XPath

mit Übungen

Patrick Sahle

Übersicht

- Wozu XPath?
- XPath als Standard
- Bäume, Hierarchien, Schachteln
- Konzepte und Grundbausteine
- Übungen

Was ist XPath?

- "XPath is a language for addressing parts of an XML document” (XPath Specifications)
- XPath dient der Navigation in XML-Dokumenten und der Erzeugung von “Rückgaben”

- XPath wird vor allem in anderen X-Technologien verwandt: XSLT, XQuery

Wo findet man XPath?

- Die XPath-Spezifikation wird vom W3C (World Wide Web Consortium) gepflegt: http://www.w3.org/TR/xpath/
- Versionen: XPath 1.0, XPath 2.0, XPath 3.0

- Einfacher Einstieg: https://www.w3schools.com/xml/xpath_intro.asp
- Einfache Referenz: https://www.w3schools.com/xml/xsl_functions.asp (nur bis 2.0)
- Vollständige Referenz: https://maxtoroq.github.io/xpath-ref/

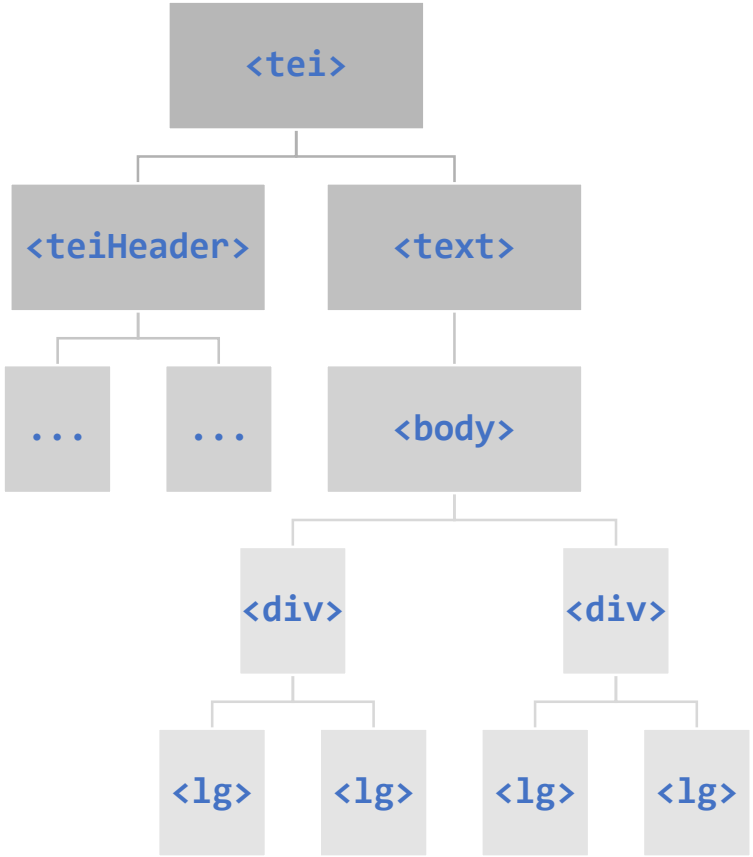
Schachteln, Hierarchien, Bäume

Sie haben es schon gelernt:

- Text ist sequentiell
- Tags + Inhalt = Elemente
- Elemente in Elementen = Verschachtelung
- Ein äußerstes Element = Eine Hierarchie!
- Ein „Baum“?

