



XPath & Co.

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

letzte Woche

- ☑ XML Parser: SAX, DOM & StAX
- ☑ Schema-Übersetzer

heutige Vorlesung

- Navigation & Verknüpfungen (XPath, XPointer, XLink)
- XSLT-Einführung

X-Familie

Heute

- XML Path Language (XPath)
- XML Linking Language (XLink)
- XML Base (XBase)
- XML Pointer Language (XPointer)
- XML Query (XQuery)

nächste Woche

- XSLT

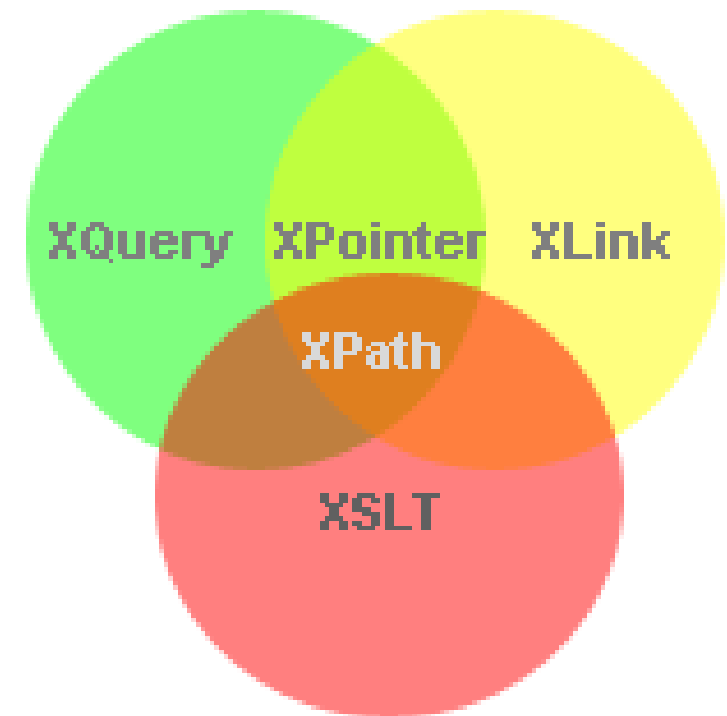


Bild-Quelle: <http://www.w3schools.com/xquery/default.asp>



XML Path Language (XPath)

XPath

- Standard zum Zugreifen auf beliebige Teile eines XML-Dokuments
- keine XML-Anwendung
- wird von XSLT benutzt
- Adressierungspfad eines Dateisystems ähnlich aber wesentlich mächtiger:
z.B. /order/item
- XPath 1.0 – W3C-Recommendation seit Nov. 1999
 - <http://www.w3.org/TR/xpath>
- XPath 2.0 – W3C-Recommendation seit Jan. 2007
 - <http://www.w3.org/TR/xpath20/>

- ähnliches Modell wie in DOM
 - XML-Dokument als Baum mit Elementen, Attributen und PCDATA als Knoten
- virtuelle Dokument-Wurzel (Wurzelknoten):
durch "/" repräsentiert (links von "/" steht nichts)
 - ⇒ Wurzel-Element immer Kind von "/":
z.B. /root

Knotentypen (I)

- **Wurzelknoten**

- oberster Knoten im Baum, dessen Kind der Elementknoten des Dokuments ist
- string-Wert: Verkettung der Zeichendaten aller Textknoten-Kinder in der Dokumentenreihenfolge

- **Elementknoten**

- Knoten für ein Element
- string-Wert: Verkettung der Zeichendaten aller Textknoten-Kinder des Elements

- **Attributknoten**

- Knoten für jedes Element zugeordnete Attribut
- string-Wert: Normalisierter Attributwert

Knotentypen (II)

- **Textknoten**

- Knoten der Zeichendaten enthält
- string-Wert: die Zeichendaten des Textknotens

