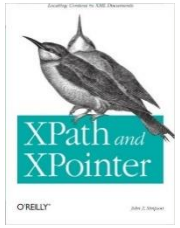
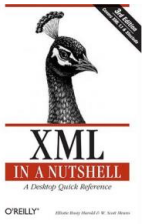
Modul 4

# XPath

Josef Altmann

//*






**Der vorliegende Foliensatz basiert vorwiegend auf:**

*Kay, M.: XPath 2.0 Programmer's Reference (3rd ed.), Wiley, 2004.*
*Lehner, W., Schöning, H.: XQuery, dpunkt.verlag, 2004.*
*Simpson, J.: XPath and XPointer, O'Reilly, 2002.*
*Becher, M.: XML (1. Auflage), Springer Campus, 2009.*
*Harold, R., Means, S.: XML in a Nutshell: A Desktop Quick Reference (3rd ed.), O'Reilly Media, 2009.*

# Inhalt

- **Einführung**
- Datenmodell
- Pfadausdrücke
- Erweiterte Ausdrücke
- Funktionen und Operatoren
- Zusammenfassung

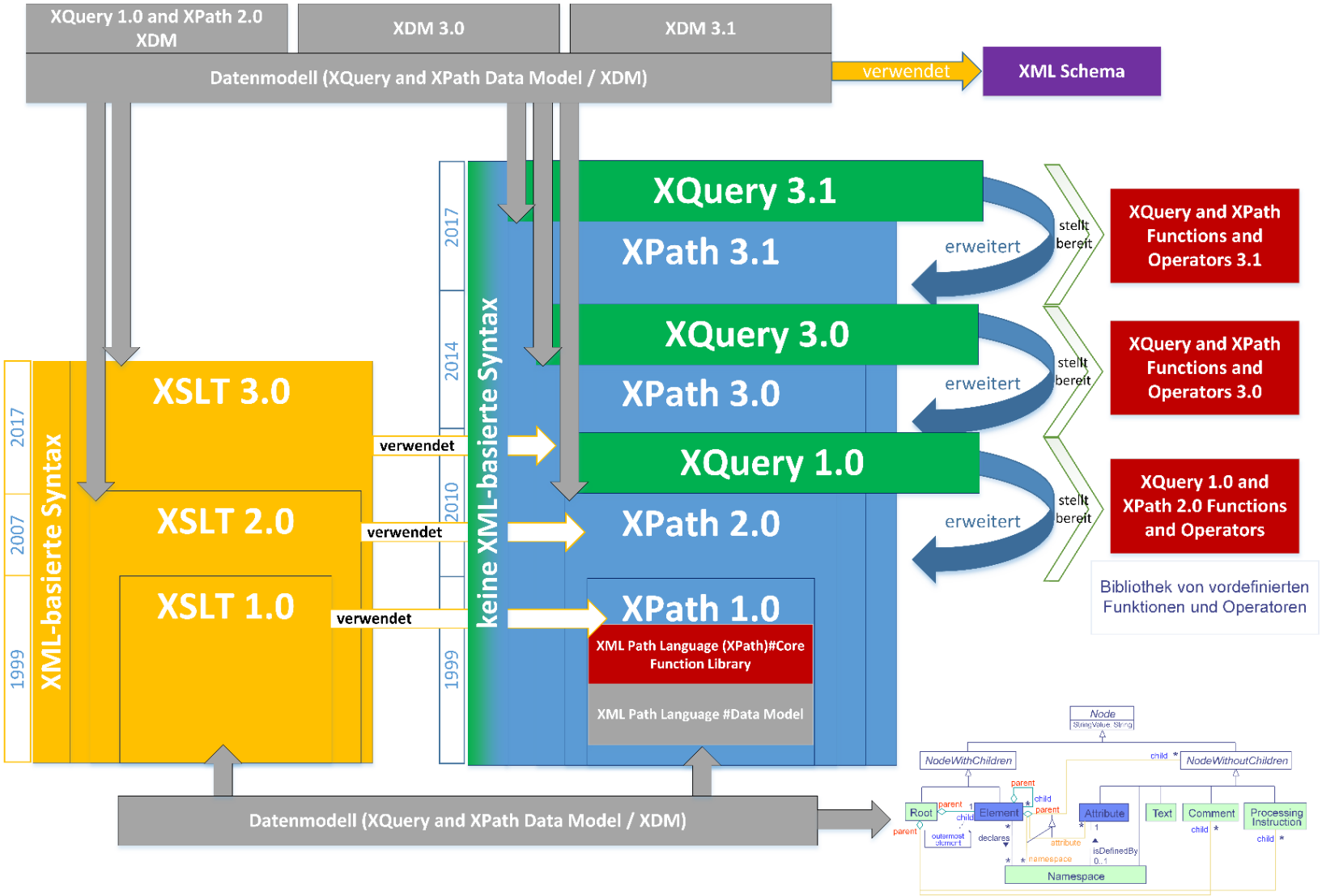
XPath-Versionen (W3C-Standards):

- XPath 1.0, Nov. 1999, ~28 PDF-Seiten
- **XPath 2.0**, Jan. 2007, ~78 PDF-Seiten
- XPath 3.0, Apr. 2014, ~100 PDF-Seiten
- XPath 3.1, März 2017, ~171 PDF-Seiten

# XPath
## XML Path Language

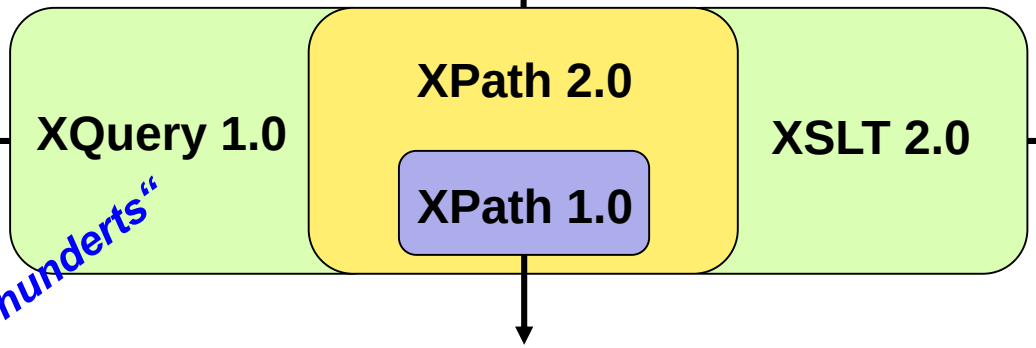
- Einführung
  - Selektion (Lokalisieren) von Knoten und Inhalten (Dokumentteilen)
  - in vielen XML-Standards verwendet: XQuery, XSLT, XML Schema, etc.
  - keine XML-Syntax - eigener (Pfadbeschreibungs-) Standard
  - Selektionskriterien: Element- und Attributnamen, Inhalt, Typ, etc.
- Grundprinzip der Verarbeitung
  - Navigation in einer Baumstruktur, ähnlich zur Navigation in einem Dateisystem
  - Ausgangspunkt ist ein **Kontext** in Form eines Baumknotens
  - Kontext wird von einem XPath-Ausdruck vorgegeben
  - **Navigation** und **Filter** modifizieren den Kontext
  - Ergebnis eines XPath-Ausdrucks = **zuletzt berechneter Kontext** (= Knotensequenz)
  - Erzeugen und Ändern von Knoten wird nicht unterstützt *(read only!)*
- W3C-Recommendations:
  - XPath 1.0, Nov. 1999
  - **XPath 2.0**, Jan. 2007
  - XPath 3.0, April 2014
  - XPath 3.1, Jänner 2017

