonymen bzw. universellen Namensraum. Im ersten Beispiel gehört `<Autor>` zum anonymen Namensraum.
- Die Konzepte Default-Namensraum (= implizite Qualifizierung ohne Präfix) und anonymer Namensraum (= keine Qualifizierung) sind sorgfältig zu unterscheiden.
- *Attribute* ohne Präfix gehören nicht zum Default-Namensraum. D.h., auch wenn ein Element zu einem bestimmten (Default-)Namensraum gehört, so gehören seine *Attribute* ohne Präfix zu dem anonymen Namensraum.
- Eine Namensraumdeklaration mit Präfix besitzt Präferenz gegenüber dem Default-Namensraum.
- Durch Bindung einer leeren URI an einen Präfix wird eine bestehende Namensraumdeklaration aufgehoben. So entsteht eine Situation identisch zu einem Dokument ohne Namensraum; d.h., die Elemente gehören zum anonymen Namensraum.
- Eine Elementinstanz kann mehrere Namensraumdeklarationen aufnehmen. In der Praxis hat es sich aus Übersichtlichkeitsgründen durchgesetzt, alle in einem XML-Dokument verwendeten Namensräume zu Beginn des Dokuments im Wurzelelement zu deklarieren.

XML-Grundlagen

XML Information Set [W3C] [Jeckle 2004]



XML-Grundlagen

XML Information Set (Fortsetzung)

Das XML Information Set (Infoset) definiert das XML-Dokumenten unterliegende Datenmodell. Es dient zur Beantwortung der Frage:

> Welche Informationen sind in einem XML-Dokument codiert?

Ein Parser, der ein XML-Dokument analysiert, orientiert sich an der Definition des XML Information Set und stellt die entsprechenden aus dem XML-Dokument ableitbaren Informationen in einer Datenstruktur bereit.

XML-Grundlagen

XML Information Set (Fortsetzung)

Das XML Information Set (Infoset) definiert das XML-Dokumenten unterliegende Datenmodell. Es dient zur Beantwortung der Frage:

> Welche Informationen sind in einem XML-Dokument codiert?

Ein Parser, der ein XML-Dokument analysiert, orientiert sich an der Definition des XML Information Set und stellt die entsprechenden aus dem XML-Dokument ableitbaren Informationen in einer Datenstruktur bereit. Beispiele:

- wie ein Element heißt
- zu welchem Namensraum ein Element gehört
- Reihenfolge der Elementinstanzen
- Code-Tabelle

Beispiele für nicht ableitbare Information:

- Größe des Leerraums zwischen Attributen
- Reihenfolge der Attribute eines Elementtyps

Bemerkungen:

- Das XML Information Set ist keine Sprache wie andere W3C-Spezifikationen, sondern ein Datenmodell. Die XML-Syntax ist eine Serialisierung dieses Datenmodells.
- Das W3C hat mittlerweile drei Datenmodelle für XML-Dokumente definiert: XML Information Set, XPath, Document Object Model (DOM). Das XPath-Datenmodell basiert auf einer Baumstruktur, die bei der Abfrage eines XML-Dokuments durchlaufen wird und ist dem XML Information Set ähnlich; DOM ist der Vorläufer beider Datenmodelle. DOM und das XPath-Datenmodell können als Interpretationen des XML Information Sets betrachtet werden. [MSDN]

XML-Grundlagen

XML Information Set (Fortsetzung)

Das XML Information Set eines XML-Dokuments wird als Baum repräsentiert.

Die Elemente des Baums heißen Informationseinheiten *(Information Items)* und sind von einem der folgenden Typen [W3C]:

1. Document Information Item $\equiv$ Wurzelknoten des Dokuments
2. Element Information Item
3. Attribute Information Item
4. Processing Instruction Information Item
5. Unexpanded Entity Reference Information Item
6. Character Information Item
7. Comment Information Item
8. Document Type Declaration Information Item
9. Unparsed Entity Information Item
10. Notation Information Item
11. Namespace Information Item

XML-Grundlagen

XML Information Set: Beispiel

```
<?xml version="1.0"
 encoding="ISO-8859-1"
 standalone="yes">

<person>
  <name geburtstag="23-06-1912">
    <vorname>Alan</vorname>
    <nachname>Turing</nachname>
  </name>
</person>
```