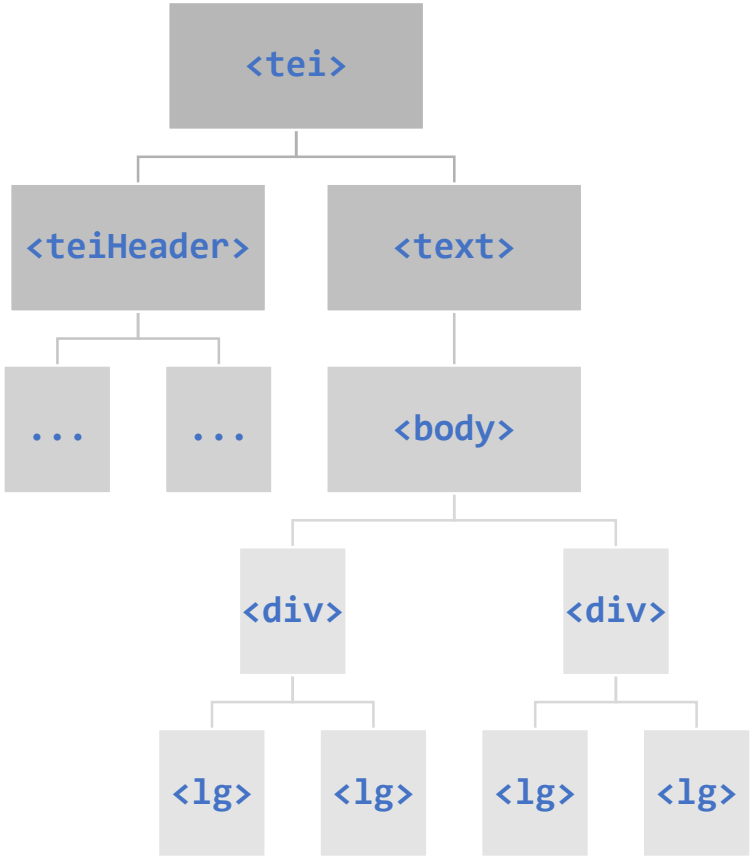
XML als Baum

XML als Baum



XML als Baum

```
<TEI xmlns="http://www.tei-c.org/ns/1.0">
 <teiHeader>...</teiHeader>
 <text>
   <body>
     <div type="sonnet">
       <head>Sonnet 18</head>
       <lg type="quatrain">
         <l n="1">Shall I compare thee to a summer's day?</l>
         <l n="2">Thou art more lovely and more temperate:</l>
        ...
       </lg>
        ...
       <lg type = "couplet">
         <l n="13">So long as men can breathe or eyes can see,</l>
         <l n="14">So long lives this and this gives life to thee. </l>
       </lg>
     </div>
   </body>
 </text>
</TEI>
```



Wichtige Begriffe

- Elemente / Attribute / Knoten
- Eltern – Kinder
- Vorfahren – Nachfahren
- Geschwister
- Wurzelknoten, Dokumentknoten)

Knotentypen

- Dokumentknoten
- Wurzelknoten
- Elementknoten
- Attributknoten
- Textknoten

- Kommentarknoten

Bewegung im Baum

Lokalisierungsschritte

„gehe von hier nach da“

Knotentests

„bist Du der, den ich suche?“

Lokalisierungspfade

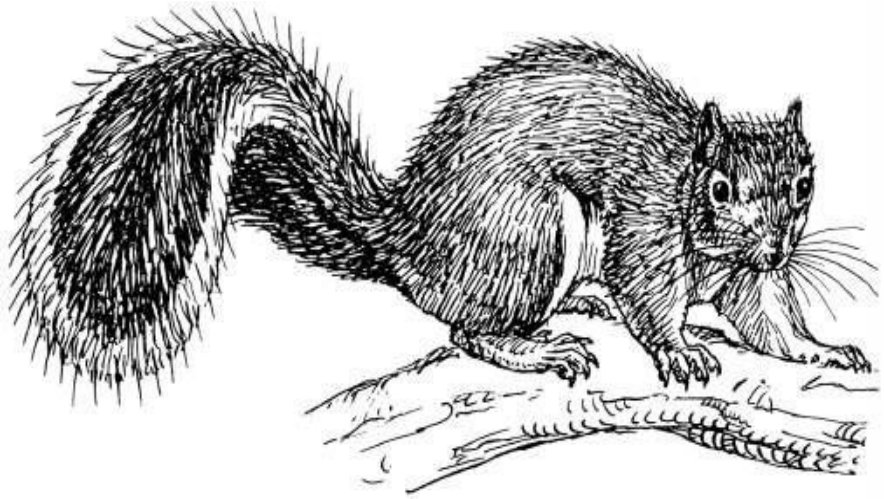
[Schritt]/[Schritt]/[Schritt]

/TEI/text/body/div

„Kontext“

absolute Pfade (vom Dokument ausgehend)

vs. relative Pfade (vom aktuellen Kontext ausgehend)



Bewegung im Baum: Achsen

Vertikale Achsen

self::		
child::	descendant::	descendant-or-self::
parent::	ancestor::	ancestor-or-self::