# XPath
## Versionen



# XPath
## Zusammenhang bis XPath 2.0

- Bedingte Ausdrücke
- Arithmetische Ausdrücke
- Quantifizierende Ausdrücke
- Viele eingebaute Funktionen (> 100)
- Unterstützung von XML-Schema-Datentypen
- Verwendung mehrerer Dokumente
- Datenmodell: Knoten*sequenz*




- FLWOR-Ausdrücke
- Restrukturierung
- Element-Konstruktoren
- XQuery-Prolog
- Benutzerdef. Funktionen

- Stylesheets
- Templates
- Literale Ergebnis-elemente
- Benutzerdef. Funktionen

„SQL des 21. Jahrhunderts“

- Pfadausdrücke (Extraktion & Reduktion)
- Knotentests und Prädikate (Selektion)
- Vergleichsausdrücke
- Einige eingebaute Funktionen (27)
- Datenmodell: Knoten*menge*

# XPath

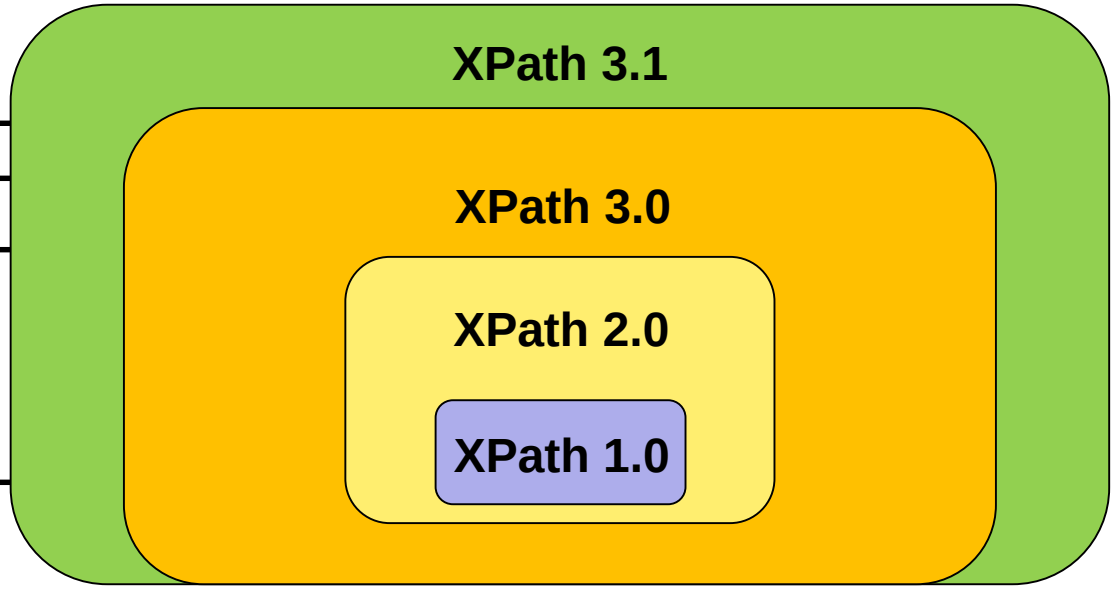
## Sprachumfang (Erweiterungen in XPath 3.0 / 3.1)

### XPath 3.1

- Neue Datentypen map und array

### XPath 3.0

- Anonyme Funktionen
- Dynamische Funktions-Aufrufe
- Vereinigungs-Typen (Union)
- Namespace-Literale
- String-Konkatenations-Operator
- Mapping-Operator




➔ Mehr zu XPath 3.0/3.1 in eigenem Foliensatz

# XPath

## Sprachumfang (Erweiterungen in XPath 3.0 / 3.1)

### XPath 3.1

- Neue Datentypen map und array

### XPath 3.0

- Anonyme Funktionen
- Dynamische Funktions-Aufrufe
- Vereinigungs-Typen (Union)
- Namespace-Literale
- String-Konkatenations-Operator
- Mapping-Operator

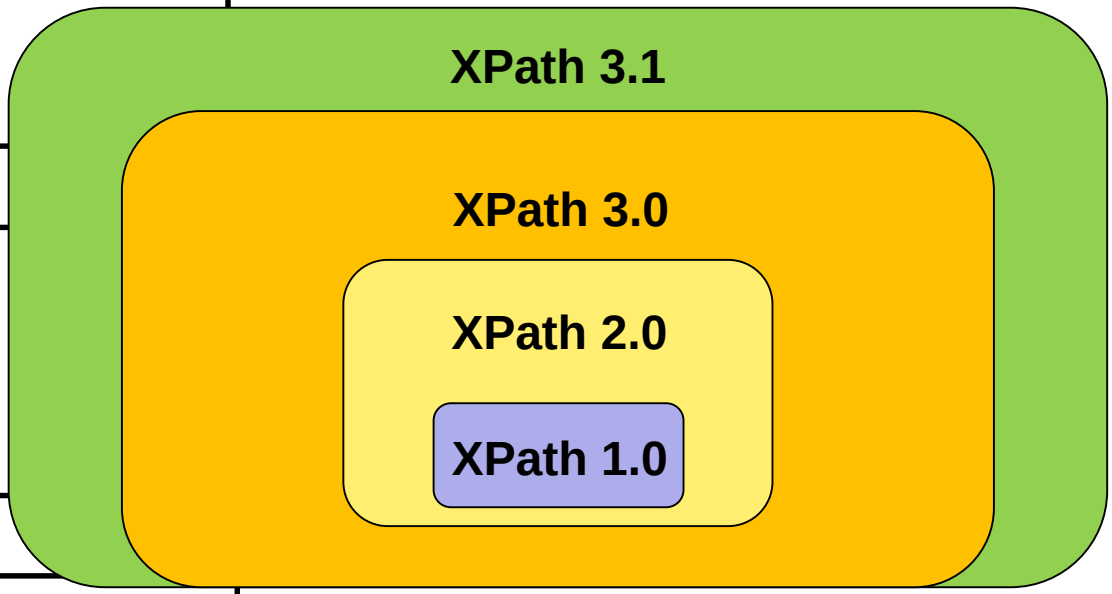
### XPath 2.0

- Bedingte Ausdrücke
- Arithmetische Ausdrücke
- Quantifizierende Ausdrücke
- Viele eingebaute Funktionen (> 100)
- Unterstützung von XML-Schema-Datentypen
- Verwendung mehrerer Dokumente
- Datenmodell: Knoten*sequenz*

### XPath 1.0

- Pfadausdrücke (Extraktion & Reduktion)
- Knotentests und Prädikate (Selektion)
- Vergleichsausdrücke
- Einige eingebaute Funktionen (27)
- Datenmodell: Knoten*menge*