- **Namensraumknoten**

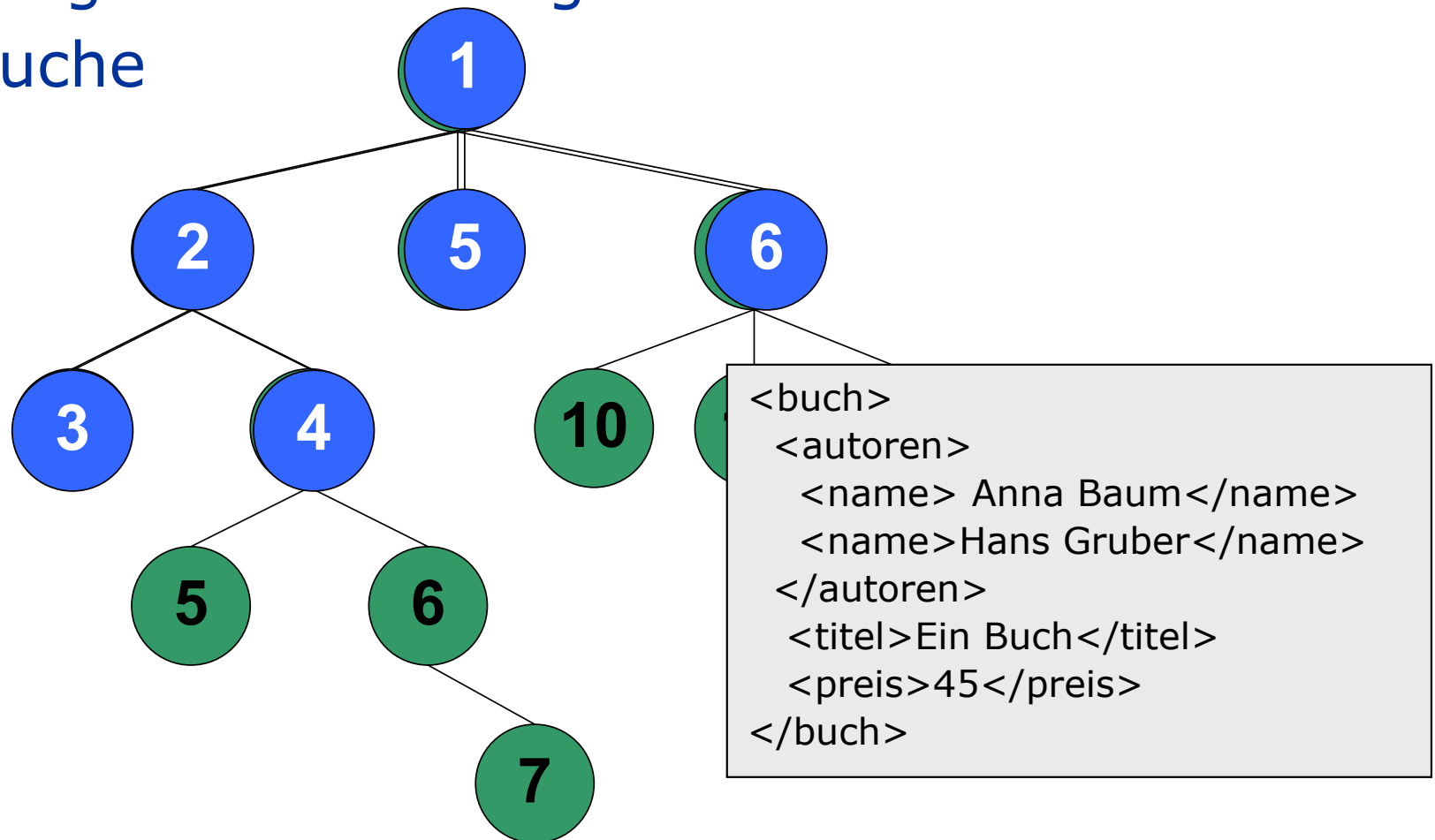
- der Namensraum ist jeweils einem Elementknoten als Elternknoten zugeordnet, ist aber nicht Kind dieses Elementknoten
- string-Wert: URI des Namensraum

- **Kommentarknoten**

- Knoten für jeden einzelnen Kommentar
- string-Wert: Kommentarinhalt

Dokumentenreihenfolge

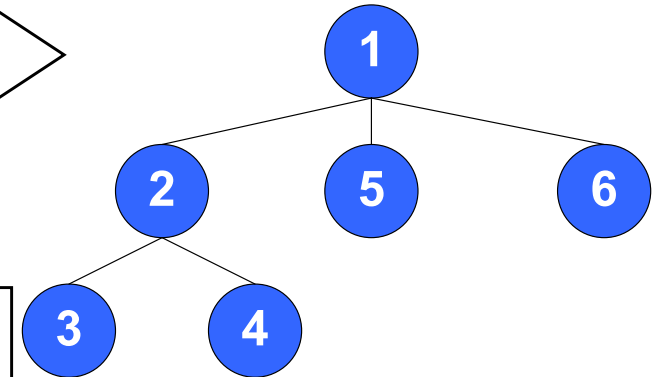
- Baummodell als Basis
- feste Dokumentreihenfolge (document order) = Reihenfolge der Start-Tags im Dokument
- Tiefensuche



```
<buch>
  <autoren>
    <name> Anna Baum</name>
    <name>Hans Gruber</name>
  </autoren>
  <titel>Ein Buch</titel>
  <preis>45</preis>
</buch>
```

Deserialisierung

Serialisierung



- **Deserialisierung** – Erzeugung eines Baums aus einem Dokument
- **Serialisierung** – Erzeugung eines Dokuments aus einem Baum

- Elemente werden einfach **über ihren Namen** identifiziert:

z.B. **order** oder **order/item**

- Attribute werden mit "**@name**" identifiziert:

z.B. **@id** oder **order/@id**

```
<?xml version="..." encoding="..."?>
<order id="O56">
  <item item-id="E16-2">
    <name>buch</name>
  </item>
</order>
```

- . aktueller Knoten
- .. Eltern-Knoten
- * beliebiges Kind-Element
- @* beliebiges Attribut
- // überspringt ≥ 0 Hierarchie-Ebenen nach unten
- [] spezifiziert ein Element
- | Auswahl (Vereinigung)
- Beispiel: * | @*
„Kind-Element oder Attribut des aktuellen Knotens“

Absolute und relative Pfade

- **absolute Pfade**

- beginnen mit /
z.B. /order/item

lesen: (➔) Folge dem Pfad von der Dokument-Wurzel zu einem Kind-Element order und von dort aus zu einem Kind-Elementen item!