Horizontale Achsen

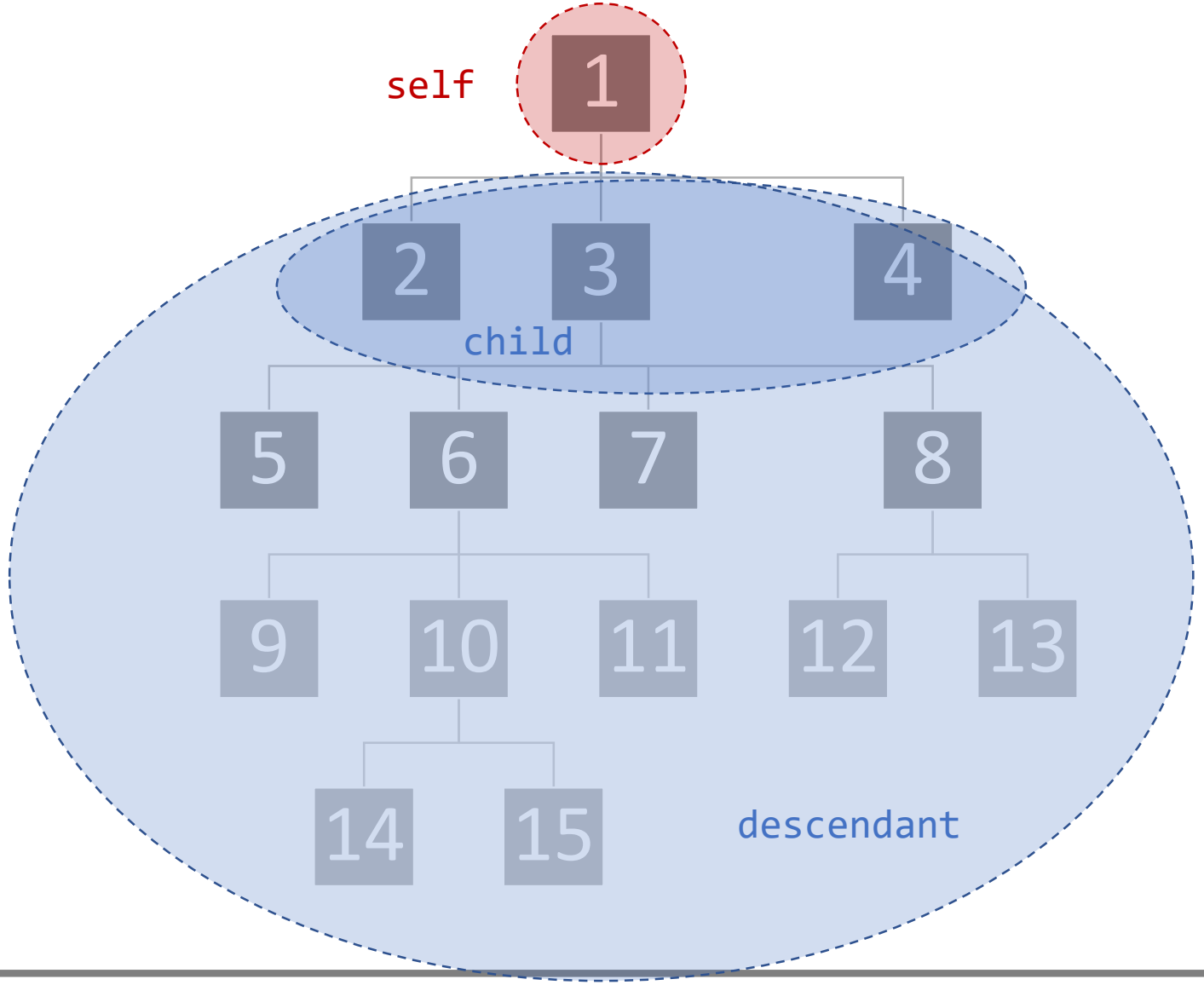
following::	following-or-self::	following-sibling::
preceding::	preceding-or-self::	preceding-sibling::

