

Übung zur Vorlesung

Digitale Medien

Vorlesung: Heinrich Hußmann

Übung: Renate Häuslschmid

Ludwig-Maximilians-Universität München

Wintersemester 2016/17

Klausur

- Donnerstag, 27.02.17, 10:00 - 13:00 Uhr (90min)
- **Anmeldung** bis Mo 20.02.2017, 09:00Uhr
- **Abmeldung** bis Fr. 23.02.2017, 09:00Uhr
- Vorsicht: Anmeldung zu einer Prüfung und unentschuldigtes Nichterscheinen zählt als durchgefallen! Meldet euch rechtzeitig ab, wenn ihr nicht mitschreiben möchtet.
- Wer nicht rechtzeitig angemeldet ist, kann nicht mitschreiben!

Open Book

- Erlaubt:
 - Ausdrücke (Folien, Übungsblätter, Internetquellen...)
 - Bücher
 - Handschriftliche Notizen
- Nicht erlaubt:
 - Elektronische Hilfsmittel (Handy, Notebook, Smartwatches ...)

Mitbringen

- Studentenausweis
- Amtlicher Lichtbildausweis (Reisepass, Personalausweis, Führerschein. Nicht MVV, Büchereiausweis oder ähnliches!!)
- Taschenrechner (nicht programmierbar!)

Bonuspunkte

- Insgesamt 100 Klausurpunkte
 - ➔ Maximal 12% $\hat{=}$ 12 Klausurpunkte durch Bonuspunkte
- Können auch zum Bestehen helfen.

Nachholklausur

- am Ende der Semesterferien (voraussichtlich in den letzten beiden Ferienwochen)
- auch zur Notenverbesserung

eXtensible Markup Language (XML)

Metasprache für hierarchisch strukturierte Textdaten.

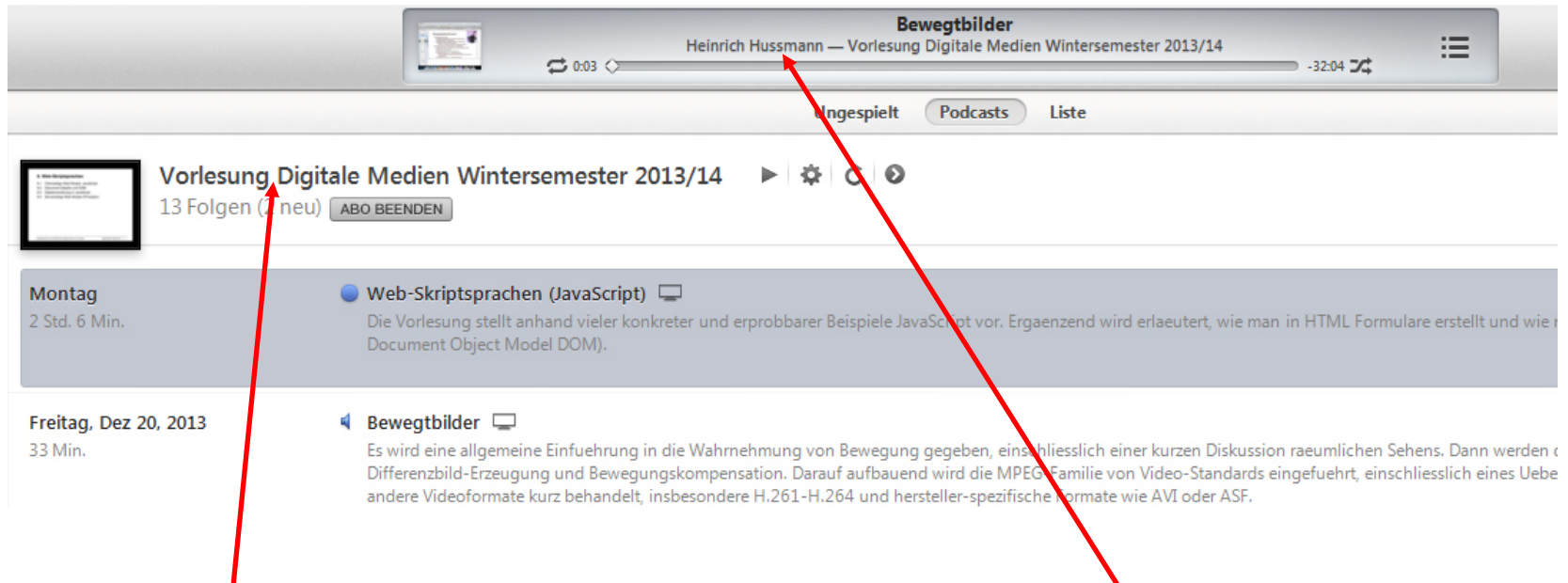
Varianten von XML sind z.B.