➔ Mehr zu XPath 3.0/3.1 in eigenem Foliensatz

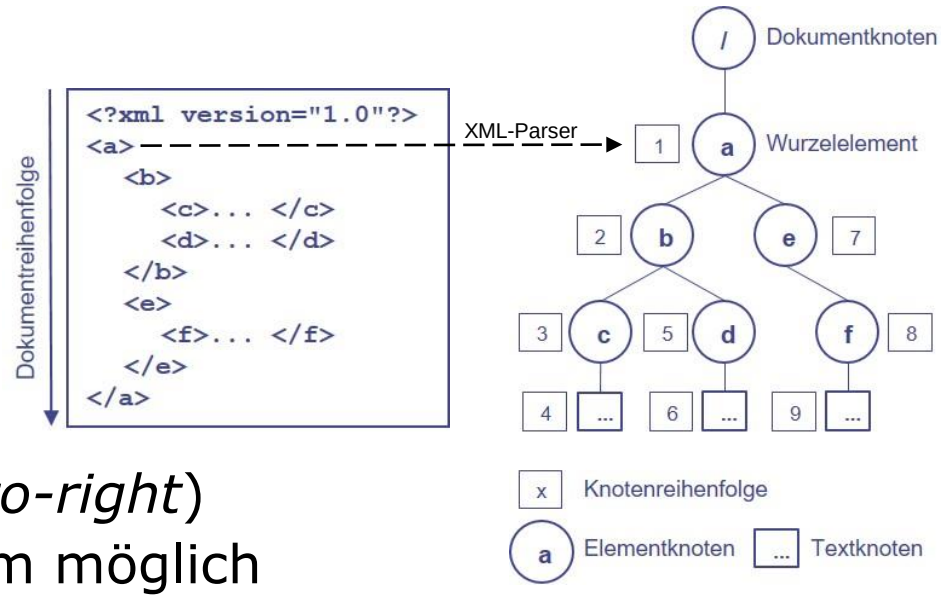


# Inhalt

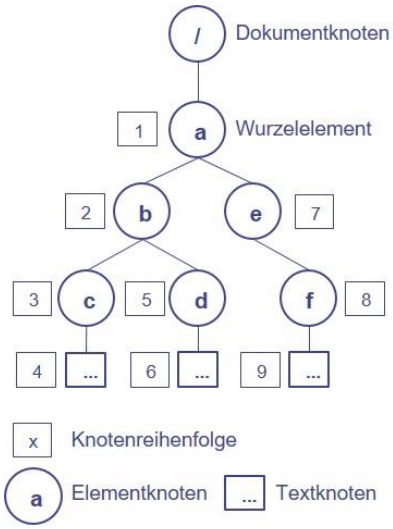
- Einführung
- **Datenmodell**
  - Knoten
  - atomarer Wert
  - Sequenz
- Pfadausdruck
- Erweiterte Ausdrücke
- Funktionen und Operatoren
- Zusammenfassung

# XPath
## Datenmodell

- XPath basiert auf dem **XQuery and XPath Data Model (XDM)**
  - siehe www.w3.org/TR/xpath-datamodel/
- XDM ist das Datenmodell für XPath, XQuery und XSLT
- Basiskomponenten von XDM sind:
  - Knoten
  - atomare Werte
  - Sequenzen
  - (Funktionen)
- XML-Dokument wird als Baum dargestellt
- Knoten im Baum sind geordnet (*top-down*, *left-to-right*)
- Somit: Navigation im Baum möglich



# XPath
## Datenmodell – Knoten

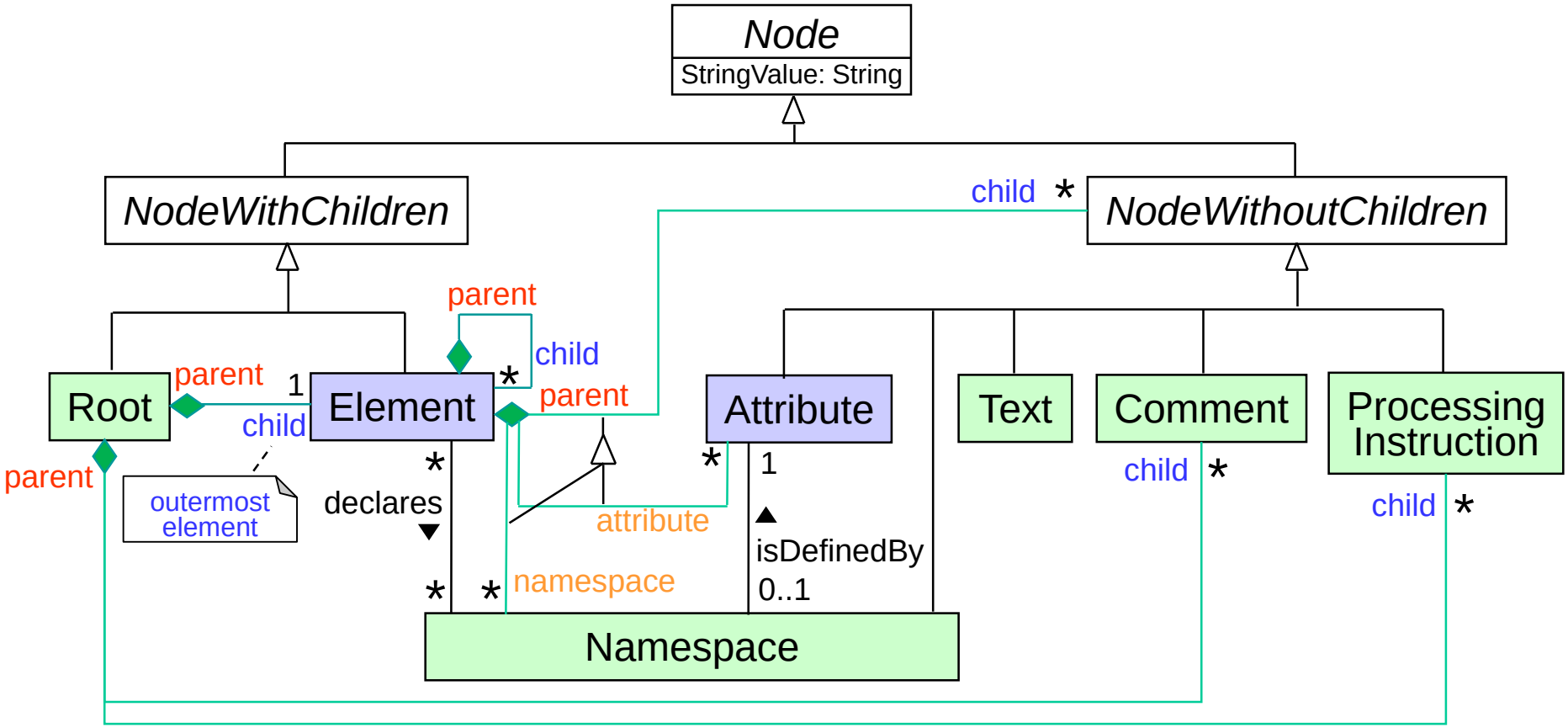



- **7 Knotenarten** im Dokumentenbaum
  - **Dokumentknoten** bzw. Wurzelknoten
    - Das gesamte XML-Dokument
  - **Elementknoten**
    - für jedes Element im XML-Dokument
  - **Attributknoten**
    - zu entsprechenden Elementknoten zugeordnet
  - **Namensraumknoten**
    - für alle NS-Präfixe und einen etwaigen Default-NS; zu entsprechenden Elementknoten zugeordnet
  - **Verarbeitungsanweisungsknoten**
  - **Kommentarknoten**
    - enthalten die in `<!-- ... -->` enthaltene Zeichenkette
  - **Textknoten**
    - enthalten ausschließlich Zeichendaten

# XPath

## Knotentypen

UML Klassendiagramm




Hinweis: Root ist nicht die Elementwurzel, sondern repräsentiert das gesamte XML-Dokument ("Dokumentwurzel")

# XPath
## Datenmodell – Knoteninformationen

- Knoten liefern folgende Informationen
  - **Knotenname**: qualifizierter Name bei Element- und Attributknoten, Präfix bei Namensknoten
  - **Elternknoten**: jeder Knoten außer dem Dokumentknoten hat genau einen Elternknoten
  - **Kindknoten**: nur bei Dokumentknoten und Elementknoten
  - **Attribute**: nur bei Elementknoten
  - **Namensräume**: nur bei Elementknoten, die Namensraumknoten enthalten die aktuelle Präfix-Bindungen
  - **Typ**: Typinformationen zum Knoten
  - **String-Wert**: siehe nächste Folie

