

# 9. Web-Dokumente

- 9.1 Generische Auszeichnungssprachen: XML
- 9.2 XML und Style Sheets
- 9.3 XML für Multimedia: SMIL
- 9.4 XML für Web-Informationendienste: RSS



## Medieninformatik-Buch: Kapitel 10



Weiterführende Literatur:

H. Vonhoegen: Einstieg in XML, Galileo Computing,  
7. Auflage 2013

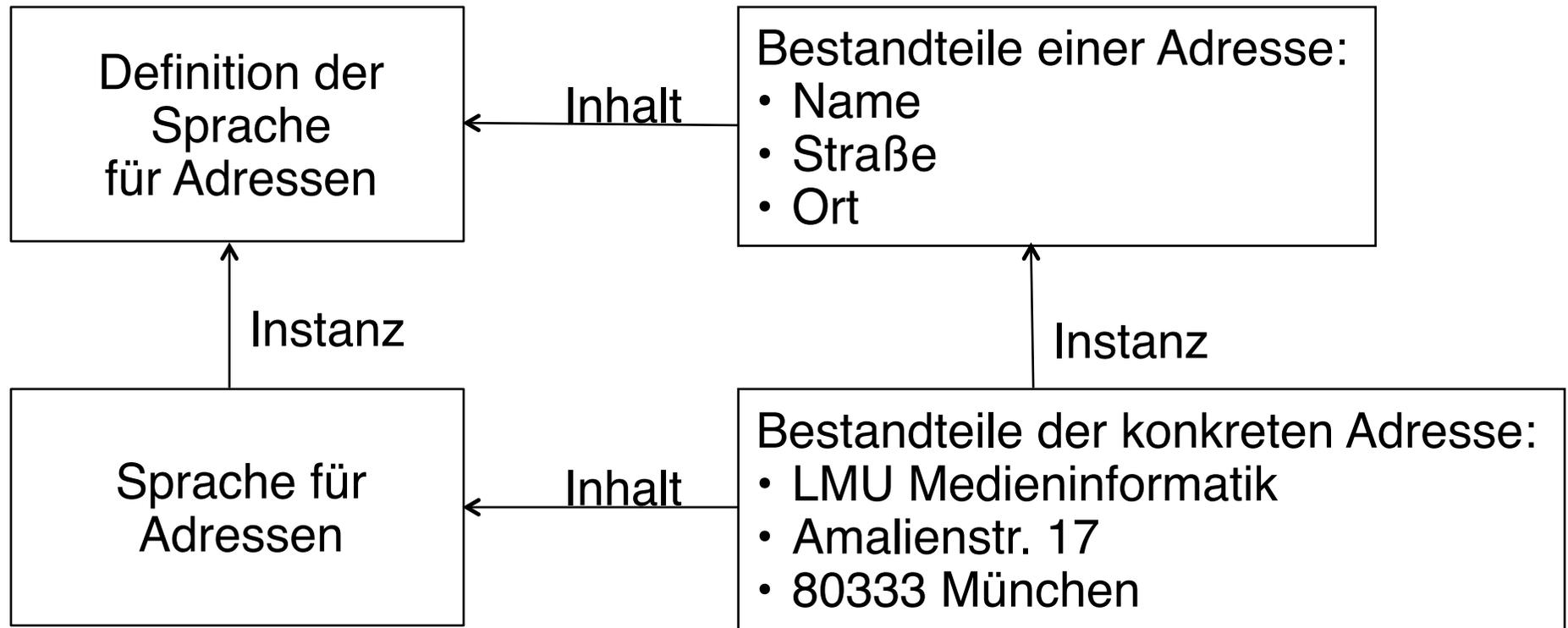
<http://www.w3schools.com/xml/>

# Generische Auszeichnungssprache: Idee

- Auszeichnungssprache (*markup language*):
  - Text und eingebettete textuelle Zusatzinformation (insbesondere zur Darstellung)
  - Tag-/Attribut-Syntax weithin bekannt durch HTML
- Idee: „Familie“ von Auszeichnungssprachen gleicher Basissyntax für verschiedenste Anwendungsgebiete
  - Web-Seiten-Formatierung (HTML)
  - Strukturierte Daten, z.B. Adressen, Briefe, Texttypen
  - Austauschformate für Textverarbeitung und andere Software
  - Standardformate für Grafik, Multimedia-Präsentationen, ...
- Vorteile:
  - ***Trennung von Inhalt, Struktur und Präsentation***
  - Lesbarkeit durch Mensch und Maschine
  - Automatische syntaktische Überprüfungen
  - Erweiterbarkeit durch Definition neuer Tags/Attribute
- Nachteil: Lange Texte