- **relative Pfade**

- beginnen mit einem Element oder Attribut
z.B. order/item


lesen: (➤) item-Elemente, die Kind eines Elementes order sind

- Element order kann an beliebiger Stelle des XML-Dokumentes stehen

- XPath-Pfade werden in XSLT immer bzgl. eines bestimmten **Kontext-Knotens** ausgewertet:
Element-, Attribut- oder Text-Knoten

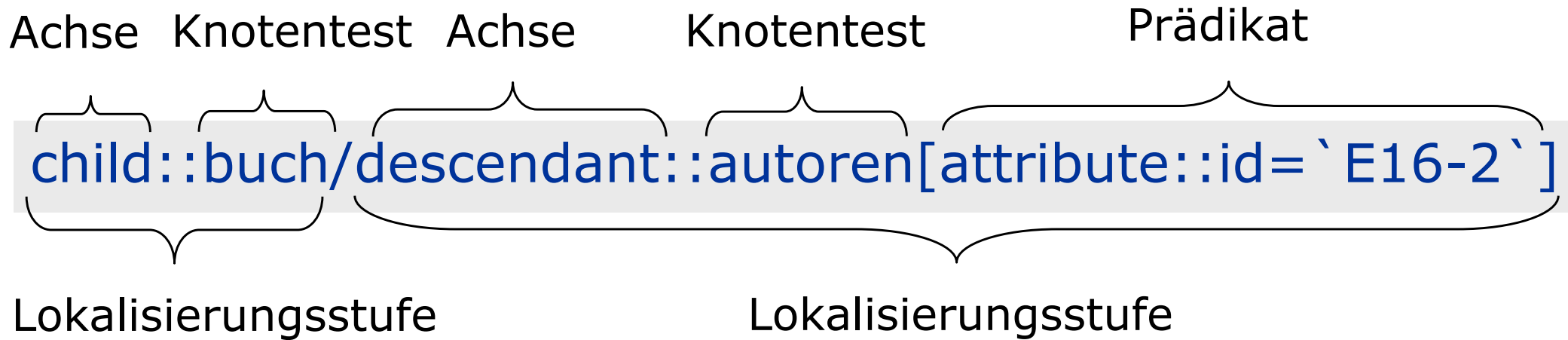
- Beispiel:

```
<xsl:template match="p">  
  <DIV>  
    <xsl:value-of select="."/>  
  </DIV>  
</xsl:template>
```

A diagram consisting of a rectangular box with a black border that encloses the text `select="."` within the `<xsl:value-of select="."/>` line of the XSLT code. A curved arrow originates from the bottom right corner of this box and points towards the text `aktueller Knoten "." ?` in the list below.

- Was bedeutet hier aktueller Knoten "." ?
- "." = Kontext-Knoten
- Kontext-Knoten = Knoten, auf den das Template angewandt wird (hier ein p-Element)

```
<?xml version="..." encoding="..."?>
<buch>
  <autoren item-id="E16-2">
    <name> Anna Baum</name>
    <name>Hans Gruber</name>
  </autoren>
  <titel>Ein Buch</titel>
  <preis>45</preis>
</buch>
```



Lokalisierungsstufe

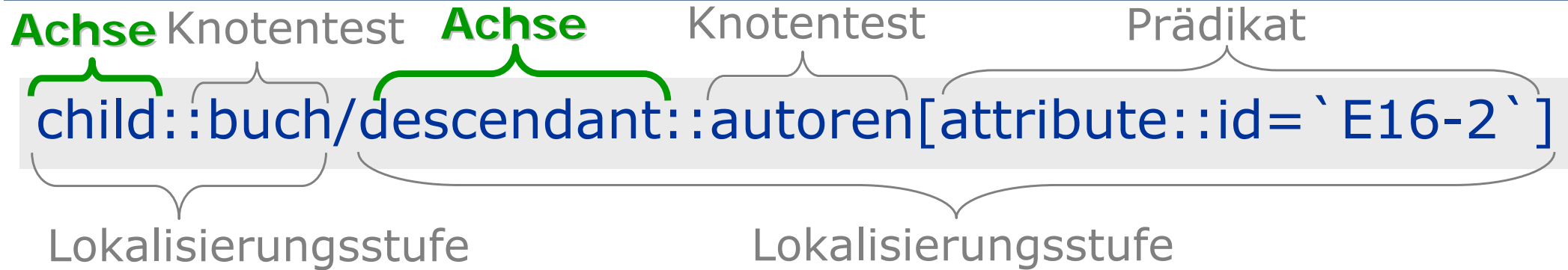


- besteht aus:
 - einem Achsenbezeichner
 - einem Knotentest
 - einem oder mehreren Prädikat (optional)
- ...in der Form:

`Achsenbezeichner::Knotentest[Prädikat1][Prädikat2]`

- :: Trennzeichen zwischen Achsenbezeichner und Knotentest

Achsen (I)