# XPath
## Datenmodell – String-Wert eines Knotens

| *Node* |
| --- |
| `StringValue: String` |

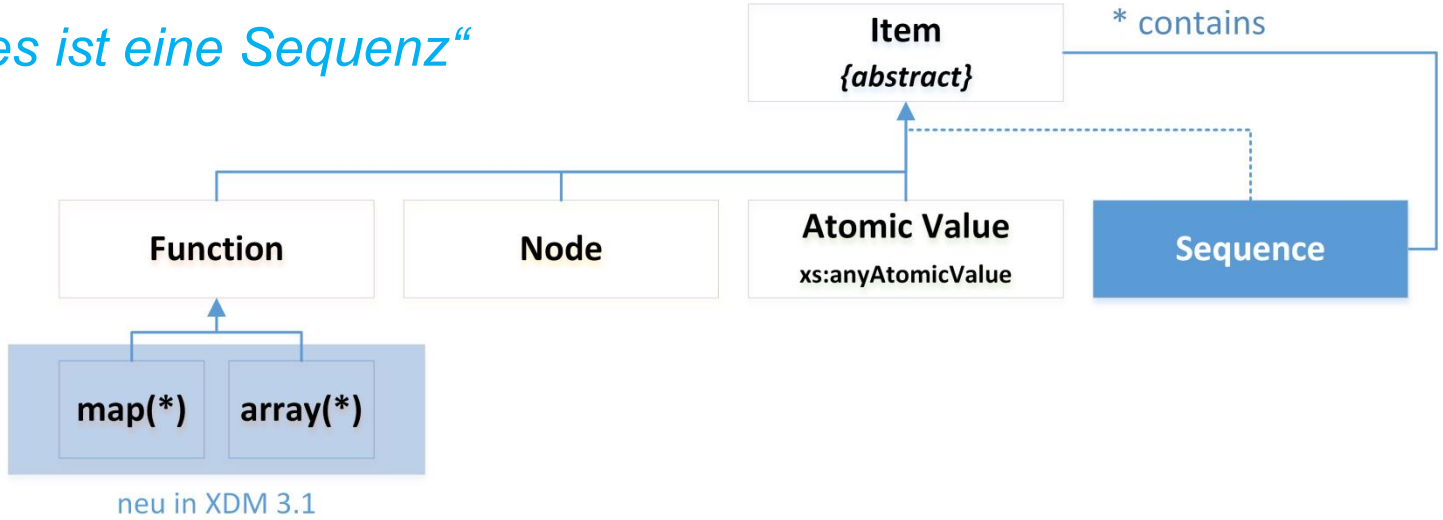
- String-Wert (**`StringValue`**) liefert bei
  - **Dokumentknoten**: Konkatenation aller Textknotennachfolger im Dokument
  - **Elementknoten**: Konkatenation aller Textknoten unterhalb eines Elementknotens
  - **Attributknoten**: normalisierter Attributwert
  - **Namensraumknoten**: Namensraum-URI
  - **Kommentarknoten**: Inhalt des Kommentars, ohne `<!-- ... -->`
  - **Textknoten**: Zeicheninhalt

String-Wert für Elementknoten `CourseCatalog`

| `CourseCatalog:Element` |
| --- |
| `StringValue := '...'` |

# XPath
## Datenmodell – Sequenz

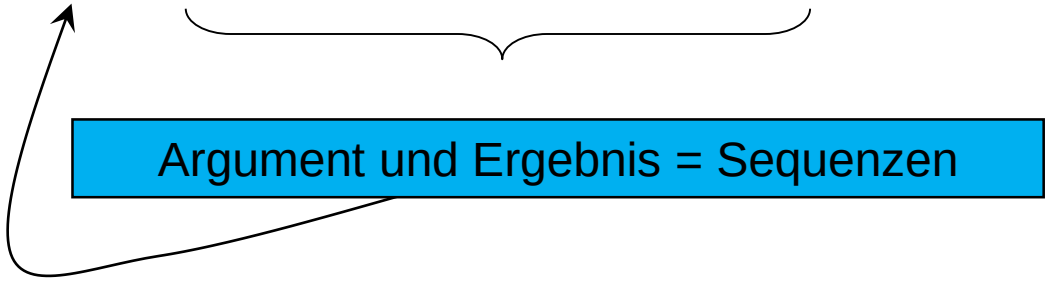
*„Alles ist eine Sequenz“*




- Ergebnis jedes XPath-Audrucks ist eine **Sequenz**
- Sequenz besteht aus beliebig vielen **Items**
- Item ist entweder ein **Knoten** oder ein **atomarer Wert** (`string`, `boolean`, `decimal`, …)
- **Duplikate** sind in Sequenzen erlaubt
- Sequenzen können **nicht verschachtelt** sein bzw. werden automatisch entschachtelt

# XPath
## Datenmodell – Sequenz

- Konsequenz aus *„Alles ist eine Sequenz“*
  - Jeder Operand eines Ausdrucks ist eine Sequenz
  - Jedes Ergebnis eines Ausdrucks ist eine Sequenz
- Zwei Eigenschaften
  - Abgeschlossen
    - Jede Operation erzeugt als Ergebnis wieder eine Sequenz
  - Ermöglicht Komposition
    - Ausdrücke können beliebig geschachtelt werden
- Beispiel
  - `count(//Course/@semesterHours)`



# XPath

## Beispiel: XML-Dokument

**CourseCatalog.xml**

```
<?xml version="1.0" encoding="UTF-8"?>
<CourseCatalog year="2019" term="summer" campus="Hagenberg">
  <!-- Software Engineering -->
  <DegreeProgramme code="0307" name="Software Engineering" abbreviation="SE">
      <Course id="cID_8314" semesterHours="1" language="en-US" semester="4">
        <Title>Introduction to semi-structured data models and XML</Title>
        <Description>Introduction of skills related to XML.<Content>Includes DTD, Schema,
                    XPath, XQuery, XSLT, JSON</Content><Exam>Final Exam required.</Exam>
                    Participation without any previous knowledge.</Description>
        <Credit formatType="ECTS">1</Credit>
        <CourseType type="Lecture"/>
        <Date startDate="02-28" endDate="05-03"/>
        <Time startTime="0800" endTime="1025" day="THU"/>
        <Room roomNumber="1.004" building="FH1">CELUM HS2</Room>
        <Instructor instructorNumber="p22080">Julian Haslinger</Instructor>
      </Course>
      <Course> ... </Course>
  </DegreeProgramme>
  <DegreeProgramme code="0456" name="Communication and Knowledge Media" abbreviation="CKM">
      ...
  </DegreeProgramme>
</CourseCatalog>
```