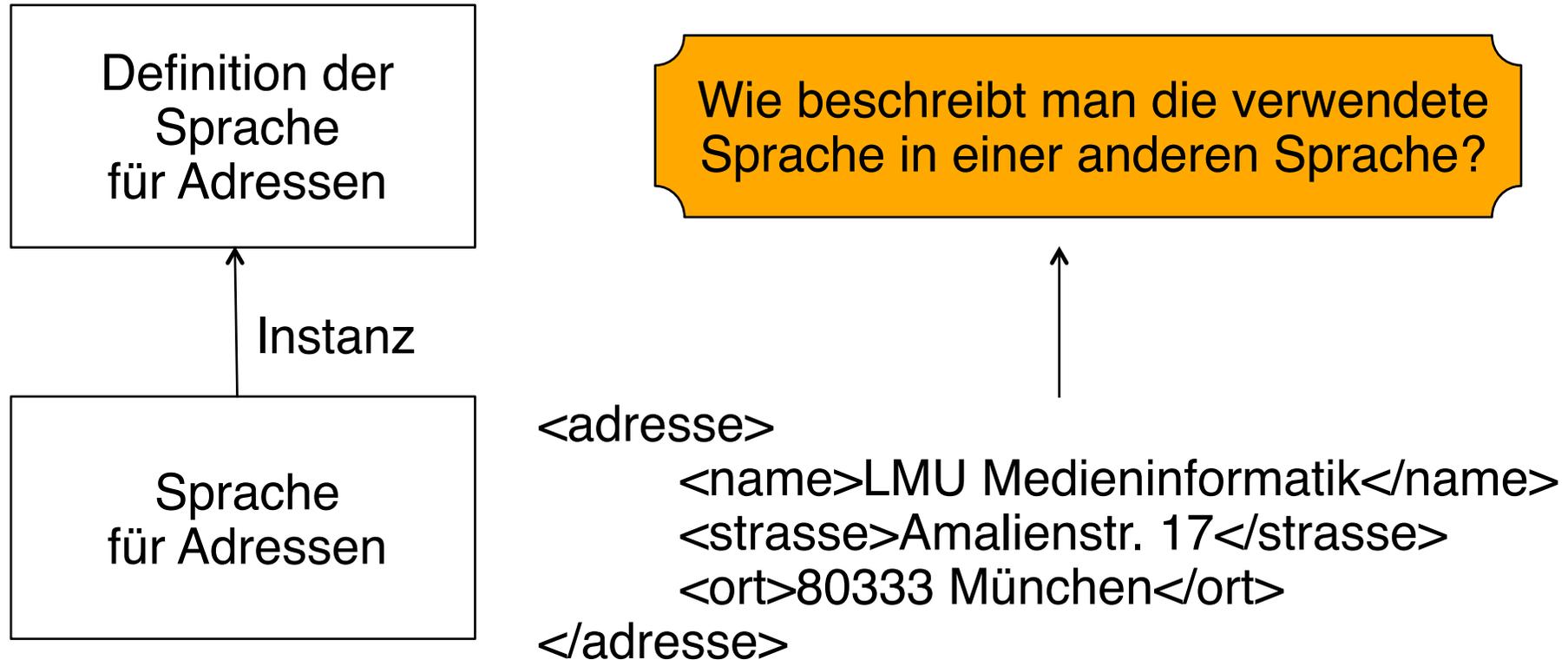
# Sprachenbeschreibung und Sprachenverwendung

Ebene der Sprachenbeschreibung (= Metaebene)