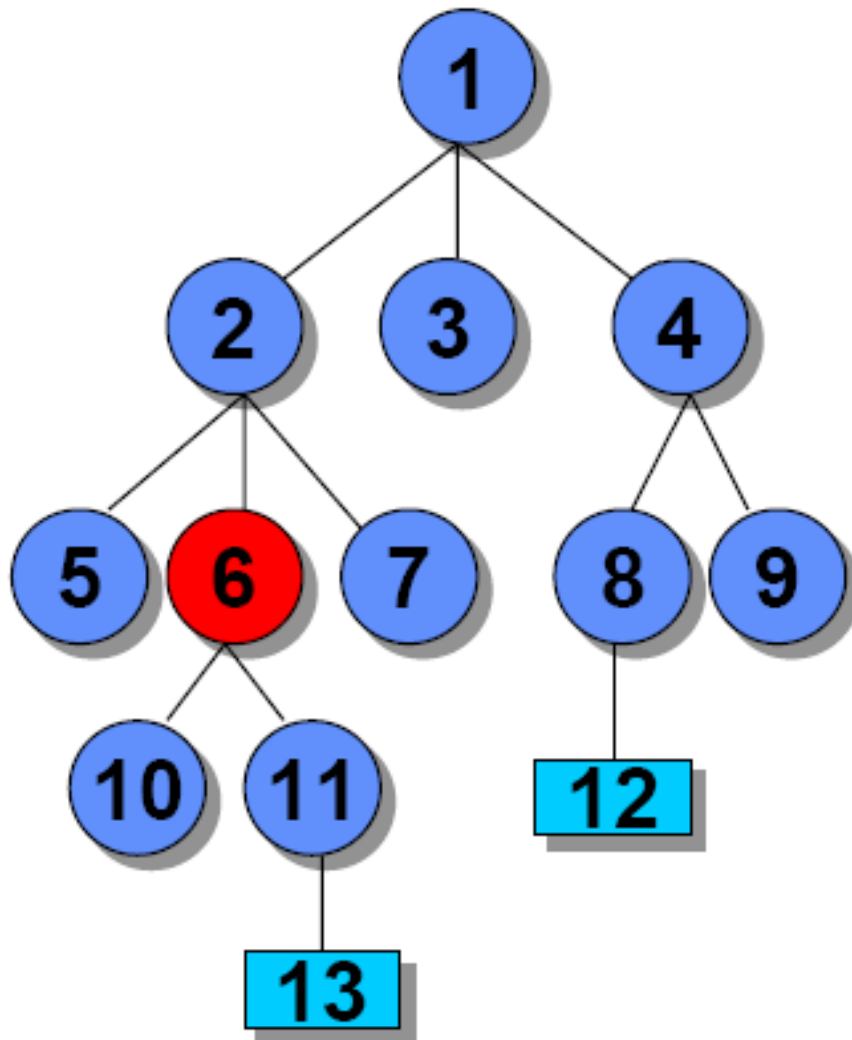
- `self::` Kontextknoten (.)
- `child::` Kinder des Kontextknoten
- `parent::` Eltern des Kontextknoten (..)
- `descendant::` Nachkommen des Kontextknoten (//)
- `descendant-or-self::` Kontextknoten & descendant-Achse
- `ancestor::` Vorfahren des Kontextknoten bis zum Wurzel
- `ancestor-or-self::` Kontextknoten & ancestor-Achse

Achsen (II)



- preceding-sibling:: vorhergehende Geschwisterknoten
- preceding:: Elementknoten vor dem Kontextknoten ohne vorfahren
- following-sibling:: alle folgende Geschwisterknoten
- following:: Elementknoten, die in der Dokumentenreihenfolge dem Kontextknotenfolgen
- attribute:: Attribute des Kontextknoten (@)
- namespace:: Namensraumknoten, falls vorhanden

Achsen – Beispiel



- self:: 6
- child:: 10, 11
- parent:: 2
- descendant:: 10, 11, 13
- descendant-or-self:: 6, 10, 11, 13
- ancestor:: 2, 1
- ancestor-or-self:: 6, 2, 1
- preceding-sibling:: 5
- preceding:: 5, 2, 1
- following-sibling:: 7
- following:: 10, 11, 13, 7, 3, 4, 8, 9, 12

Quelle: <http://swt.cs.tu-berlin.de/informatik2000/skripte/xml-datenbank.pdf>



- Filterung der Knotenmenge
- Filterungs-Kriterium:
 - Knotenname
z.B.: `child::buch`
 - Knotentyp
z.B.: `child::text()`
`child::node()`

Prädikate



- Verfeinerung der Filterung durch Prädikate
- Anzahl der Prädikate ≥ 0
- [] Bedingung
- Prädikatausdruck unterstützen
 - logische Operatoren: $<$, $>$, \leq , \geq , $=$, \neq
wobei $>$, $<$ müssen als Entity-Referenzen $>$ und $<$ benutzt werden
 - numerische Operatoren: $+$, $-$, $*$, div , mod

Prädikate – Randbedingungen für Pfade

- `order/item[@item-id = 'E16-2']`
 - item-Elemente, die Kind von order sind und Attribut item-id mit Wert 'E16-2' haben

```
<order id="4711">  
  <item item-id="E16-2">  
    <name>buch</name>  
  </item>  
</order>
```

- Randbedingungen können an beliebiger Stelle in einem Pfad vorkommen:

- `order[@order-id = '4711']/item`

```
<orders>  
  <order id="4711">  
    <item item-id="E16-2">  
      <name>buch</name>  
    </item>  
  </order>  
  <order id="4711">  
    <item item-id="E16-3"/>  
  </order>  
</orders>
```