# XPath

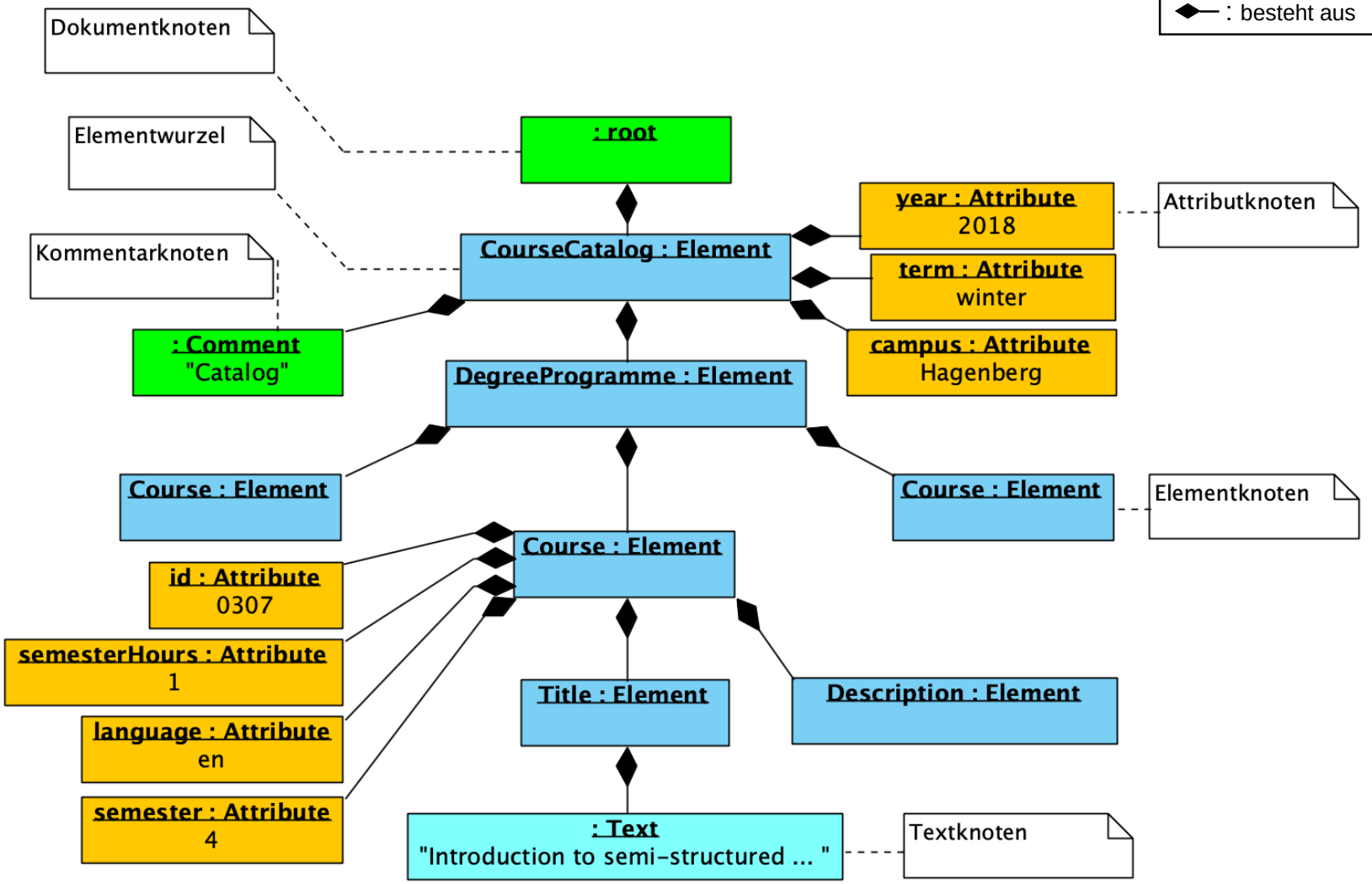
## Datenmodell – Beispiel CourseCatalog.xml

UML-Objektdiagramm

**Legende:**

Knotenname: Knotentyp
Knotenwert

: besteht aus



# XPath
## Datenmodell – Sequenz

- Beispiele
  - Geordnet
    - `(1, 2, 3, 4)` **ist verschieden von** `(4, 3, 2, 1)`
  - Duplikate erlaubt
    - `(1, 2, 3, 4)` **ist verschieden von** `(1, 1, 2, 3, 4)`
  - Heterogene Einträge (Items) möglich
    - `(1, 2, 3, "Hallo")`
    - `(1, 2, "Hallo", function($a as xs:double, $b as xs:double) as xs:double {$a * $b}, "Test")` (ab XPath 3.0)
  - Keine Schachtelung möglich (flach)
    - `(1,2, (1,2,3,4), 3)` **ist äquivalent zu** `(1, 2, 1, 2, 3, 4, 3)`

# Inhalt

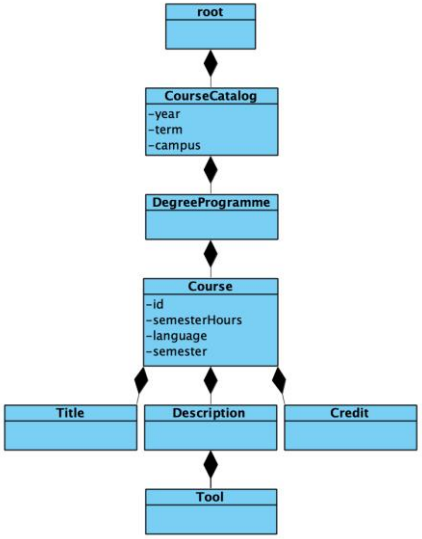
- Einführung
- Datenmodell
- **Pfadausdruck**
  - Lokalisierungsschritt
  - Achsen
  - Knotentest
  - Filter (Prädikat)
- Erweiterte Ausdrücke
- Funktionen und Operatoren
- Zusammenfassung

# XPath
## Pfadausdruck

- Pfadausdruck dient zur Lokalisierung und dem Zugriff auf Bestandteile eines XML-Dokuments (als Instanz des Datenmodells beschrieben)
  - Absoluter Pfad `/CourseCatalog/DegreeProgramme/Course`
    - Auswertung beginnt bei der Dokumentwurzel ("`/`") UNABHÄNGIG vom aktuellen Kontext
  - Relativer Pfad `DegreeProgramme/Course`
    - Auswertung beginnt beim aktuellen Kontextknoten (z.B. durch vorangegangenen `Lokalisierungsschritt` bestimmt)
- Pfadausdruck besteht aus
  - **Lokalisierungsschritten** und **Achsen**: geben die Richtung an, in der die zu selektierenden Knoten gesucht werden
  - **Knotentest**: filtert die durch die Lokalisierungsschritte selektierten Knoten
  - **Prädikate** (optional): kann die Knotenmenge weiter filtern

# XPath

## Pfadausdruck – Lokalisierungsschritt

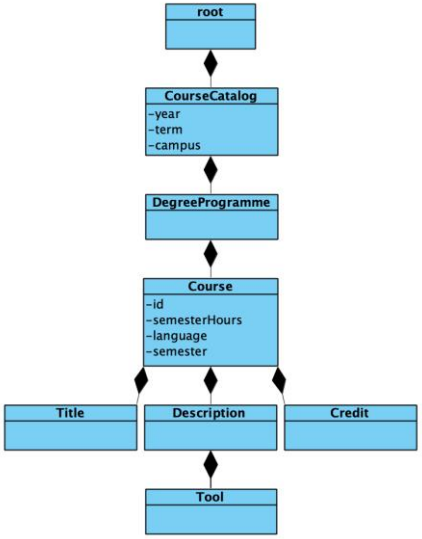



- Lokalisierungsschritt selektiert eine Knotenmenge
  - der nächste Schritt operiert auf dieser Knotenmenge
    - (jeder Knoten ist dabei einmal der Kontextknoten)
  - pro Schritt kann die Knotenmenge wachsen oder schrumpfen
  - Abarbeitung zusammengesetzter Schritte erfolgt "von links"

- Hierarchieoperatoren `/` und `//`
  - `/`
    Dokumentwurzel (*root node*)
  - `//Course`
    alle `Course`-Elemente, beliebig tief im Kontext (ausgehend von Dokumentwurzel)
  - `//Course/Title`
    alle `Title`-Subelemente von `Course`-Elementen, beliebig tief im Kontext

