

# Textnavigation mit XPath

Daniel Schopper  
daniel.schopper@oeaw.ac.at



# Inhalt

- Was ist XPath (und wofür brauche ich es?)
- Knotentypen in XML
- Achsen
- Funktionen
- Reguläre Ausdrücke

# Was ist XPath?

"XPath is a language for **addressing** parts of an **XML document**, designed to be used by both XSLT and XPointer."

<https://www.w3.org/TR/xpath/>  
(Version 1.0 von 1999)

Die aktuelle Version ist 3.1.



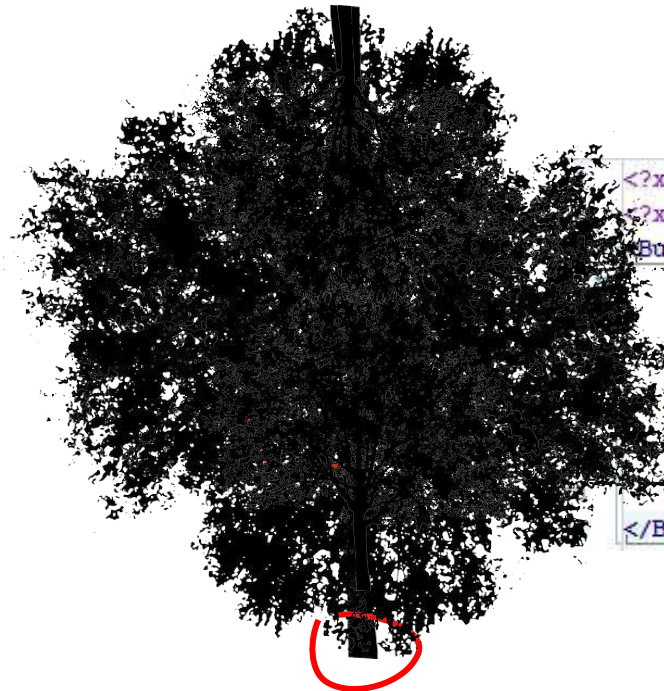
## ... und wozu brauche ich das?

- Einfache Auswertungen in einem XML-Dokument
- Basis für andere “X-Technologien”:
  - XSLT (Transformationen)
  - XQuery (komplexere Abfragen) - diese können *neue* Dokumente generieren (z.B. HTML-Ausgabe)
  - Schematron (Regeln, die sich auf bestimmte Teile von XML-Dateien beziehen)



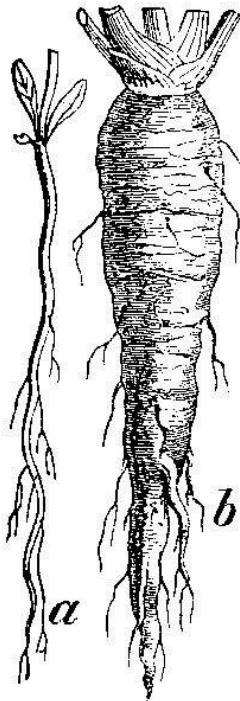


# XML-Dokumentstruktur



```
<?xml version="1.0" encoding="UTF-8"?>
<?xml-stylesheet type="text/xsl" href="style.xsl"?>
<Buch xml:id="B01234">
  <Titel>
    <Original Sprache="Englisch">Alice's Adventures in Wonderland</Original>
    <Übersetzung Sprache="Deutsch">Alices Abendteuer im Wunderland</Übersetzung>
  </Titel>
  <!-- Achtung Tippfehler, bitte ausbessern! -->
  <Autor VIAFID="66462036">Lewis Caroll</Autor>
  <Erscheinungsdatum>1965-07-04</Erscheinungsdatum>
</Buch>
```

# "Dokument-Knoten"



https://commons.wikimedia.org/wiki/File:Nsr-slika-001.png

```
1 <?xml version="1.0" encoding="UTF-8"?>
2 <?xml-stylesheet type="text/xsl" href="style.xsl"?>
3 <Buch xml:id="B01234">
4   <Titel>
5     <Original Sprache="Englisch">Alice's Adventures in Wonderland</Original>
6     <Übersetzung Sprache="Deutsch">Alices Abendteuer im Wunderland</Übersetzung>
7   </Titel>
8   <!-- Achtung Tippfehler, bitte ausbessern! -->
9   <Autor VIAFID="66462036">Lewis Caroll</Autor>
10  <Erscheinungsdatum>1965-07-04</Erscheinungsdatum>
11 </Buch>
```

root()

# XML-Dokumentstruktur: Elemente



```
1 <?xml version="1.0" encoding="UTF-8"?>
2 <?xml-stylesheet type="text/xsl" href="style.xsl"?>
3 <Buch xml:id="B01234">
4   <Titel>
5     <Original Sprache="Englisch">Alice's Adventures in Wonderland</Original>
6     <Übersetzung Sprache="Deutsch">Alices Abendteuer im Wunderland</Übersetzung>
7   </Titel>
8   <!-- Achtung Tippfehler, bitte ausbessern! -->
9   <Autor VIAFID="66462036">Lewis Caroll</Autor>
10  <Erscheinungsdatum>1965-07-04</Erscheinungsdatum>
11 </Buch>
```