XPath Funktionen



Grundlegende Datentypen

- node-set – liefert eine ungeordnete Knotenmenge
- string – ist eine Zeichenfolge
- number – Fließkommazahl
- boolean – Werte true und false

Knotenmenge-Funktionen

- Funktionen:

- *number* last() – eine Zahl, die die Größe der aktuellen Knotenmenge entspricht
- *number* position() – Position eines Knotens
- *number* count(node-set) – Anzahl der Knoten in der Knotenmenge

- Beispiele:

order/item[position() = 1]

order/item[position()=last()]

- **Funktionen:**

- *string* `string(object)` – interpretiert ein übergebenes Argument als Zeichenkette und gibt die ermittelte Zeichenkette zurück
- *string* `string-length(string)` – Länge von String (Anzahl der Zeichen)
- *boolean* `starts-with(string, string)` – true wenn die erste Zeichenkette mit der zweiten Zeichenkette anfängt

- Funktionen:

- *boolean* `boolean(object)` – nimmt ein Objekt und liefert booleschen Wert zurück
- *boolean* `not(object)` – nimmt booleschen Ausdruck und liefert `true` wenn Argument `false` ist
- *boolean* `true()` – immer `true`
- *boolean* `false()` – immer `false`

- Beispiel:

`order/item[not(position)=last()]`

- Funktionen:

- *number* number(object) – versucht eine Zeichenkette als Zahl zu interpretieren und gibt die ermittelte Zahl zurück
- *number* sum(node-set) – Gesamtsumme der Zahlenwerte des Ausgangsknotens (nach Umwandlung des String-Werte)
- *number* round(number) – rundet den Wert zur nächsten Ganzzahl

- Beispiel:

`number(3xy) → 3`

Wähle das
Wurzelement AAA
aus:

```
<AAA>
  <BBB/>
  <CCC/>
  <BBB/>
  <BBB/>
  <DDD>
    <BBB/>
  </DDD>
  <CCC/>
</AAA>
```

/AAA

Wähle alle CCC Elemente
aus, die Kinder des
Elements AAA sind:

```
<AAA>
  <BBB/>
  <CCC/>
  <BBB/>
  <BBB/>
  <DDD>
    <BBB/>
  </DDD>
  <CCC/>
</AAA>
```

/AAA/CCC

//BBB

```
<AAA>
  <BBB/>
  <CCC/>
  <BBB/>
  <DDD>
    <BBB/>
  </DDD>
  <CCC>
    <DDD>
      <BBB/>
      <BBB/>
    </DDD>
  </CCC>
</AAA>
```

//DDD/BBB

```
<AAA>
  <BBB/>
  <CCC/>
  <BBB/>
  <DDD>
    <BBB/>
  </DDD>
  <CCC>
    <DDD>
      <BBB/>
      <BBB/>
    </DDD>
  </CCC>
</AAA>
```

Beispiele

/*/*/*/BBB****

```

<AAA>
  <XXX>
    <DDD>
      <BBB/>
      <FFF/>
    </DDD>
  </XXX>
  <CCC>
    <BBB>
      <BBB>
        <BBB/>
      </BBB>
    </BBB>
  </CCC>
</AAA>
    
```

//*

```

<AAA>
  <XXX>
    <DDD>
      <BBB/>
      <FFF/>
    </DDD>
  </XXX>
  <CCC>
    <BBB>
      <BBB>
        <BBB/>
      </BBB>
    </BBB>
  </CCC>
</AAA>
    
```

/AAA/BBB[last()]

```
<AAA>
  <BBB/>
  <BBB/>
  <BBB/>
  <BBB/>
</AAA>
```

//@id

```
<AAA>
  <BBB id="b1"/>
  <BBB id="b2"/>
  <BBB
    name="bbb"/>
  <BBB/>
</AAA>
```

Beispiele

//CCC | //BBB

```
<AAA>
  <BBB/>
  <CCC/>
  <DDD>
    <CCC/>
  </DDD>
  <EEE/>
</AAA>
```

//CCC/following-sibling::*

```
<AAA>
  <BBB>
    <CCC/>
    <DDD/>
  </BBB>
  <XXX>
    <EEE/>
    <CCC/>
    <FFF/>
    <FFF>
    <GGG/>
  </FFF>
</XXX>
</AAA>
```

⇒ <http://www.futurelab.ch/xmlkurs/xpath.de.html>



XPath 2.0

XPath 2.0

- Januar 2007 – W3C Recommendation für neue Version von XPath
- zeitgleich mit XQuery 1.0 & XSLT 2.0
- rückwärts kompatibel zu XPath 1.0

Was hat sich geändert? Was ist neu?