# XPath

## Pfadausdruck – Lokalisierungsschritt

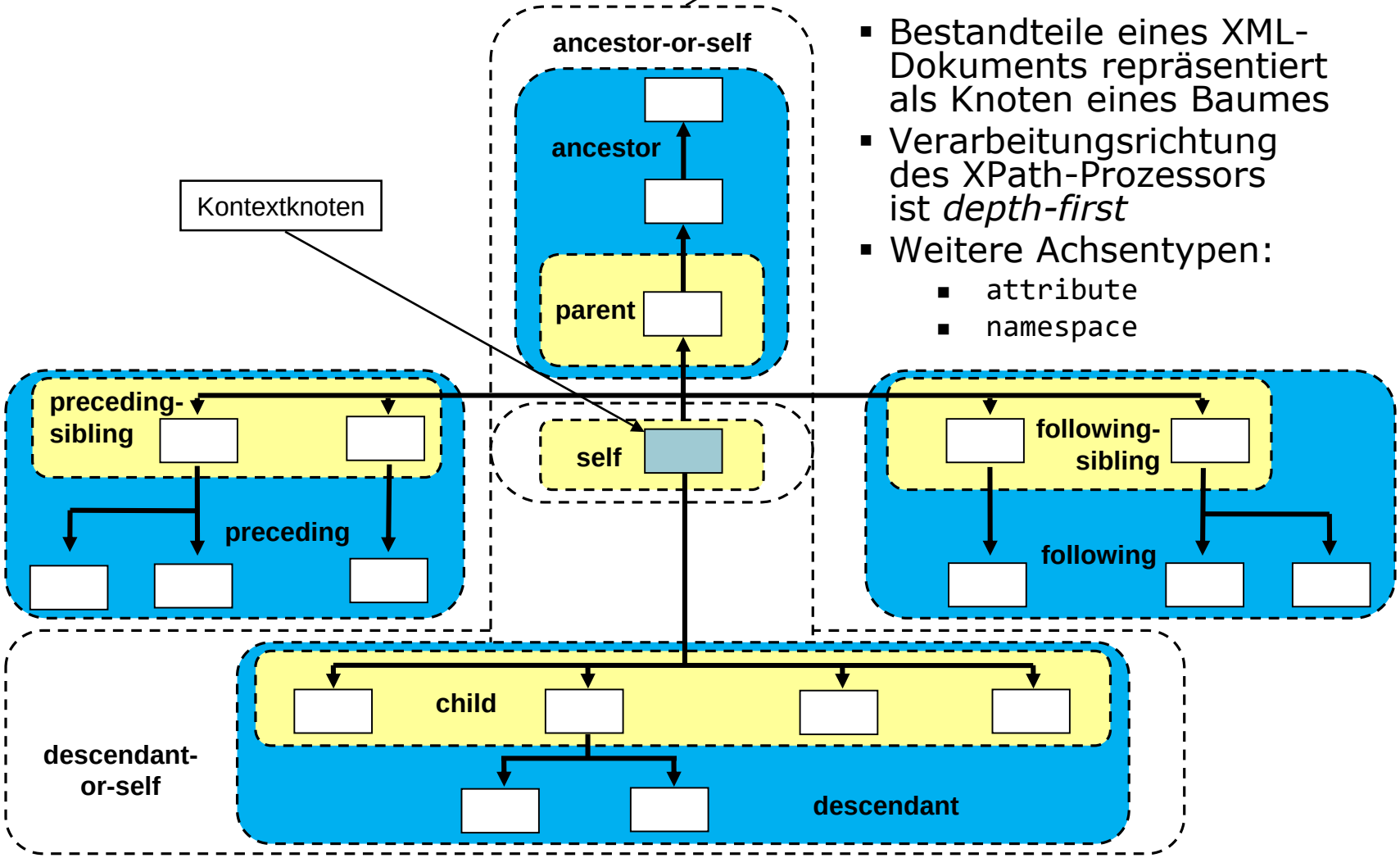



- Zugriff auf beliebige Elemente *
  - /*
    Elementwurzel
  - //*
    alle Elemente, inklusive Elementwurzel
  - `//DegreeProgramme/*/Credit`
    alle `Credit`-Elemente, sofern sie Enkel des `DegreeProgramme`-Elements sind
- Zugriff auf Attribute @
  - `//@*`
    alle Attribute aller Elemente
  - `//Course/@semester`
    alle `semester`-Attribute aller `Course`-Elemente

# XPath

## Pfadausdruck – Achsen

Achse = Beziehung zw. Kontextknoten und den zu selektierenden Knoten




- Bestandteile eines XML-Dokuments repräsentiert als Knoten eines Baumes
- Verarbeitungsrichtung des XPath-Prozessors ist *depth-first*
- Weitere Achsentypen:
  - `attribute`
  - `namespace`

# XPath
## Pfadausdruck – Achsen

- Verarbeitungsrichtung des XPath-Prozessors ist *depth-first*
- Dokumentausschnitt und Aufbau der Ergebnismenge bei unterschiedlichen Ausdrücken:

```
<CourseCatalog>
    <DegreeProgramme>
      <Course>
        <Title/>
        <Description><Content/><Exam/></Description>
        <Credit/>
        <CourseType/>
        <Date/>
        <Time/>
        <Room/>
        <Instructor/>
      </Course>
    </DegreeProgramme>
</CourseCatalog>
```

```
/CourseCatalog/DegreeProgramme/Course/*
<Title/><Description><Description><Credit/>
<CourseType/><Date/><Time/><Room/><Instructor/>
```

```
/CourseCatalog/DegreeProgramme/Course//*
<Title/><Description/><Content/><Exam/><Credit/>
<CourseType/><Date/><Time/><Room/><Instructor/>
```

Kontext-Knoten im letzten Schritt: *Course*

Ergebnis 1:

- Alle **direkten** Kind-Elemente (/*) von *Course*
- (*Content* / *Exam* ist kein Kind-Element von *Course*).

Ergebnis 2:

- Ergebnis: Alle Kind-Elemente von *Course*
- (inkl. *Content* und *Exam* als Eigene Knoten)

# XPath

## Pfadausdruck – Lokalisierungsschritte und Achsen

- Achsenname - Navigation über Achsenbezeichnung

| Langform | Kurzform |
|---|---|
| `child::element-name` | `element-name` |
| `attribute::attributename` | `@attributename` |
| `/descendant-or-self::node()/child::element-name` | `//element-name` |
| `self::node()` | `.` |
| `parent::node()` | `..` |

### Beispiele

| | |
|---|---|
| Attributwerte aller `DegreeProgramme`-Elemente | `/child::CourseCatalog/child::DegreeProgramme/attribute::name`<br>(kürzer) `/CourseCatalog/DegreeProgramme/@name` |
| Alle `Course`-Elemente, unabhängig von der Position im Baum | `/descendant-or-self::node()/child::Course`<br>(kürzer) `//Course` |

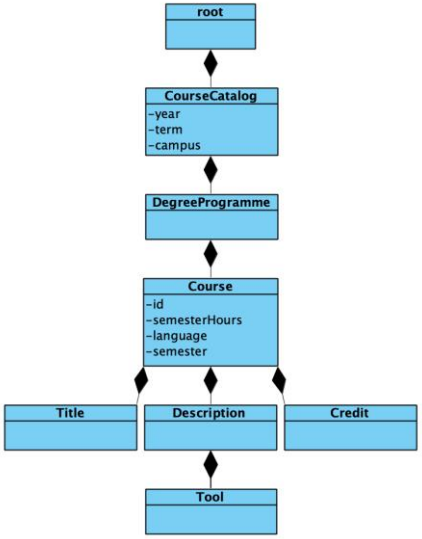
# XPath
## Pfadausdruck – Knotentest

- Knotentest gibt ein Filterkriterium für die Auswahl von Knoten an
- Knotentest erfolgt durch
  - **Namenstest**: Angabe des Knotennamens
    - `/CourseCatalog/DegreeProgramme/Course`
  - **Wildcard-Test** mittels *
    - `//* (: gibt alle Kind-Elementknoten zurück :)`
    - `//@* (: gibt alle Attributknoten zurück :)`
  - **Kindtest**: Überprüfen des Knotentyps
    - *comment*(): wählt alle Kommentarknoten aus
      z.B. `/CourseCatalog/comment()`
    - *attribute()*: wählt alle Attributknoten
    - *node()*: wählt alle Knoten unabhängig von ihrem Typ
    - *text()*: wählt alle Textknoten aus
    - ...