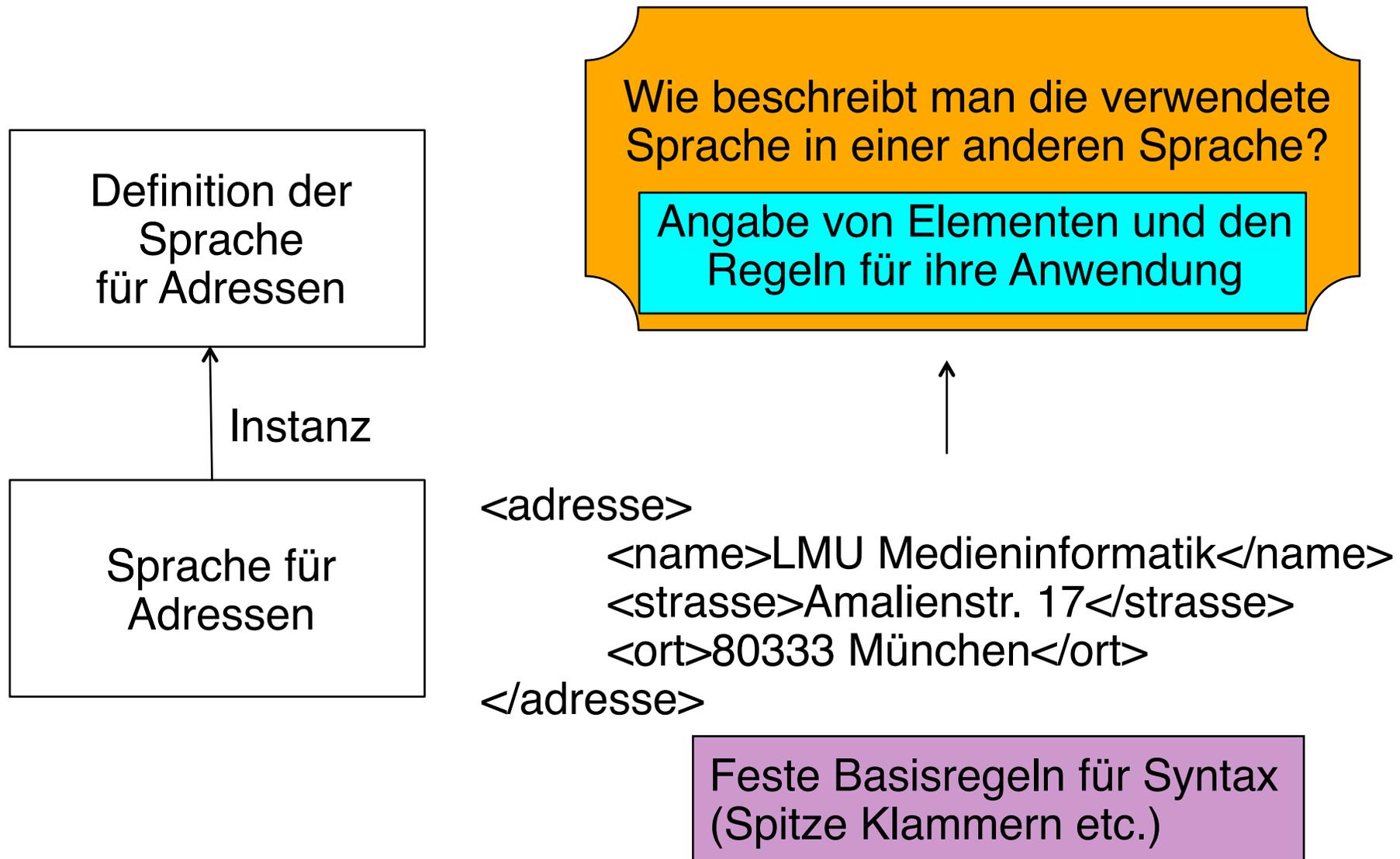
Ebene der Sprachenverwendung

# Metaebenen



Beispiel für eine relativ allgemeine Metasprache: Backus-Naur-Form (BNF)

# Vereinfachung durch Sprachfamilie



# Geschichte: SGML, HTML, XML, XHTML

- 1967: GenCode-Komitee
  - Norman Scharpf, Trennung Inhalt-Layout
- 1969: Goldfarb, Mosher, Lorrie (IBM):
  - Generalized Markup Language GML
- 1978: ISO-Standard 8879
  - Standard Generalized Markup Language SGML
  - Erlaubt Definition beliebiger *Dokumenttypen*
  - Sehr komplex, Verbreitung vorwiegend im akademischen Bereich und in der Definition weiterer Standards
- 1989: Berners-Lee, Cailleau
  - Hypertext Markup Language HTML, spezieller Dokumenttyp von SGML
- 1998: WWW Consortium (W3C)
  - eXtensible Markup Language (XML), Teilsprache von SGML
- 1999: Reformulierung von HTML als XML-Dokumenttyp
  - eXtensible Hypertext Markup Language XHTML (1.0)
- 2008: HTML5-Entwurf erlaubt zwei verschiedene Serialisierungen
  - HTML-spezifisch und als XML-Dokumenttyp (“XHTML5”)

# Beispiele von XML-Anwendungen

- *XML-Anwendung* = Definition eines XML-Dokumenttyps für einen bestimmten Zweck
- Grafik-Sprachen:
  - SVG (Scalable Vector Graphics): 2D-Vektorgrafik
  - X3D: 3D-Vektorgrafik, Fortführung von VRML
- Ablage- und Austauschformat für Bürosoftware:
  - Open Document Format (odt, ods, odp, odb, odg, odf)
  - Office Open XML (Microsoft/ECMA), in MS Office seit Office 2007
- VoiceXML
  - Dialogbeschreibung für natürlichsprachige Dialoge
- MusicXML
  - Austauschformat für Musiknoten (westliche Musiknotation)
- Diverse Gebiete der Naturwissenschaften:
  - MathML (für mathematische Formeln)
  - CML (für chemische Formeln)
  - BSMML (Bioinformatic Sequence Markup Language)
- ...

# XML als Basistechnologie

- Netzdienste:
  - News-Feeds mit RSS (Really Simple Syndication) basieren auf XML
    - » siehe später
  - iTunes-Podcasts sind RSS-Feeds (also XML)
  - Messaging-Protokoll XMPP basiert auf XML
  - ...
- Software:
  - "Preference"-files in MacOS X
  - diverse proprietäre Dateiformate

# XML-Dokumente

- Prolog:  
`<?xml version="1.0" encoding="UTF-8"?>`
- Element- und Attribut-Syntax wie in HTML, aber etwas strenger:
  - Jedes geöffnete Element muss explizit geschlossen werden.  
`<xy> ... </xy>`
  - Leere Elemente müssen mit „/>“ enden  
`<br/>`
  - Jedes XML-Dokument hat genau ein Wurzel-Element (*root*)
  - Strenge hierarchische Schachtelung von Elementen
  - Attributwerte immer in (einfachen oder doppelten) Anführungszeichen  
`<xy a="..."> ... </xy>`
  - Keine doppelten Attributwerte
  - Groß- und Kleinschreibung wird unterschieden

# Beispiel: Adresse als XML-Datei

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

```
<adresse>
```

```
<name>LMU Medieninformatik</name>
```

```
<strasse>Amalienstr. 17</strasse>
```

```
<ort>80333 München</ort>
```

```
</adresse>
```

adresse\_einfach.xml

# Beispiel: Fehlerhafte XML-Syntax

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

```
<address>
```

```
  <Name>LMU Medieninformatik</strasse>
```

```
  <straße>Amalienstr. 17</name>
```

```
  <ort>80333 München
```

```
</adresse>
```