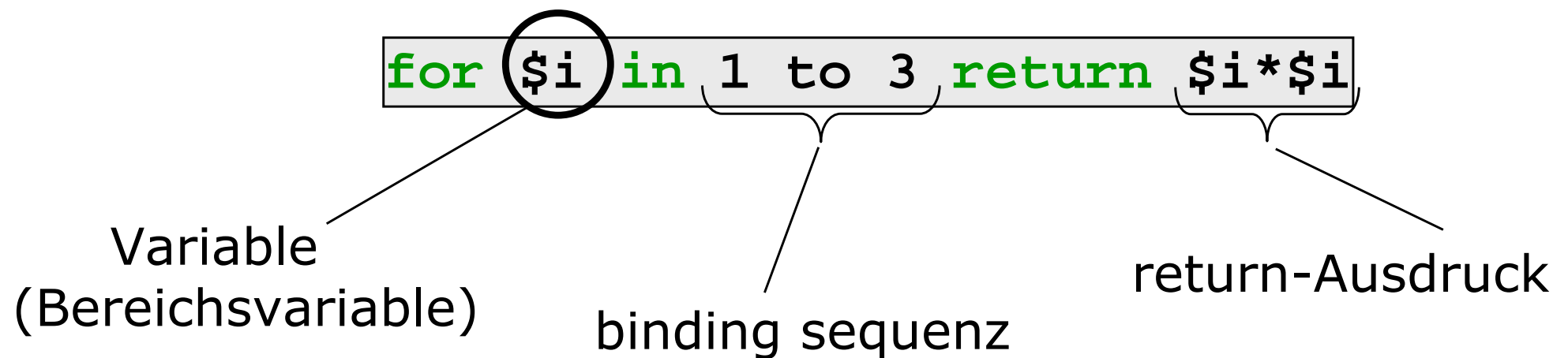
- erweitertes Datenmodell
- neue Konstrukte für Ausdrücke
- neue Datentypen
- neue Operatoren
- erweiterte Funktionsbibliothek

Erweitertes Datenmodell

- Berücksichtigung einzelner Werte (**atomic values**)
- Daten unterschiedlichen Typs:
 - Zeichenfolgen, Zahlen, logische Werte, Datums- und Zeitwerte
 - qualifizierte Namen & URIs
 - einfache Sequenzen & Listen
- Ergebnis eines XPath Ausdrucks: Auswahl von Knoten, Einzelwert oder Sequenz
- XPath 2.0 auf Knotenbaum
 - nur Daten auslesen
- XPath 2.0 auf Einzelwerten & Sequenzen
 - neue Werte/Sequenzen erzeugen

Neue Konstrukte für Ausdrücke

- für Operationen mit Sequenzen Verwendung des *for*-Ausdrucks
- Beispiel



Ergebnis: 1, 4, 9

Weitere Ausdrücke

- bedingte Ausdruck *if*

```
if @menge > 1000 then "gut" else "weniger gut"
```

- quantifizierende Ausdrücke *some* und *every*

```
some $a in $lager/artikel satisfies $lager/artikel/menge=0
```

- wahr, wenn die Menge mind. bei einem Artikel = 0

```
every $a in $lager/artikel satisfies $lager/artikel/menge>0
```

- wahr, wenn von allem Artikel mind. einer vorhanden ist.

Neue Datentypen

- Unterstützung der XML-Schema Datentypen
- XPath 1.0
 - number – Fließkommazahl
- XPath 2.0
 - integer
 - decimals
 - single precision
 - Datums-, Zeit- und Dauerwerte

- **Knotenvergleiche**

- *is* – prüft, ob zwei Ausdruck den selben Knoten liefern
- *<<, >>* – prüfen, welcher von zwei Knoten in der Dokumentreihenfolge früher oder später erscheint

- **Kombination von Knotensequenzen**

- *union* – Vereinigung zwei Knotensequenzen zu einer Sequenz
- *intersect* – erzeugt aus zwei Sequenzen eine Sequenz, die Knoten enthält, die in beiden vorkommen
- *except* – erzeugt aus zwei Sequenzen eine Sequenz, die Knoten enthält, die in der ersten Sequenz aber nicht in der zweiten vorkommen

- Behandlung von Zeichenketten
- Verwendung regulärer Ausdrücke
- Funktionen für Arbeit mit Sequenzen
- Funktionen für Datums- und Zeitwerte



XML Linking Language (XLink)

- Entwicklung beeinflusst von vieler etablierter Hypermedia-Systeme und –Standards:
 - **HTML** definiert mehrere Elementtypen, die Links repräsentieren
 - **HyTime** definiert Strukturen u.a. für eingebettete, eingehende und Third-Party-Links
 - **Text Encoding Initiative Guidelines** bietet Strukturen zum Erzeugen von Links, aggregierten Objekten und Linksammlungen.

XLink – Ziele & Eigenschaften

- Ziel:
 - innerhalb von XML-Dokumenten Links zu erzeugen
 - Vokabular für Beschreibung von Links
- Eigenschaften
 - definiert, wie Dokumente miteinander verknüpft werden
 - erlaubt Links zwischen Ressourcen zu erzeugen & zu beschreiben
 - nutzt XML-Syntax
 - bietet ein minimales Link-Verhaltensmodell
 - bietet Link-Datenstrukturen
- Rahmen für die Erzeugung
 - grundlegender unidirektionaler Links
 - komplexerer Link-Strukturen

XLink – Features

- Juni 2001 W3C Recommendation
- Anforderungen:
 - Links, die in beide Richtungen begangen werden können
 - Links, die mehrere Ziele ansteuern können
 - Links, die mit Metadaten versehen sind
 - Links, die unabhängig von den Dokumenten gespeichert sind

XLink – Verwendung

- keine eigenen Elementen → Benutzung vorhandener Elemente
- Eigenschaften eines XLinks-Element über Attribute
- Globale Attribute für die Definition von XLink-Elementtypen
- XLink Namensraum → <http://www.w3.org/1999/xlink/>
 - Elementen werden XLink-Eigenschaften mitgeben