# XPath
## Pfadausdruck – Filter und Prädikate

- Lokalisierungsschritt kann Filter (Prädikate) enthalten
  - Auswertung der Filterbedingung für alle im Knotentest ausgewählten Knoten
  - Filter sind XPath-Ausdrücke in eckigen Klammern
- Typische Anwendungen von Filtern sind: Finde die Knoten, bei denen
  - ein bestimmter Attributwert mit einer Zeichenkette übereinstimmt
  - ein Elementinhalt mit einer Zeichenkette übereinstimmt,
  - ein Element ein bestimmtes Kindelement enthält oder
  - ein Knoten eine bestimmte Position hat.
- Zur Formulierung der Filter können Operatoren verwendet werden
  - Arithmetische Operatoren: `+, -, *, div, mod`
  - Logische Operatoren: `and, or, not`
  - Vergleichsoperatoren: `=, !=, <, <=, >, >=, ...`

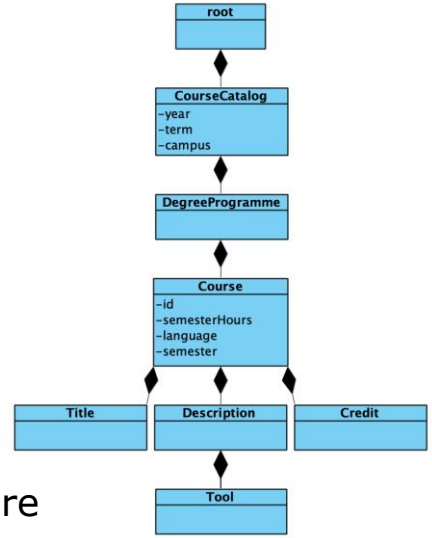
# XPath
## Filter – Existenztest




- `//Course[Title]`
  - alle `Course`-Elemente, die ein `Title`-Element enthalten
- `//Course[Title]/Description[Tool]`
  - alle `Description`-Elemente mit `Tool`-Element in `Course`-Elementen, die ein `Title`-Element enthalten
- `//DegreeProgramme[Course/Title]`
  - alle `DegreeProgramme`-Elemente, die ein `Course`-Element enthalten, das ein `Title`-Element als Kind hat
- `//Course[Title and Description]`
  - alle `Course`-Elemente mit `Title`- und `Description`-Subelementen
- `//Course[starts-with(Title, "Introduction to")]`
  - alle `Course`-Elemente, deren `Title`-Element mit „Introduction to" beginnt.
- `//Course[@id = "cID7540"]`
  - alle `Course`-Elemente, deren Attribut `id` den Wert `cID7540` hat
- `//DegreeProgramme[Course[@id = "cID7540"]]`
  - alle `DegreeProgramme`-Elemente, die ein `Course`-Element enthalten, deren Attribut `id` den Wert `cID7540` hat

# XPath

## Filter – Funktionen und Filterlisten




- XPath stellt Bibliothek von Funktionen bereit, die in Filtern verwendet werden können
  - Beispiel: Filter über Kontextposition des Knotens

| | |
|---|---|
| `//Credit[1]`<br>oder<br>`//Credit[position()=1]` | alle ersten `Credit`-Elemente von den entsprechenden Eltern-Knoten (auch mehrere möglich) |
| `(//Credit)[1]`<br>`(//Credit)[position()=1]` | das erste `Credit`-Element im gesamten Dokument |
| `//Course[last()]` | alle letzten `Credit`-Elemente von den entsprechenden Eltern-Knoten (auch mehrere möglich) |
| `(//Credit)[last()]` | das letzte `Credit`-Element im gesamten Dokument |

- Auswertung von Filterlisten
  - `//Course[starts-with(@id, "cID")][last()]/Title`
  - alle `Course`-Elemente, bei denen die `id` mit "cID" beginnt, aus dieser Knotenmenge wird der letzte `Title`-Knoten zurückgegeben. [Reihenfolge der Filter ist von Bedeutung – Auswertung von links nach rechts]

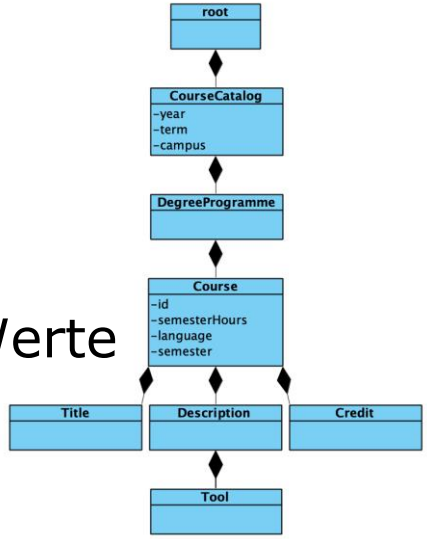
# Inhalt

- Einführung
- Datenmodell
- Pfadausdruck
- **Erweiterte Ausdrücke**
  - Vergleichsausdruck
  - Schleifenausdruck
  - Konditionaler Ausdruck
  - Quantifizierender Ausdruck
- Funktionen und Operatoren
- Zusammenfassung

# XPath
## Vergleichsausdruck

- 3 Arten von Vergleichsausdrücken
  - Wertevergleich: `eq`, `ne`, `lt`, `le`, `gt` und `ge`
  - Allgemeiner Vergleich: `=`, `!=`, `<`, `<=`, `>` und `>=`
  - Knotenvergleich: `is`, `<<` und `>>`

# XPath
## Vergleichsausdruck – Wertevergleich




- Operatoren: **eq**, **ne**, **lt**, **le**, **gt** und **ge**
- wird verwendet, um zwei einzelne (atomare) Werte gleichen Typs zu vergleichen
- bei Sequenzen mit mehr als einem Item wird ein Fehler erzeugt

- Beispiele:
  - `//Course[1]/Credit/@formatType eq "ECTS"`
    ➔ `true`
  - `//Course/Credit/@formatType eq "ECTS"`
    ➔ Fehler, da die `@formatType`-Sequenz mehrere Items enthält

# XPath

## Vergleichsausdruck – Allgemeiner Vergleich

- Operatoren: `=`, `!=`, `<`, `<=`, `>` und `>=`
- Vergleich von Sequenzen mit beliebig vielen Items
- Auswertung von Ausdrücken der Form `x <comp> y`:
  - Wert jedes Items aus `x` wird mit jedem Item-Wert aus `y` verglichen (entsprechend `<comp>`)
  - liefert einer der Vergleiche `true`, so liefert der gesamte Ausdruck `true`

- Beispiele:
  - `//* = "1"`

    ➔ `true`, da einer der Textknoten im Dokumentenbaum den Wert `'1'` enthält (ECTS)).
  - `//@* = "ECTS"`