- XHTML (Webseiten),
- SVG (Vektorgrafik),
- SMIL (Animationen),
- X3D (3D-Szenen),
- RSS (Webfeeds),
- etc.

Vorteile u.a.

- von Menschen und Maschinen lesbar
- Trennung von Inhalt und Präsentation
- beliebig erweiterbar

Beispiel



```
<title>  
  Vorlesung Digitale Medien Wintersemester 2008/09  
</title>
```

```
<itunes:author>  
  Heinrich Hussmann  
</itunes:author >
```



```
<?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 2013/14</title>
    <itunes:author>
      Heinrich Hussmann,
      Ludwig-Maximilians-Universitaet München
    </itunes:author>
    <link>http://www.medien.ifi.lmu.de/lehre/ws1314/dm/</link>
    <itunes:subtitle>
      Eine Einführung in Technologien
      für digitale Medien
    </itunes:subtitle>
    <description>
      Es werden Basiskenntnisse über die physikalischen und
      wahrnehmungsphysiologischen Grundlagen der Realisierung
      digitaler Mediensysteme sowie elementare Techniken der
      digitalen Medienrepräsentation (einschließlich
      Datenkompressionstechniken) vermittelt. Es soll ein
      grundlegendes Verständnis der verschiedenen
      Multimedia-Datenformate und ihrer Vor- und Nachteile
      erworben werden.
    </description>
    <language>de-de</language>
    <copyright>Heinrich Hussmann, LMU</copyright>
    <itunes:category text="Education"></itunes:category>
  </channel>

```

...

(http://www.medien.ifi.lmu.de/lehre/ws1314/dm/dm_podcast.rss)

XML Schema-Definitionen (XSD)

- Beschreibt die Struktur/die Regeln einer XML-Sprache
 - Ein Dokument ist valide, wenn es den Regeln entspricht, die in dem Schema (oder einer DTD) festgelegt wurden
- Nachfolger von DTD
- Unterstützt verschiedene Datentypen
 - Z.B.: String, Date, Numeric
- Selbst in XML geschrieben

XSD Element

- **Syntax:**
`<xs:element name="someName" type="someType"/>`
- **Default Werte**
`<xs:element ... default="defaultValue"/>`
- **Unveränderbare Werte**
`<xs:element ... fixed="immer das hier"/>`
- **Anzahl**
`<xs:element ... minOccurs="min" maxOccurs="max"/>`

XSD Element

- Beispiel:

```
<xs:element name="vorname" type="xs:string" default="Hans" />
```

```
<vorname>Martin</vorname>
```

XSD Element (Complex Type)

- **Syntax:**

```
<xs:element name="someName">
  <xs:complexType>
    <xs:sequence>
      <xs:element name="child1" type="xs:string"/>
      ...
    </xs:sequence>
  </xs:complexType>
</xs:element>
```

Kinder anordnen

- `<xs:sequence>`
Die Kinder müssen in dieser Reihenfolge im Dokument vorkommen.
Attribute: `maxOccurs`, `minOccurs`
- `<xs:choice>`
Eines der Kinder muss im Dokument vorkommen.
Attribute: `maxOccurs`, `minOccurs`

XSD Attribute

- **Syntax:**
`<xs:attribute name="someName" type="someType"/>`
- **Default Werte**
`<xs:attribute ... default="defaultValue"/>`
- **Fixed Werte**
`<xs:attribute ... fixed="immer das hier"/>`
- **Required Attribute:**
`<xs:attribute ... use="required"/>`

XSD Attribute

- **Beispiel:**

```
<xs:element name="someName">
  <xs:complexType>
    <xs:sequence>
      ...
    </xs:sequence>
    <xs:attribute name="att1" type="xs:string"/>
  </xs:complexType>
</xs:element>
```


Datentyp String

- Verschiedene Typen. Z.B: ID, IDREF, string, language
- Restriktionen. Z.B.: length, maxLength, minLength

Beispiel:

Eindeutiges Element „matrikelnummer“ und Element „name“, das maximal 20 Zeichen haben darf. Außerdem ein Element „teilnehmer“, das auf einen Studenten verweist.

```
<xs:element name="matrikelnummer" type="xs:ID"/>
```

```
<xs:element name="name" type="xs:string" maxLength="20"/>
```

```
<xs:element name="teilnehmer" type="xs:IDREF"/>
```