- **einfache Links**

- einfache Verallgemeinerung von HTML-Links
- können einem Link und seinem Ziel einen Namen geben (title und role)
- können das Verhalten eines Links spezifizieren (actuate und show)
- werden bisher nur von wenigen Browsern unterstützt

- **erweiterte Links**

- verbinden mehr als zwei Ressourcen
- beschreiben eine Menge von Ressourcen zusammen mit ihrer Link-Struktur (Netzwerk)
- werden von keinem Browser unterstützt

Einfaches Link – Beispiel



- XML Element verwendbar als XLink-Element wenn es mind. ein Attribut `xlink:type` mit einem gültigen Wert enthält
- Problem: nur weniger/eingeschränkte Implementierungen von XLink

XLink – Attribute

- **xlink:href** gibt an, wo eine Ressource zu finden ist (URI)
- **xlink:actuate** gibt an, wann die Ressource geladen wird
 - "onLoad" gleichzeitig mit dem Dokument
 - "onRequest" nur nach Aufforderung
- **xlink:show** Gibt an, wo die Ressource dargestellt wird
 - "new": in einem neuen Fenster
 - "replace": im aktuellen Fenster und dieses ersetzen
 - "embeded": ins aktuelle Fenster integrieren

- beschreiben eine Menge von Ressourcen zusammen mit ihrer Link-Struktur (Netzwerk)
 - Ressourcen existieren unabhängig von Links
 - extra Linking-Elemente definieren Beziehungen
 - erlaubt many-to-many-Beziehungen
- werden von keinem Browser unterstützt
- stoßen bisher auf wenig Akzeptanz
- mögliche Alternative:
 - RDF (*Resource Description Framework*)



XML Base (XBase)

- in HTML 4.01 `<base>` für Basisadresse für URIs
- XBase
 - Spezifikation für XML-Attribut `xml:base`
 - Bestimmung einer anderen Basisadresse als die, die ein XML Dokument selbst verwendet
 - Präfix `xml` gebunden an den Namensraum (XML 1.1):
<http://www.w3.org/XML/1998/namespace>
- Juni 2001 W3C Recommendation

XBase mit XLink – Beispiel

```
<?xml version="1.0" encoding="..."?>
<image xml:base="http://www.mybase.de/mylinking"
xmlns:xlink="http://www.w3.org/1999/xlink"
xlink:type="simple"
xlink:href="bild.gif"
xlink:actuate="onRequest"
xlink:show="new">Zeige das Bild
</image>
```

xml:base

+

relativer URI

=

vollständiger URI

<http://www.mybase.de/mylinking/bild.gif>

XLink & XBase

- XLink stellt flexibles Linking für XML bereit
- XML Base liefert einen bewährten Ansatz zur Nutzung von Base URI für XML
- XLink, XML Base bringt Tiefe und Leistungsfähigkeit in die XML-Familie

Quelle: <http://www.w3c.de/Press/link-base-pressrelease.html>



XML Pointer Language (XPointer)

XPointer

- 1997 – Grundlagen für XPointer in Zusammenhang mit XLink
- März 2003 – W3C Recommendation
 - grundlegendes Framework
 - Schemas, die dem Framework entsprechen
- basiert auf & erweitert XPath in einigen Bereichen
- Ziel
 - Zeigen auf bestimmte Elemente innerhalb eines XML Dokumentes (engl. to point ~ zeigen)

XPointer – Eigenschaften

- XPointer nur in wohlgeformten XML-Dokumenten möglich
- XPointer ermöglicht
 - Punkte, Bereiche und Knoten anzusprechen
 - Informationen durch String-Vergleiche zu lokalisieren
 - Adressen in URI-Referenzen als Identifikationspunkte zu nutzen
 - Adressierung ohne anchor-Markup in Zieldokument
 - Adressierung von Knoten (Elemente, Attribute, Text)
- Extension von XPath: zusätzlich auch Adressierung von Dokumentteilen, die keine Knoten sind
 - Adressierung von Punkten (points) und Bereichen (range)

XPointer – Aufbau

- XPointer Framework

- <http://www.w3.org/TR/xptr-framework/>

- element() Scheme

- <http://www.w3.org/TR/xptr-element/>

- xmlns() Scheme

- <http://www.w3.org/TR/xptr-xmlns/>

W3C Recommendations

- xpointer() Scheme

- <http://www.w3.org/TR/xptr-xpointer/>

W3C Working Draft

Zeigertyp xpointer()

- entspricht einem XPath-Ausdruck
- Funktionen und Schreibweisen von XPointer möglich
- Beispiele

```
meinequelle.xml#xpointer(id("a1"))
```

→ wählt aus dem Dokument `meinequelle.xml` Element mit ID-Attribut `id="a1"`

(Beim Attribut `id`

```
meinequelle.xml#a1
```

→ einfache Schreibweise)

Zeigertyp element()

- zeigt auf ein einzelnes Element innerhalb des Dokuments durch
 - Element-ID
 - Indexsequenz – besteht aus Indexnummern der einzelnen Elemente
- Beispiel
 - im Dokument zeige auf Element "D"

```
<A id="12345">
  <B></B>
  <C>
    <D/>
  </C>
</A>
```