Fehlermeldung eines XML-Editors:

Message Kind	Line	Details
Parse Error	4	Opening and ending tag mismatch: Name line 0 and strasse

adresse\_nonwf.xml

# XML Editoren

- XML Dateien können erstellt werden mit:
  - Normalen Texteditoren
  - Texteditoren mit Syntax-Unterstützung für XML
  - Speziellen XML-Editoren
  - Entwicklungsumgebungen (z.B. Eclipse)
  - ...
- Funktionen eines Editors mit XML-Sprachunterstützung:
  - Syntaktisch korrektes Highlighting von Textbestandteilen
  - Überprüfung der Syntaxregeln und Fehlermeldungen
  - Automatische Vervollständigung  
(macht Eingabe falscher Syntax fast unmöglich)

# QUIZ

- Sind diese Beispieltexte korrektes (wohlgeformtes) XML?
  1. `<t1> <t2> </t1> </t2>`
  2. `<t1> </t2> </t1> <t2>`
  3. `<t1> <t2> </t2> </t1>`
  4. `<t1> <t2/> <t1/>`
  5. `<t1> <t2/> </t1>`
  6. `<t1/> <t2/>`
  7. `</t1> </t2>`

# XML Inhaltsmodell (vereinfacht)

- Festlegung der zulässigen Werte für Elemente, Attribute etc.
  - Definition einer Sprache auf Meta-Ebene (also in einer weiteren Sprache)
  - Meist in separater Datei, aber auch als Bestandteil eines XML-Dokuments
- Zwei verschiedene Syntax-Alternativen:
  - „**XML Schema**“ (hier beschrieben)
  - klassische „**Document Type Definition**“-Syntax (siehe später)
- Datentypen für Inhalte in XML-Dateien (gemäß XML Schema):
  - String, Date, Time, Decimal, Integer, Boolean und andere
- Einfache Typen (haben nur einen Wert):
  - einfache Elemente, Attribute
- Komplexe Typen (umfassen mehrere Werte)
  - Elemente mit Attributen
  - Geschachtelte Elemente

# XML Schema-Definitionen (XSD)

- Definition einer XML-Sprache (zulässige Elemente usw.)
  - muss wieder in einer maschinenlesbaren Sprache geschehen
  - wird von Software-Werkzeugen ausgewertet, die die Definition anwenden
  - findet in der Meta-Sprache zu XML statt
- XML Schema Definitions
  - verwenden ***XML selbst*** als Metasprache (!)
- Vorteile der “rekursiven” Sprachdefinition von XML mit XML:
  - Keine zusätzliche Syntax, keine zusätzlichen Werkzeuge
  - Metasprache (XML Schema Definitions) kann verwendet werden, XML Schema Definitions selbst zu definieren!
- Wir betrachten XML-Dateien, die beschreiben, was in einer Klasse von anderen XML-Dateien erlaubt bzw. verlangt ist.
  - Unterscheidungs-Hilfsmittel (erste Erklärung):  
Bestandteile der Schema-Definitionssprache beginnen mit Präfix “**xs:**” !

# XSD: Einfache Element-Definition

- Syntax:

```
<xs:element name="elementname" type="elementtyp">
```

- *Elementname*:

- *elementname* muss der Syntax für XML-Elemente entsprechen (v.a. mit Buchstaben beginnen, keine Leerzeichen enthalten und nicht mit “xml”, “XML” oder “Xml” beginnen)
- Groß- und Kleinschreibung wird unterschieden

- *Elementtyp*:

- Vordefinierte XML-Typen (z.B. **xs:string**)
- Selbstdefinierte Typen

- Beispiele:

- ```
<xs:element name='strasse' type='xs:string' />
```
- ```
<xs:element name='hausnummer' type='xs:integer' />
```

# XSD: Spezielle numerische Typen

- XML Schema-Definitionen haben ein fein strukturiertes Typsystem
  - Untertypen bestehender Typen
  - Zusätzliche Einschränkungen („Restrictions“, z.B. für Zahlbereich)
- Beispiele für Untertypen von **Decimal**:
  - **decimal**: Allgemeine Dezimalzahlen (Gleitkommazahlen)
  - **integer**: Ganzzahl
  - **nonNegativeInteger**: Natürliche Zahl inklusive 0
  - **positiveInteger**: Echte natürliche Zahl (ohne 0)
  - **byte**: 8-Bit-Zahl mit Vorzeichen
  - **short**: 16-Bit-Zahl mit Vorzeichen
  - **int**: 32-Bit-Zahl mit Vorzeichen
  - **long**: 64-Bit-Zahl mit Vorzeichen

# XSD: Komplexe Elemente mit Unterelementen

- Syntax:

```
<xs:element name="elementname">
  <xs:complexType>
    <xs:sequence>
      ... Elements ...
    </xs:sequence>
  </xs:complexType>
</xs:element>
```

- *Elementname*: wie vorher
- *Elements*:
  - Liste von `<xs:element>`-Definitionen (ohne Trennzeichen)
  - Direkte Angabe weiterer Elemente (z.B. einfache Element-Definition)
  - (empfohlen:) Referenz auf Element eines separierten Typs, z.B.:  
`<xs:element ref='hausnummer' />`
- `<xs:sequence>`: “Indikator” für Anordnung der Elemente
  - nur eine von mehreren Möglichkeiten