    ➔ `true`, da ein Attributknoten den Wert “ECTS” enthält

# XPath
## Vergleichsausdruck – Knotenvergleich

- Operatoren: `is`, `<<` und `>>`
- Vergleich von zwei Knoten `x <comp> y`
- `is` liefert `true`, wenn `x` der selbe Knoten wie `y` ist
- das Ergebnis von `<<` und `>>` wird von der Dokumentenreihenfolge bestimmt
  - `<<` liefert `true`, wenn `x` ein Vorgänger von `y` ist
  - `>>` liefert `true`, wenn `x` ein Nachfolger von `y` ist
- Beispiele:
  - `/Course[1] is /Course[1]`
    ➔ `true`
  - `//Course[1] << //Course[2]`
    ➔ `true` (`Course`-Element 1 kommt vor dem `Course`-Element 2)

# XPath
## Schleifenausdruck – for

- Ergebnis des Ausdrucks wird elementweise an die Variable **$preis** gebunden und das nachfolgende Anfragekonstrukt für jedes Element einzeln ausgeführt
  - `for $instructor in //Instructor return upper-case($instructor/@instructorNumber)`

    ```
    P20621 P22080 P22100 P22101
    ```

  - `for $etcs in //Credit[@formatType="ECTS"] return $etcs`

    ```
    <?xml version="1.0" encoding="UTF-8"?>
    <Credit formatType="ECTS">1</Credit>
    <Credit formatType="ECTS">1,5</Credit>
    <Credit formatType="ECTS">0,5</Credit
    ```

- Schachtelung von **"for"**-Ausdruck erlaubt, da Ergebnis wieder eine Sequenz ist
  - `count(for $hours in //@semesterHours return $hours)`

    ```
    3
    ```

# XPath
## Konditionaler Ausdruck – if

- Testausdruck in Klammern entscheidet, ob das Ergebnis der **then**- bzw. der **else**-Klausel zurückgeliefert wird.
  - ```
    if (/CourseCatalog/DegreeProgramme [@code="0307"])
      then "SE"
      else "not SE"
    ```

- Schachtelung von Ausdrücken erlaubt
  ⇨ Ausdrucksstärke!
  - ```
    sum(
      for $course in //Course
      return
        if ($course/CourseType/@type = "LabSession")
          then $course/@semesterHours
        else
          ())
    ```

# XPath

## Quantifizierender Ausdruck – some/every

- Existenzielle Quantifizierung
  - `some $courseTypes in //Course/CourseType satisfies $courseTypes/@type = "LabSession"`
    - ➔ `true`
- Universelle Quantifizierung
  - `every $courseType in //Course/CourseType satisfies ($courseType/@type != "Training")`
    - ➔ `false`
  - `every $x in (1 to 10) satisfies ($x > 11)`
    - ➔ `false`
- Ausdrucksstärker, da beliebige Bedingung nach **satisfies** möglich ist (nicht nur **=, !=, <, ...**)
  - `some $x in (1 to 10) satisfies $x * $x < 10`
    - ➔ `true`

# Inhalt

- Einführung
- Datenmodell
- Pfadausdruck
- Erweiterte Ausdrücke
- **Funktionen und Operatoren**
- Zusammenfassung

# XPath
## Funktionen und Operatoren

- Funktionen und Operatoren, die (u.a.) in XPath verwendet werden können, werden als Katalog in **XQuery and XPath Functions and Operators (FO)** definiert
  - siehe https://www.w3.org/TR/xquery-operators/
- Standard wird zwischen unterschiedlichen Technologien geteilt (u.a. XPath, XQuery und XSLT)
- Konkrete Syntax der Funktionen und Operatoren wird in den jeweiligen Technologien beschrieben

# XPath
## Operatoren – Sequenzen

- Vereinigung
  - `(A, B) union (A, B) → (A, B)`
  - `(A, B) union (B, C) → (A, B, C)`
- Durchschnitt
  - `(A, B) intersect (A, B) → (A, B)`
  - `(A, B) intersect (B, C) → (B)`
- Differenz
  - `(A, B) except (A, B) → ()`
  - `(A, B) except (B, C) → (A)`

# XPath
## Operatoren – Vergleiche

- Wertevergleich
  - `eq`, `ne`, `lt`, `le`, `gt` und `ge`
- Allgemeiner Vergleich
  - `=`, `!=`, `<`, `<=`, `>` und `>=`
- Knotenvergleich
  - `is`, `<<` und `>>`

# XPath
## Sequenzfunktionen

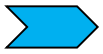
- Strukturfunktionen
  - `insert-before((7, 9, 10), 2, 8)` → `(7, 8, 9, 10)`
  - `remove((1, 2, 3), 2)` → `(1, 3)`
  - `index-of((10, 20, 30), 20)` → `2`
  - `empty(())` → `true`
  - `exists((1, 2, 3))` → `true`
  - ...
- Aggregatfunktionen
  - `sum(1, 2, 3)` → `6`  `(: Kommentar :)`
  - `count(1, 2, 3)` → `3`
  - `avg(1, 2, 3)` → `2`
  - `min(1, 2, 3)` → `1`
  - `max(1, 2, 3)` → `3`

# XPath
## Zeichenkettenfunktionen

- Konvertierung von Klein- bzw. Großbuchstaben
  - `upper-case('Introduction')` → `'INTRODUCTION'`
  - `lower-case()`
- Konkatenation
  - `concat('FH3', '.', '108')` → `'FH3.108'`
- div. weitere Zeichenketten-Funktionen
  - `ends-with()`,
  - `starts-with()`,
  - `substring-before()`,
  - `string-length()`,
  - `contains()`,
  - `normalize-space, etc.`

# XPath
## ... und viele weitere Funktionen

- Funktionen für boolesche Werte
  - `boolean(), false(), true(), not()`
- Funktionen für numerische Werte
  - `abs(), ceiling(), floor(), round()`
- Funktionen für Zeitangaben und Zeitdauer
  - `current-date(), current-time(), current-dateTime()`
- Funktionen für die Knotensuche
  - `collection(), doc(), id(), root()`
- Funktionen auf Knoten
  - `name(), local-name(), namespace-uri()`
- Funktionen für Konvertierungen
  - `number(), string(), boolean()`
- Kontextfunktionen
  - `position(), last()`


siehe www.w3.org/TR/xpath-functions/

# Inhalt

- Einführung
- Datenmodell
- Pfadausdruck
- Erweiterte Ausdrücke
- Funktionen und Operatoren
- **Zusammenfassung**

# Zusammenfassung

- XPath dient zum Navigieren in XML-Dokumenten und zur Selektion (von Teilen) von XML-Dokumenten
  - XPath 2.0 wird oft als Sprache zur Verarbeitung von Sequenzen, die auch zur Navigation in XML-Dokumenten verwendet werden kann, bezeichnet
- Bildet auch gemeinsame Grundlage von XQuery und XSLT
- XPath 2.0 Recommendation
  - siehe www.w3.org/TR/xpath20/
- Verwendet dasselbe Datenmodell wie andere XML-Technologien (**XDM**)
- Unterstützt eine Vielzahl von eingebauten Funktionen und Operatoren (**FO**)
  - siehe www.w3.org/TR/xpath-functions/
- XPath 3.0 und 3.1 sinnvolle Erweiterungen bei fortgeschrittener Anwendung von XPath

# Ressourcen

- W3C-Recommendations
  - XPath 3.1: https://www.w3.org/TR/xpath-31/
  - XPath 3.0: https://www.w3.org/TR/xpath-30/
  - XPath 2.0: https://www.w3.org/TR/xpath20/
  - XPath 1.0: https://www.w3.org/TR/1999/REC-xpath-19991116/
- Blog zum Thema XPath
  - http://www.wilfried-grupe.de/XPath.html