Datentyp Date

- Verschiedene Typen. Z.B: `date`, `dateTime`, `gDay`, `gMonth`
- Restriktionen. Z.B.: `maxInclusive`, `maxExclusive`

Beispiel: Element „datum“ mit beliebigem Datum.

```
<xs:element name="datum" type="xs:date"/>
```

Datentyp Numeric

- **Verschiedene Typen.** Z.B: `byte`, `decimal`, `integer`, `negativeInteger`
- **Restriktionen.** Z.B.: `maxExclusive`, `maxInclusive`

Beispiel:

Element „negativ“, das nur negative Integer zulässt
(z.B. `<negative>-20</negative>`).

```
<xs:element name="negativ" type="xs:negativeInteger"/>
```

Buch mit Kochrezepten:

- Buch enthält mind. ein Rezept
- Jedes Rezept hat einen Namen, mind. eine Zutat und mind. einen Arbeitsschritt
- Die Arbeitsschritte sind durchnummeriert (als Attribute)

XSD-Übersicht:

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

```
<kochbuch>
  <rezept>
    <name>Apfelkuchen</name>
    <zutat>Aepfel</zutat>
    <zutat>Mehl</zutat>
    <zutat>Zucker</zutat>
    <schritt nummer="1">
      ...
    </schritt>
    <schritt nummer="2">
      ...
    </schritt>
  </rezept>
</kochbuch>
```

Buch mit Kochrezepten:

- Buch enthält mind. ein Rezept
- Jedes Rezept hat einen Namen, mind. eine Zutat und mind. einen Arbeitsschritt
- Die Arbeitsschritte sind durchnummeriert (als Attribute)

XSD-Übersicht:

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

```
<?xml version="1.0"?>
<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema">

<xs:element name="kochbuch">
  <xs:complexType>
    <xs:sequence>
      <xs:element name="rezept" maxOccurs="unbounded">
        <xs:complexType>
          <xs:sequence>
            <xs:element name="name" type="xs:string"/>
            <xs:element name="zutat" type="xs:string" maxOccurs="unbounded"/>
            <xs:element name="schritt" type="xs:string" maxOccurs="unbounded">
              <xs:complexType>
                <xs:attribute name="nummer" type="xs:integer" use="required"/>
              </xs:complexType>
            </xs:element>
          </xs:sequence>
        </xs:complexType>
      </xs:element>
    </xs:sequence>
  </xs:complexType>
</xs:element>

</xs:schema>
```

XML & CSS

XML-Dateien enthalten nur Inhalt, die Darstellung ist nicht vorgegeben.
Darstellung einer XML-Datei ist möglich über CSS 2.0 mit den Standard-, sowie speziellen neuen Elementen:

```
<leute>
  <person geschlecht="m" alter="48">
    <vorname>Hans</vorname>
    <nachname>M&#x00FC;ller</nachname>
  </person>
  <person geschlecht="w" alter="72">
    <vorname>Anneliese</vorname>
    <nachname>Schmidt</nachname>
  </person>
</leute>
```

```
person
{
    color:#0000FF;
    display: block;
}
person:after
{
    content: "("
        attr(geschlecht) ", "
        attr(alter) ")";
}
```

```
- <leute>
- <person geschlecht="m" alter="48">
  <vorname>Hans</vorname>
  <nachname>Müller</nachname>
</person>
- <person geschlecht="w" alter="72">
  <vorname>Anneliese</vorname>
  <nachname>Schmidt</nachname>
</person>
</leute>
```

Hans Müller (m, 48)
Anneliese Schmidt (w, 72)

XML & CSS

```
<?xml version="1.0" encoding="UTF-8"?>
<?xml-stylesheet type="text/css" href="present.css"?>

<leute>
  <person geschlecht="m" alter="48">
    <vorname>Hans</vorname>
    <nachname>M&#x00FC;ller</nachname>
  </person>
  <person geschlecht="w" alter="72">
    <vorname>Anneliese</vorname>
    <nachname>Schmidt</nachname>
  </person>
</leute>
```

leute.xml

Hans Müller (m, 48)
Anneliese Schmidt (w, 72)

Einbindung über
<xml-stylesheet>-Element:

```
person
{
    color: #0000FF;
    display: block;
}
person:after
{
    content: "("
        attr(geschlecht)
        ", "
        attr(alter) ")";
}
```

present.css

XML & CSS

Beispieldatei:

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

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

einfügen

"present.css" erstellen und ändern.
Ergebnisse im Firefox überprüfen.