# XSD-Beispiel: Komplexe Elemente

```
<xs:element name='haus' >
  <xs:complexType>
    <xs:sequence>
      <xs:element ref='strasse' />
      <xs:element ref='hausnummer' />
    </xs:sequence>
  </xs:complexType>
</xs:element>
<xs:element name='strasse' type='xs:string' />
<xs:element name='hausnummer' type='xs:positiveInteger' />
```

Passender XML-Inhalt:

```
<haus>
  <strasse>Amalienstr.</strasse>
  <hausnummer>17</hausnummer>
</haus>
```

# XSD-Beispiel: Definition von Adressen (ganz)

```
<?xml version="1.0" encoding="utf-8" standalone="no"?>
  <xs:schema xmlns:xs='http://www.w3.org/2001/XMLSchema'>
    <xs:element name='adresse'>
      <xs:complexType>
        <xs:sequence>
          <xs:element ref='name' />
          <xs:element ref='haus' />
          <xs:element ref='ort' />
        </xs:sequence>
      </xs:complexType>
    </xs:element>
    <xs:element name='haus'>
      <xs:complexType>
        <xs:sequence>
          <xs:element ref='strasse' />
          <xs:element ref='hausnummer' />
        </xs:sequence>
      </xs:complexType>
    </xs:element>
    <xs:element name='ort'>
      <xs:complexType>
        <xs:sequence>
          <xs:element ref='plz' />
          <xs:element ref='ortsname' />
        </xs:sequence>
      </xs:complexType>
    </xs:element>
    <xs:element name='name' type='xs:string' />
    <xs:element name='strasse' type='xs:string' />
    <xs:element name='hausnummer' type='xs:positiveInteger' />
    <xs:element name='plz' type='xs:positiveInteger' />
    <xs:element name='ortsname' type='xs:string' />
  </xs:schema>
```

# XSD-Beispiel: Definition von Adressen (Detail)

```
<?xml version="1.0" encoding="utf-8"?>
<xs:schema xmlns:xs='http://www.w3.org/2001/XMLSchema' >
  <xs:element name='adresse' >
    <xs:complexType>
      <xs:sequence>
        <xs:element ref='name' />
        <xs:element ref='haus' />
        <xs:element ref='ort' />
      </xs:sequence>
    </xs:complexType>
  </xs:element>
```

adresse\_einfach.xsd

...

- Wurzelement einer Schema-Definition ist **<schema>**
- Dass der Präfix “xs” sich auf die XML-Schema-Sprache bezieht, muss explizit deklariert werden (siehe zweite Zeile oben)

# XSD-Beispiel: Adresse als XML-Inhalt

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

<adresse xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
        xsi:schemaLocation="adresse_einfach.xsd">
  <name>LMU Medieninformatik</name>
  <haus>
    <strasse>Amalienstr.</strasse>
    <hausnummer>17</hausnummer>
  </haus>
  <ort>
    <plz>80333</plz>
    <ortsname>München</ortsname>
  </ort>
</adresse>
```

adresse\_einfach\_schema.xml

- Alle Vorgaben der Schema-Definition müssen eingehalten werden
- Dazu wird das zugehörige Schema durch einen “Vorspann” (beim Wurzelement) definiert

# Wohlgeformtheit und Gültigkeit

- Ein XML-Dokument ist *wohlgeformt (well-formed)*, wenn es den allgemeinen Regeln der XML-Syntax genügt.
  - Beispiel (*well-formed*, aber nicht *valid*):

```
<haus>  
  <straße>Amalienstr.</straße>  
  <hausnr>17</hausnr>  
  <hausteil>Rgb</hausteil>  
</haus>
```
- Ein XML-Dokument ist *gültig (valid)*, wenn es der angegebenen Schema-Definition entspricht.
- *Erweiterbarkeit*:
  - Anwendungen erzwingen Gültigkeit oft nicht
  - z.B. zusätzliche herstellerspezifische Tags – werden im Zweifelsfall ignoriert

# QUIZ

Ist dieser Text gemäß XML wohlgeformt?

```
<hausse>  
    <origine>Magie</origine>  
</hausse>
```

Ist dieser Text gemäß XML gültig, bzw. unter welchen Voraussetzungen?

```
<hausse>  
    <origine>Magie</origine>  
</hausse>
```

# XSD: Weitere Indikatoren für Unterelemente

- Reihenfolge-Indikatoren:
  - **<xs:sequence>**:  
Unterelemente müssen genau in angegebener Ordnung auftreten
  - **<xs:choice>**:  
Eines der Unterelemente muss auftreten
  - **<xs:all>**:  
Alle Unterelemente müssen in beliebiger Ordnung auftreten
- Häufigkeits-Indikatoren:
  - Attributwerte, die zu XSD-Elementdefinitionen angegeben werden können
  - **maxOccurs**: Maximalanzahl, in der ein Element auftreten kann
    - » Sonderwert „**unbounded**“ für unbegrenzt häufiges Auftreten
  - **minOccurs**: Minimalzahl, in der ein Element auftreten muss
  - Standardwert, wenn nicht angegeben, in der Regel 1

# XSD: Beispiel für Benutzung von Indikatoren