Achsen

Selbst
`self`

Eltern / Kind
`parent` / `child`

Vorfahren / Nachfahren
`ancestor` / `descendant`

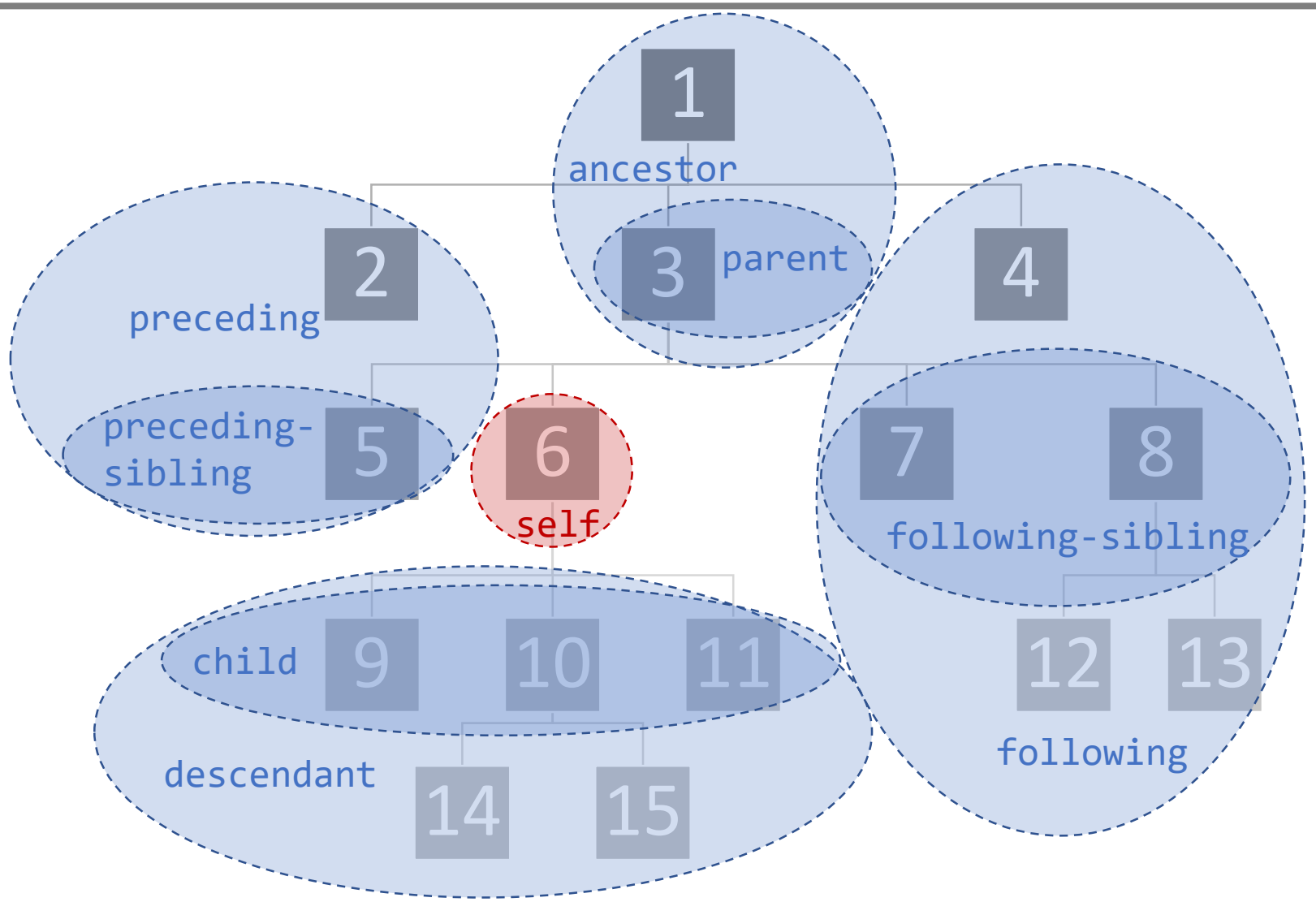


Achsen

- Selbst
  `self`
- Eltern / Kind
  `parent / child`
- Vorfahren / Nachfahren
  `ancestor`
  `descendant`
- Geschwister

  `preceding-sibling`

  `following-sibling`



Achsen: abgekürzte Syntax

self::	.
child::	/Elementname
parent::	/..
descendant-or-self::	//Elementname
attribute::	/@Attributname
Knotentest, beliebiger Elementname	*

Beispiel: /TEI/text/body/div//person/@id

Bewegung im Baum

Lokalisierungsschritte

Prinzip: Achse + Knotentest + Prädikat

Syntax: achse::knotentest[Prädikat]

Beispiel

```
//body/descendant::persName[@key='A000584']
```

Der Grundbaukasten

- Verkettete Lokalisierungsschritte /…/…/
- Bedingungen, Prädikate [……]
- Klammern, Schachtelung (…(…))
- Operatoren
  and or | = != < > + - * div
- Funktionen
  Funktionsname(*Argument*, *Argument*, …)
- Syntax: 'Strings' in Anführungszeichen! Zahlen nicht!