XML-Grundlagen

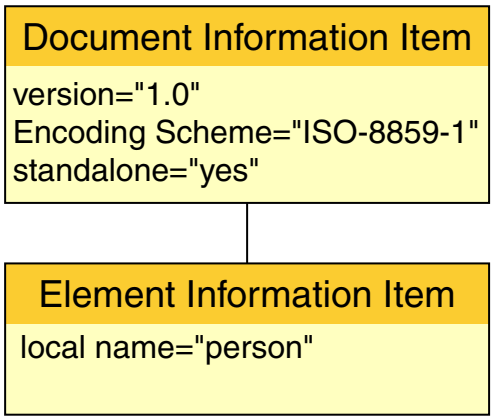
XML Information Set: Beispiel (Fortsetzung)

Document Information Item
version="1.0" Encoding Scheme="ISO-8859-1" standalone="yes"

```
<?xml version="1.0"
 encoding="ISO-8859-1"
 standalone="yes">

<person>
  <name geburtstag="23-06-1912">
    <vorname>Alan</vorname>
    <nachname>Turing</nachname>
  </name>
</person>
```

XML-Grundlagen

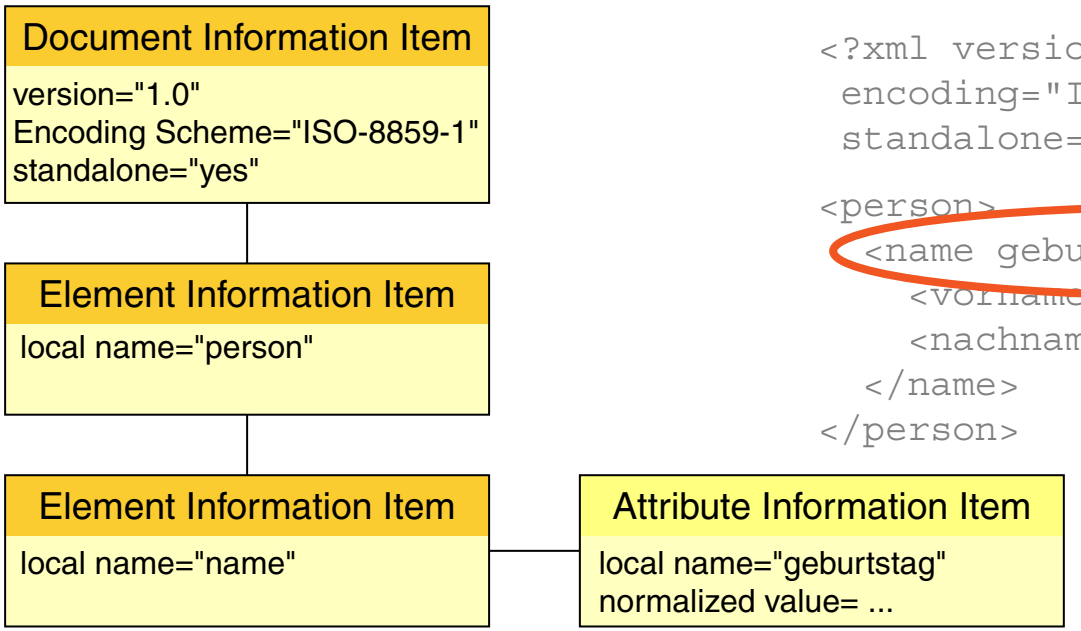
XML Information Set: Beispiel (Fortsetzung)




```
<?xml version="1.0"
 encoding="ISO-8859-1"
 standalone="yes">

<person>
  <name geburtstag="23-06-1912">
    <vorname>Alan</vorname>
    <nachname>Turing</nachname>
  </name>
</person>
```