```
<xs:element name="adresse">
  <xs:complexType>
    <xs:sequence>
      <xs:element ref="name" maxOccurs="unbounded" />
      <xs:choice>
        <xs:element ref="postfach" />
        <xs:element ref="haus" />
      </xs:choice>
      <xs:element ref="ort" />
    </xs:sequence>
  </xs:complexType>
</xs:element>
<xs:element name="haus">
  <xs:complexType>
    <xs:sequence>
      <xs:element ref="strasse" />
      <xs:element ref="hausnummer" />
      <xs:element ref="hausteil" minOccurs="0" />
    </xs:sequence>
  </xs:complexType>
</xs:element>
```

adresse\_indikatoren.xsd  
adresse\_indikatoren1.xml  
adresse\_indikatoren2.xml

# XSD: Attribut-Definition

- Syntax:

```
<xs:attribute name="attrname" type="attrtyp">
```

- *Attrname:*

- *attrname* muss der Syntax für XML-Elemente entsprechen
- Groß- und Kleinschreibung wird unterschieden

- *Attrtyp:*

- Vordefinierte XML-Typen (z.B. **xs:string**)

- Standardwert für Attribut: **default="defaultwert"**

- Benötigtes Attribut: **use="required"**

- Beispiele:

```
– <xs:attribute name="sprache" type="xs:string"  
  default="DE" />
```

```
– <xs:attribute name="adresstyp" type="xs:string"  
  use="required" />
```

# XSD: Beispiele für Benutzung von Attributen

```
<xs:element name="adresse">
  <xs:complexType>
    <xs:sequence>...</xs:sequence>
    <xs:attribute
      name="adresstyp" type="xs:string" use="required" />
  </xs:complexType>
</xs:element>
<xs:element name="ortsname">
  <xs:complexType>
    <xs:simpleContent>
      <xs:extension base="xs:string">
        <xs:attribute
          name="sprache" type="xs:string" default="DE" />
      </xs:extension>
    </xs:simpleContent>
  </xs:complexType>
</xs:element>
```

adresse\_attribute.xsd  
adresse\_attribute1.xml  
adresse\_attribute2.xml

- Achtung: Einfache Elemente werden durch Attribute komplex, und Attribute müssen als „Erweiterung“ eines Basistyps angegeben werden

# XSD-Beispiel: Adresse mit Attributen

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

<adresse xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xsi:schemaLocation="adresse_attribute.xsd"
  adresstyp="Arbeit">
  <name>LMU Medieninformatik</name>
  <name>Prof. Hußmann</name>
  <postfach>
    PF2012
  </postfach>
  <ort>
    <plz>80333</plz>
    <ortsname sprache="EN">Munich</ortsname>
  </ort>
</adresse>
```

# Elemente vs. Attribute

- Zusatzinformation zu Elementen als Unterelemente oder Attribute?

```
<buch>  
  <isbn>3-932588-96-7</isbn>  
  <autor>Fritz Müller</autor>  
  ...  
</buch>
```

oder

```
<buch  
  isbn="3-932588-96-7"  
  autor="Fritz Müller">  
  ...  
</buch>
```

- Unterelemente
  - sind leichter zu lesen
  - ermöglichen Wiederholungen
  - können weiter in Elemente untergliedert werden
  - können auch in anderem Kontext genutzt werden
- Attribute
  - sind leicht zu überprüfen
  - erlauben keine Wiederholungen
  - verringern die Hierarchietiefe
  - binden die Zusatzinformation eng an das zugehörige Element

# Document Type Definitions (DTDs)

- Einfachere Syntax, weniger genaues Typsystem (z.B. für Zahlen)
- Beispiel, validiert die gleichen XML-Dateien wie das Beispiel-Schema:

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

```
<!ELEMENT adresse (name+, (haus|postfach), ort)>
<!ELEMENT haus (strasse, hausnummer, hausteil?)>
<!ELEMENT ort (plz, ortsname)>
<!ELEMENT name (#PCDATA)>
<!ELEMENT strasse (#PCDATA)>
<!ELEMENT hausnummer (#PCDATA)>
<!ELEMENT hausteil (#PCDATA)>
<!ELEMENT postfach (#PCDATA)>
<!ELEMENT plz (#PCDATA)>
<!ELEMENT ortsname (#PCDATA)>
<!ATTLIST adresse adresstyp CDATA #REQUIRED>
<!ATTLIST ortsname sprache CDATA "DE">
```

# Abkürzungen mit (internen) Entities in DTDs

- Hintergrund für HTML-Abkürzungen wie `&szlig;`
- (Interne) Entities in DTDs:
  - Abkürzungsmechanismus für XML-Abschnitte
  - Paar Name – Inhalt
- Syntax (Definition):  
`<!ENTITY Entityname Entityinhalt >`
- Syntax (Verwendung):  
`&Entityname;`

# Namensräume (*Namespaces*)

- Mischen von XML-Information, die mehreren verschiedenen DTDs entspricht?
  - Mehrdeutige Elemente?
- Namespace-Deklaration
  - Syntax:  
***<Elementname xmlns:Namensraumname="URI" ...>***
  - Definiert frei gewählten *Namensraumnamen*
    - » *URI* definiert den Urheber des Namensraums.
    - » Namensraum verwendbar in untergeordneten Dokumentteilen
    - » Deshalb meist bei Wurzel-Elementen angegeben
  - *Namensraumname* wird als *Präfix* verwendet:  
***Namensraumname : Element***
    - » Unterscheidung von evtl. gleichnamigen anderen Elementen