XML & CSS Beispiel

XML + CSS: <http://de.selfhtml.org/xml/darstellung/css.htm>

'Generated Content' (CSS 2.0): <http://www.w3.org/TR/CSS21/generate.html>

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

```
<breakfast_menu>
```

```
  <food>
```

```
    <name>Belgian Waffles</name>
```

```
    <price>$5.95</price>
```

```
    <description>two of our famous Belgian Waffles with plenty of real  
                maple syrup</description>
```

```
    <calories>650</calories>
```

```
  </food>
```

```
  <food>
```

```
    <name>Strawberry Belgian Waffles</name>
```

```
    <price>$7.95</price>
```

```
    <description>light Belgian waffles covered with strawberries and  
                whipped cream</description>
```

```
    <calories>900</calories>
```

```
  </food>
```

```
  <food>
```

```
    ...
```

```
</breakfast_menu>
```

Belgian Waffles only \$5.95!

two of our famous Belgian Waffles with plenty of real maple syrup

Strawberry Belgian Waffles only \$7.95!

light Belgian waffles covered with strawberries and whipped cream

XML & CSS Beispiel

```
food
{
    display: block;
}

price:after
{
    content: "!";
}

name
{
    color:#FF0000;
    font-weight: bold;
}

description
{
    font-style: italic;
    font-size: small;
    display: block;
}

price:before
{
    font-style: italic;
    content: "only ";
}

calories
{
    visibility: hidden;
}
```

Übungsblatt 11

- Übungsblatt 11:
<https://www.medien.ifi.lmu.de/lehre/ws1516/dm/>
- Abgabe bis Freitag den 05.02.2016, 09:00 Uhr
morgens in [UniWorX](#)

www.digitale-medien-uebung.de

ONLINE LERNPLATTFORM

Lösung zu Aufgabe 1

```
<?xml version="1.0" encoding="utf-8" ?>
<root>
  <vorlesung semester="ws1314">
    <titel>Vorlesung Digitale Medien</titel>
    <dozent>Prof. Dr. Heinrich Hussmann</dozent>
    <uebung>

<korrektur><korrektor>Hanna</korrektor></korrektur>
  </uebung>
</vorlesung>
  <vorlesung semester="ws1213">
    <Titel>Vorlesung Digitale Medien</Titel>
    <dozent>Prof. Dr. Heinrich Hussmann</dozent>
    <uebung>

<korrektur><korrektor>Alexander</korrektor></korrektur>
  </uebung>
</vorlesung>
</root>
```

Lösung zu Aufgabe 2

```
vorlesung:before{
    content:attr(semester);
    color:red;
}
vorlesung {
    display:block;
    padding-bottom:15pt;
}
titel {
    display:block;
    font-size:120%;
    font-weight:bold;
}

dozent:before{
    content: "Leitung: ";
    font-style:italic;
}
dozent {
    display:block;
}
uebungsleitung:before{
    content: "Übungsleitung: ";
    font-style:italic;
}
uebungsleitung {
    display:block;
}
```

Lösung zu Aufgabe 3

Element „ponyhof“

```
<xs:element name="ponyhof">
  <xs:complexType>
    <xs:sequence>
      <xs:element name="pony" minOccurs="0" maxOccurs="unbounded">
        ...
      </xs:element>
      <xs:element name="reiter" minOccurs="0" maxOccurs="unbounded">
        ...
      </xs:element>
    </xs:sequence>
  </xs:complexType>
</xs:element>
```

Lösung zu Aufgabe 3

Element „pony“

```
<xs:element name="pony" minOccurs="0" maxOccurs="unbounded">
  <xs:complexType>
    <xs:sequence>
      <xs:element name="name" type="xs:ID"/>
      <xs:element name="rasse" type="xs:string" minOccurs="0"/>
    </xs:sequence>
    <xs:attribute name="mutter" type="xs:IDREF" />
    <xs:attribute name="vater" type="xs:IDREF" />
  </xs:complexType>
</xs:element>
```


Lösung zu Aufgabe 3

Element „reiter“

```
<xs:element name="reiter" minOccurs="0" maxOccurs="unbounded">
  <xs:complexType>
    <xs:sequence>
      <xs:element name="name" type="xs:string"/>
      <xs:element name="alter" type="xs:integer" minOccurs="0"/>
      <xs:element name="kennung" type="xs:ID"/>
    </xs:sequence>
  </xs:complexType>
</xs:element>
```