Die Grund-Funktionsweise

- Der letzte bzw. äußerste Schritt bestimmt, was ein XPath-Ausdruck zurückgibt
- Pfade werden von vorne nach hinten abgearbeitet
- Klammern werden von innnen nach außen aufgelöst

Beispiel

```
//body/descendant::persName[@key='A000584']
```

Rückgaben

XPath-Ausdrücke ergeben Rückgaben verschiedenen Typs:

- Knoten
- Knotenmengen
- Zahlen
- Strings
- Wahrheitswerte / Boolean
- Sequenzen

Funktionen

- Funktionen können mit ihrem Namen aufgerufen werden.
- Funktionen bestehen aus Ihrem Namen und runden Klammern:
  funktion()
- Manche Funktionen erwarten in der Klammer die Übergabe von "etwas"
  - Das kann ein Knoten sein, ein Knotensatz, ein String, eine Zahl …
- Die Übergaben sind durch Kommata getrennt
  - *funktion*(*parameter*,*parameter*)
- Funktionen geben dann etwas zurück
  - das kann ein Wahrheitswert sein, eine Zahl, ein String, eine Sequenz …

Funktionen: Ein Beispiel

contains(string,string)

- prüft, ob ein Element oder String (das erste Argument) einen anderen String (das zweite Argument) enthält
- liefert einen Wahrheitswert zurück
- Auf Deutsch: Enthält der erste String den zweiten? Wahr oder falsch? True / False?

- contains('Schnecke','ecke')
  - Deutsch: Enthält der String Schnecke den String ecke?
  - Rückgabe: true

- //forename[contains(.,‘Patrick')]
  - Deutsch: Gibt es in meinem Baum ein Element forename, das den String Patrick enthält?
  - Rückgabe: Alle Elemente forename, für die das Prädikat "true" zurückgibt (Knotenset)

Einige Funktionen I

count(nodeset)

- zählt etwas, erwartet eine Sequenz oder ein Knotenset
- liefert eine Zahl zurück

position()

- gibt die Position eines Knotens an, liefert eine Zahl zurück
- Abgekürzte Syntax: *//person[position()=11] == //person[11]*

string-length(string)

- zählt die Länge eines Strings (in Zeichen)
- liefert eine Zahl zurück

Einige Funktionen II

starts-with(string, string)

- prüft, ob ein String mit einem anderen String beginnt, liefert einen Wahrheitswert zurück
- Starts-with('Patrick‘,'P') → true

not(boolean)

- dreht einen Wahrheitswert um, liefert einen Wahrheitswert
- not(1 > 2) → true

max(sequence of numbers) ähnlich: min(), sum(), avg()

- ermittelt den maximalen Wert aus einer Reihe von Werte, liefert eine Zahl zurück
- max(//preis) → *eine Zahl*

distinct-values(sequence of strings)

- gibt eine Sequenz von (unterschiedlichen) Werten zurück
- distinct-values(//vorname) → eine Sequenz unterschiedlicher Strings

Einige Funktionen III

- substring(string,start,length)
- matches(string, pattern)
- tokenize(string, pattern)
- name()
- number(string), string(number)
- doc(URI)

Übungen

1. persons.xml in oXygen öffnen (**https://tinyurl.com/wsde19**)
2. Ein Gefühl für die Daten entwickeln
3. Den XPath-Evaluator benutzen
4. Üben heißt Übersetzen (Deutsch-XPath / XPath-Deutsch)
   Man kann sich schrittweise an das Ergebnis herantasten!

Wie lautet die Überschrift des Dokuments?

/TEI/text/body/div/head

Gib mir alle Personen zurück

/TEI/text/body/div/listPerson/person

oder //person (?) oder //listPerson/person

Übungen

Welche ist die dritte Person in der Liste?

```
//listPerson/person[position()=3]  oder  //listPerson/person[3]
```

Gib mir alle Frauen zurück

```
//listPerson/person[sex/@value='F']
```

Gib mir alle Vornamen aller Frauen zurück

```
//listPerson/person[sex/@value='F']/persName/forename
```

Bib mir alle Frauen, die mit C anfangen

```
//listPerson/person[sex/@value='F']/persName/forename[starts-with(.,'C')]
```

Gib mir die unterschiedlichen Vornamen aller Frauen zurück

```
distinct-values(//listPerson/person[sex/@value='F']/persName/forename)
```

Übungen

Zu welchen Personen liegt kein GND-Link vor?

```
//listPerson/person[not (contains(persName/@ref,'gnd'))]
```

Wieviele sind das?

```
count(//listPerson/person[not (contains(persName/@ref,'gnd'))])
```

Wieviel Prozent der Liste haben keine GND-Information?

```
count(//listPerson/person[not (contains(persName/@ref,'gnd'))])
div count(//listPerson/person) * 100
```

Das früheste Geburtsjahr in der Liste?

```
min(//listPerson/person/birth/date/@when-iso)
```

Wie alt sind die Leute im Durchschnitt geworden?

Pseudocode! Sterbejahr minus Geburtsjahr. Das einer Funktion übergeben …

```
avg(//listPerson/person/((death/date/@when-iso) - (birth/date/@when-iso)))
```

Fun: „Hallo Welt“ in XPath ?

XPath ist ein großer Spaß !