# Namensräume (*Namespaces*), *Forts.*

- Default-Namensraum
  - gilt für Elemente ohne Präfix
  - Deklariert durch  
`<Elementname xmlns="URI" ...>`
  - Beispiel:  
`<html xmlns="http://www.w3.org/1999/xhtml" ...>`

# 9. Web-Dokumente

- 9.1 Generische Auszeichnungssprachen: XML
- 9.2 XML und Style Sheets ←
- 9.3 XML für Multimedia: SMIL
- 9.4 XML für Web-Informationsdienste: RSS

## Medieninformatik-Buch: Kapitel 10



Weiterführende Literatur:

H. Vonhoegen: Einstieg in XML, Galileo Computing 2013

<http://www.w3schools.com/xml/>

# QUIZ

- Kann man Texte schreiben, die in einem Web-Browser vom Server abgerufen und korrekt formatiert werden, ***ohne HTML zu benutzen?***

# CSS und XML

- XML-Dateien enthalten „reinen Inhalt“ (gemäß gegebener Struktur)
  - Zur Anzeige z.B. im Browser Zusatzangaben nötig
- Alternative Wege von einer XML-Datei zu einer Browseranzeige:
  - Cascading Style Sheets
  - Transformation in HTML-Text (z.B. mit Transformationssprache XSLT)
  - Kombination beider Ansätze
- Cascading Style Sheets
  - Separate Datei(en) mit Formatierungsangaben
  - Anbindung an XML-Dokument durch eine sogenannte *processing instruction* (PI):  

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

# Beispiel: CSS-Datei für Dokumenttyp “adresse”

```
adresse {  
    font-family:sans-serif  
}
```

```
name {  
    display:block;  
    font-size:120%;  
    font-weight:bold;  
}
```

```
haus {  
    display:block;  
    padding-top:10pt  
}
```

```
ort {  
    display:block;  
}
```

adresse\_attribute2\_style.xml  
adressenstil.css

**LMU Medieninformatik**  
**Prof. Hußmann**

Amalienstr 17  
80333 München

# Fortgeschrittene Konzepte in CSS

- Pseudo-Formate:
  - z.B. `display:none` zum Ausblenden (nicht darstellen)
- Pseudo-Elemente:
  - z.B. `:first-letter`, `:first-line` zur speziellen Formatierung von Textteilen
  - z.B. `:before`, `:after` zum Modifizieren von Texten bei der Anzeige
- Pseudo-Klassen
  - z.B. `:hover`, `:focus`, `:active` zur Darstellung abhängig von Benutzeraktionen
- Kontextabhängige Formatierung
  - z.B. für Elemente abhängig von bestimmten Attributwerten
  - z.B. für Unterelemente abhängig von den im Dokument vorhandenen Oberelementen
- Strukturierung von Formatierungsinformation
  - Vererbung und verschiedene Formen zur Einbindung von Stylesheets

# Beispiel: CSS-Datei für Dokumenttyp “adresse”

```
adresse {  
    font-family:sans-serif  
}  
adresse:before {  
    content: "Adresstyp: " attr(adresstyp) ;  
    color:blue;  
}  
name:hover {  
    color:red  
}  
ortsname[sprache=DE]  
    {color:green}
```

adresse\_attribute2\_style2.xml  
adressenstil2.css

Adresstyp: Arbeit  
**LMU Medieninformatik**  
**Prof. Hußmann**

Amalienstr 17  
80333 München

# 9. Web-Dokumente

- 9.1 Generische Auszeichnungssprachen: XML
- 9.2 XML und Style Sheets
- 9.3 XML für Multimedia: SMIL 
- 9.4 XML für Web-Informationendienste: RSS

Weiterführende Literatur:

Dick Bulterman, Lloyd Rutledge: SMIL 3.0 – Interactive Multimedia for the Web, Mobile Devices and Daisy Talking Books, Springer 2008

# SMIL - Idee und Geschichte

- Synchronized Multimedia Integration Language (gesprochen: "Smile")
- Standardsprache für die koordinierte Kombination von zeitabhängigen Medien zu einer Multimedia-Präsentation
  - zeitliche Abhängigkeiten im Ablauf
  - berücksichtigt auch nicht-zeitabhängige Medientypen (Text, Standbild)
  - auch geeignet für "Streaming", d.h. kontinuierliches Laden von Mediendaten über das Netz
- Standardisierung durch W3C (WWW Consortium)
  - Erster Entwurf November 1997
  - SMIL 1.0 Standard Juni 1998
  - ab 1998: Implementierungen u.a. durch Real, Apple, [ambulantplayer.org](http://ambulantplayer.org)
  - 1999: Pläne für eine erweiterte und verbesserte Fassung ("Boston SMIL")
  - SMIL 2.0 Standard August 2001
  - SMIL 2.1 Recommendation Dec. 2005 (z.B. Profil für mobile Endgeräte)
  - SMIL 3.0 Recommendation Dec. 2008

# Grundstruktur einer SMIL-Datei

```
<smil xmlns="http://www.w3.org/2001/SMIL20/Language">
  <head>
    <layout>
      <root-layout width="356" height="356"
        backgroundColor="black"/>
      <region id="imgReg" width="256" height="256"
        left="50" top="50"/>
    </layout>
  </head>
  <body>
    <seq>
      
      
      
    </seq>
  </body>
</smil>
```