XML-Grundlagen

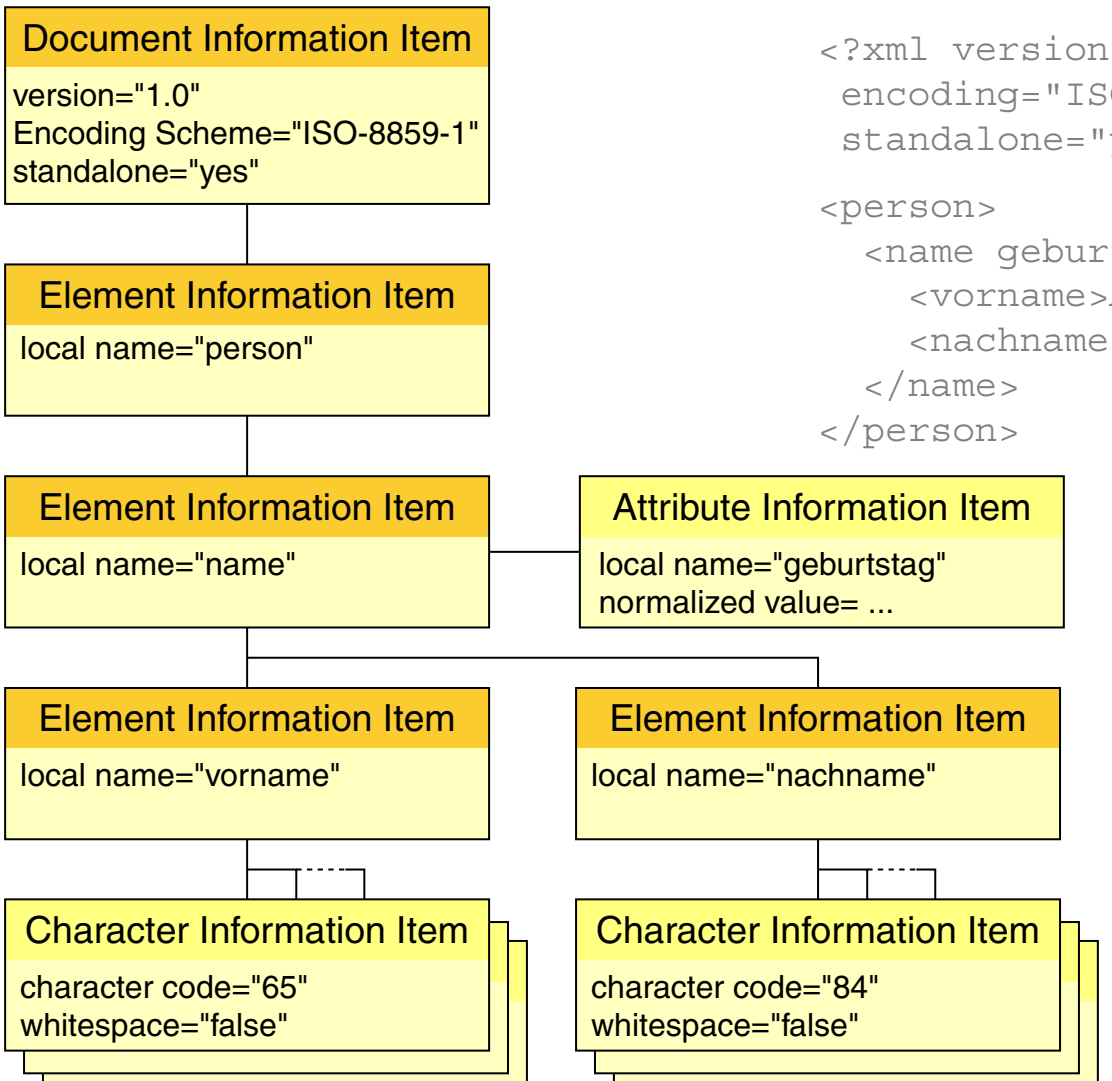
XML Information Set: Beispiel (Fortsetzung)




```
<?xml version="1.0"
 encoding="ISO-8859-1"
 standalone="yes">

<person>
  <name geburtstag="23-06-1912">
    <vorname>Alan</vorname>
    <nachname>Turing</nachname>
  </name>
</person>
```

XML-Grundlagen

XML Information Set: Beispiel (Fortsetzung)




```
<?xml version="1.0"
 encoding="ISO-8859-1"
 standalone="yes">

<person>
  <name geburtstag="23-06-1912">
    <vorname>Alan</vorname>
    <nachname>Turing</nachname>
  </name>
</person>
```

XML-Grundlagen

Quellen zum Nachlernen und Nachschlagen im Web: XML

- Jeckle. *XML Vorlesung.*
  www.mario-jeckle.de
- W3C. *Namespaces in XML 1.1.*
  www.w3.org/TR/xml-names11
- W3C. *XML Information Set, Second Edition.*
  www.w3.org/TR/xml-infoset
- W3 Schools. *XML Tutorial.*
  www.w3schools.com/xml

XML-Grundlagen

Quellen zum Nachlernen und Nachschlagen im Web: Internationalisierung

- Joel on Software. *Unicode Essay.*
  www.joelonsoftware.com/articles/Unicode.htm

- Unicode.*Glossary.*
  www.unicode.org/glossary

- W3C. Internationalization.

  Character Encodings for Beginners.
  www.w3.org/International/questions/qa-what-is-encoding

  Character encodings: Essential concepts.
  www.w3.org/International/articles/definitions-character

- W3C. *Handling character encodings in HTML and CSS (tutorial).*
  www.w3.org/International/tutorials/tutorial-char-enc