#element(12345/2/1)

#element(/1/2/1)

Pfad vom root-Element aus: erstes Kind des zweiten Kindes des ersten Kindes

Verknüpfung xmlns()

- um eine XML-Ressource an einen Namespace zu binden
- Beispiel

```
<A xmlns="http://elementA">  
  <B xmlns="http://elementB">  
    </B>  
  </A>
```

Dokument

```
xmlns(NameA=http://elementA) xmlns(NameB=http://elementA)  
xpointer(/NameA:A/NameB:B)
```

XPointer

→ Zugriff auf Elemente über die Namensräume hinweg

Erweiterungen zu XPath

- Punkt (point)

`meinequelle.xml#xpointer(/A/[0])`

Position vor dem ersten Kinderknoten von A

`meinequelle.xml#xpointer(/A/[3])`

Position nach dem dritten Kinderknoten von A

- Bereich (range)

`meinequelle.xml#xpointer(/A/[3] range-to /A/[5])`

Startpunkt
Endpunkt

Bereich zwischen dem dritten und fünften Zeichen



XML Query

Was ist XQuery?

XQuery ...

- ist **die Abfragesprache** für XML-Daten
 - XML-Dateien &
 - alles was in XML darstellbar ist (auch DBs)
- ist für XML das, was SQL für Datenbanken
- basiert auf **XPath-Ausdrücken**
- wird bei fast allen DB-Engines unterstützt (IBM, Oracle, Microsoft, etc.)
- ist sein Januar 2007 eine **W3C Recommendation**
→ <http://www.w3.org/TR/xquery/>

- Informationen zu extrahieren, um sie in Web Services zu nutzen
- Reports zu generieren
- XML ins XHTML zu transformieren
- in Web-Dokumenten nach relevanten Informationen zu suchen

Schlüsselwörter & Variablen

- **for** – zur (Schleifen-)Verarbeitung von einzelnen Elementen innerhalb eines XML-Dokuments
- **let** – zur Erstellung von Variablen und Zuweisung von Werten
- **where** – konditionale Anweisung, zusammen mit dem Schlüsselwort **for** verwendet
- **return** – zur Rückgabe von Werten an die aufrufende Instanz des Ausdrucks
- **order** – sortier die Ausgabe
- **if-then-else** – bedingte Abfragen

- **\$** - kennzeichnet eine Variable
- **doc("file.xml")** – öffnet das spezifizierte XML-Dokument (hier: file.xml)

for – Beispiel

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<bookstore>
  <book category="COOKING">
    <title lang="en">Everyday Italian</title>
    <author>Giada De Laurentiis</author>
  </book>
  <book category="WEB">
    <title lang="en">Learning XML</title>
    <author>Erik T. Ray</author>
  </book>
</bookstore>
```

Dokument

```
<title lang="en">Everyday Italian</title>
<title lang="en">Learning XML</title>
```

**XQuery-
Anfrage an
das
Dokument**

**Ergebnis der
Anfrage**

```
for $x in doc("books.xml")/bookstore/book/title
order by $x
return $x
```

for – Beispiel mit HTML

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<bookstore>
  <book category="COOKING">
    <title lang="en">Everyday Italian</title>
    <author>Giada De Laurentiis</author>
  </book>
  <book category="WEB">
    <title lang="en">Learning XML</title>
    <author>Erik T. Ray</author>
  </book>
</bookstore>
```

Dokument

XQuery-
Anfrage an
das
Dokument

```
<ul>
{
for $x in doc("books.xml")/bookstore/book/title
order by $x
return <li>{data($x)}</li>
}
</ul>
```

```
<ul>
<li>Everyday Italian</li>
<li>Learning XML</li>
</ul>
```

Ergebnis der
Anfrage

Quelle: http://www.w3schools.com/xquery/xquery_flwor_html.asp

if-then-else – Beispiel

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<bookstore>
  <book category="COOKING">
    <title lang="en">Everyday Italian</title>
    <author>Giada De Laurentiis</author>
  </book>
  <book category="WEB">
    <title lang="en">Learning XML</title>
    <author>Erik T. Ray</author>
  </book>
</bookstore>
```

Dokument

```
<various>Everyday Italian</various>
<web>Learning XML</web>
```

XQuery-
Anfrage an
das
Dokument

Ergebnis der
Anfrage

```
for $x in doc("books.xml")/bookstore/book return
if ($x/@category="WEB")
  then <web>{data($x/title)}</web>
  else <various>{data($x/title)}</various>
```

Quelle: http://www.w3schools.com/xquery/xquery_flwor_html.asp

Wie geht es weiter?

heutige Vorlesung

- ☑ XPath 1.0 & 2.0
- ☑ XLink
- ☑ XBase
- ☑ XPointer
- ☑ XQuery

Vorlesung nächste Woche

- XLST

Arten von XLinks

- ***traversal*** – Übergang von einer Startressource zu einer Endressource
- ***arcs*** – Pfad, der beim Folgen eines Link zu gehen ist
- ***outbound link*** – wenn arc von einer lokalen zu einer entfernten Ressource zeigt
- ***inbound link*** – wenn arc von einer entfernten auf eine lokale Ressource verweist
- ***third-party-links*** – arcs, die von einer entfernten zu einer anderen entfernten Ressource zeigen