Spatiale Struktur  
(Layout)

Temporale Struktur  
(Ablauf)

# Beispiel: Multimediale Diashow (1)

```
<smil xmlns="http://www.w3.org/2001/SMIL20/Language">
  <head>
    <layout>
      <root-layout width="356" height="356"/>
      <region id="brush_region" z-index="1"/>
      <region id="img_region" width="256" height="256"
        left="50" top="50" z-index="2"/>
    </layout>
    <transition id="img_wipe" type="barWipe"
      dur="3s"/>
    <transition id="bkg_wipe" type="barWipe"
      direction="reverse" dur="3s"/>
  </head>
```

slideshow.smil

## Beispiel: Multimediale Diashow (2)

```
...
<body>
  <par>
    <seq>
      
      ...
    </seq>
    <seq>
      <brush color="green" region="brush_region"
        ... transIn="bkg_wipe" fill="transition"/>
    </seq>
    <audio src="...mp3" end="32s"/>
  </par>
</body>
</smil>
```

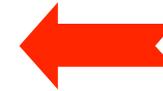
# 9. Web-Dokumente

9.1 Generische Auszeichnungssprachen: XML

9.2 XML und Style Sheets

9.3 XML für Multimedia: SMIL

9.4 XML für Web-Informationsdienste: RSS



Weiterführende Literatur:

Jörg Kantel: RSS und Atom kurz und gut, O'Reilly 2007

<http://cyber.law.harvard.edu/rss/rss.html>

# Really Simple Syndication RSS

- *Syndikation*: Zusammenführen und Integrieren von Informationen (Nachrichten) aus verschiedenen Quellen
- RSS:
  - XML-basiertes Format für Nachrichtenquellen im Internet
  - 1997 von der Firma Userland definiert
  - 1999: my.Netscape.com ("RDF/Rich Site Summary")
  - Heute meistverwendetes Format für Nachrichten, Weblogs, Podcasts
- Konkurrenzformat: *Atom Syndication Format* (ASF) (ebenfalls XML)
- Grundstruktur:
  - *Channel* ist Liste von *Items*
  - Jedes *Item* ist durch einen *Globally Unique Identifier* (guid) definiert, meist ein Link
  - Umfangreiche Möglichkeiten für *Metadaten* zu Items und Channels
- Beispiel:
  - Ein *Podcast* ist in der Regel eine RSS-Datei

# Beispiel: RSS Feed zu einer Vorlesung (1)

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

```
<rss version="2.0">
```

```
<channel>
```

```
  <title>Digitale Medien News Wintersemester 10/11</title>
```

```
  <link>http://www.medien.ifi.lmu.de/lehre/ws1011/dm/</link>
```

```
  <description>Site Updates und Mitteilungen für die Vorlesung  
  Digitale Medien im Wintersemester 10/11</description>
```

```
  <language>de</language>
```

```
  <pubDate>
```

```
    Fri, 01 Oct 2010 12:29:32 GMT
```

```
  </pubDate>
```

```
  <lastBuildDate>
```

```
    Tue, 25 Jan 2011 18:37:59 GMT
```

```
  </lastBuildDate>
```

```
  <docs>http://blogs.law.harvard.edu/tech/rss</docs>
```

```
  <managingEditor>webmaster@ifi.lmu.de</managingEditor>
```

# Beispiel: RSS Feed zur Vorlesung (2)

```
<item>
  <title>Übungsblatt 9 verfügbar</title>
  <link>http://www.medien.ifi.lmu.de/lehre/ws1011/dm/</link>
  <description>
    <![CDATA[
      Das neunte Übungsblatt kann nun heruntergeladen werden.
      Die Abgabe muss spätestens bis zum 19.01.2011 um 14:00
      erfolgen.
    ]]>
  </description>
  <pubDate>
    Mon, 10 Jan 2011 11:22:18 GMT
  </pubDate>
  <guid>
    http://www.medien.ifi.lmu.de/lehre/ws1011/dm/blatt9.pdf
  </guid>
</item>
```

# Beispiel: dm\_podcast.rss (Auszug)

```
<?xml version="1.0" encoding="UTF-8"?>
<rss xmlns:itunes="http://www.itunes.com/dtds/podcast-1.0.dtd"
  version="2.0">
  <channel>
    <title>Vorlesung Digitale Medien Wintersemester 2010/11</title>
    <itunes:author>Heinrich Hussmann, LMU</itunes:author>
    ...
    <item>
      <title>Informationstheorie, Codierung Teil I</title>
      <description>Es wird eine Einführung ...</description>
      <guid>http://www.medien.ifi.lmu.de/team/
        heinrich.hussmann/files/dm2a.m4b</guid>
      <enclosure url="http://www.medien.ifi.lmu.de/team/
        heinrich.hussmann/files/dm2a.m4b"
        length="23424928" type="audio/x-m4a"></enclosure>
      <pubDate>Fri, 22 Oct 2010 22:30:00 +0200</pubDate>
      <itunes:explicit>no</itunes:explicit>
      <itunes:duration>01:25:44</itunes:duration>
    </item>
    ...
  </channel>
</rss>
```