Buch  
oeaw: Buch  
\*  
element ()  
node ()

# XML-Dokumentstruktur: Textknoten



```
1 <?xml version="1.0" encoding="UTF-8"?>
2 <?xml-stylesheet type="text/xsl" href="style.xsl"?>
3 <Buch xml:id="B01234">
4   ...<Titel>
5     ...<Original Sprache="Englisch">Alice's Adventures in Wonderland</Original>
6     ...<Übersetzung Sprache="Deutsch">Alices Abendteuer im Wunderland</Übersetzung>
7   ...</Titel>
8   ...<!-- Achtung Tippfehler, bitte aushessern! -->
9   ...<Autor VIAFID="66462036">Lewis Caroll</Autor>
10  ...<Erscheinungsdatum>1965-07-04</Erscheinungsdatum>
11 </Buch>
```

text ()  
node ()



# XML-Dokumentstruktur: Attribute



```
1 <?xml version="1.0" encoding="UTF-8"?>
2 <?xml-stylesheet type="text/xsl" href="style.xsl"?>
3 <Buch xml:id="B01234">
4   <Titel>
5     <Original Sprache="Englisch">Alice's Adventures in Wonderland</Original>
6     <Übersetzung Sprache="Deutsch">Alices Abendteuer im Wunderland</Übersetzung>
7   </Titel>
8   <!-- Achtung Tippfehler, bitte ausbessern! -->
9   <Autor VIA ID="66462036">Lewis Caroll</Autor>
10  <Erscheinungsdatum>1965-07-04</Erscheinungsdatum>
11 </Buch>
```

@Sprache

@xml:id

@\*

attribute::\*


# XML-Dokumentstruktur: Kommentare



```
1 <?xml version="1.0" encoding="UTF-8"?>
2 <?xml-stylesheet type="text/xsl" href="style.xsl"?>
3 <Buch xml:id="B01234">
4   <Titel>
5     <Original Sprache="Englisch">Alice's Adventures in Wonderland</Original>
6     <Übersetzung Sprache="Deutsch">Alices Abendteuer im Wunderland</Übersetzung>
7   </Titel>
8   <!-- Achtung Tippfehler, bitte ausbessern! -->
9   <Autor VIAFID="66462036">Lewis Carol</Autor>
10  <Erscheinungsdatum>1965-07-04</Erscheinungsdatum>
11 </Buch>
```

comment ()  
node ()

# XML-Dokumentstruktur: Verarbeitungsanweisungen

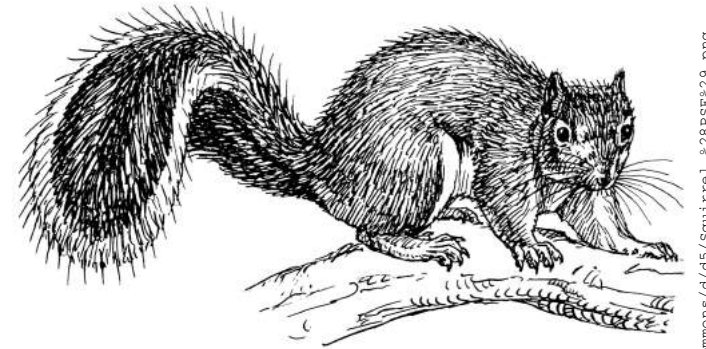


```
1 <?xml version="1.0" encoding="UTF-8"?>
2 <?xml-stylesheet type="text/xsl" href="style.xsl"?>
3 <Buch xml:id="B01234">
4   <Titel>
5     <Original Sprache="Englisch">Alice's Adventures in Wonderland</Original>
6     <Übersetzung Sprache="Deutsch">Alices Abendteuer im Wunderland</Übersetzung>
7   </Titel>
8   <!-- Achtung Tippfehler, bitte ausbessern! -->
9   <Autor VIAFID="66462036">Lewis Caroll</Autor>
10  <Erscheinungsdatum>1965-07-04</Erscheinungsdatum>
11 </Buch>
```

processing-instruction()  
node()

# Bewegung im Baum

- Schrittweise Bewegung im Pfad durch **XPath-Ausdrücke**
- Schritte im Pfad werden durch "/" getrennt
- XML-Verarbeitung beginnt am Dokument-Knoten
- Es gibt **absolute Pfadangaben**, die vom Dokument-Knoten aus starten und **relative Pfadangaben**, die vom aktuellen Kontext ausgehen
- XPath-Schritte haben eine **Richtung**, die sog. "Achse"



[https://upload.wikimedia.org/wikipedia/commons/d/d5/Squirrel\\_1\\_28PSF%29.png](https://upload.wikimedia.org/wikipedia/commons/d/d5/Squirrel_1_28PSF%29.png)



## “Familienverhältnisse”

- **Eltern / Kind**  
unmittelbar unter- bzw. übergeordnete Knoten
- **Vorfahren / Nachfahren**  
mittelbar unter- bzw. übergeordnete Knoten
- **Geschwister**  
Knoten auf gleicher hierarischer Ebene
- **Selbst**  
Der Knoten des aktuelle Kontexts

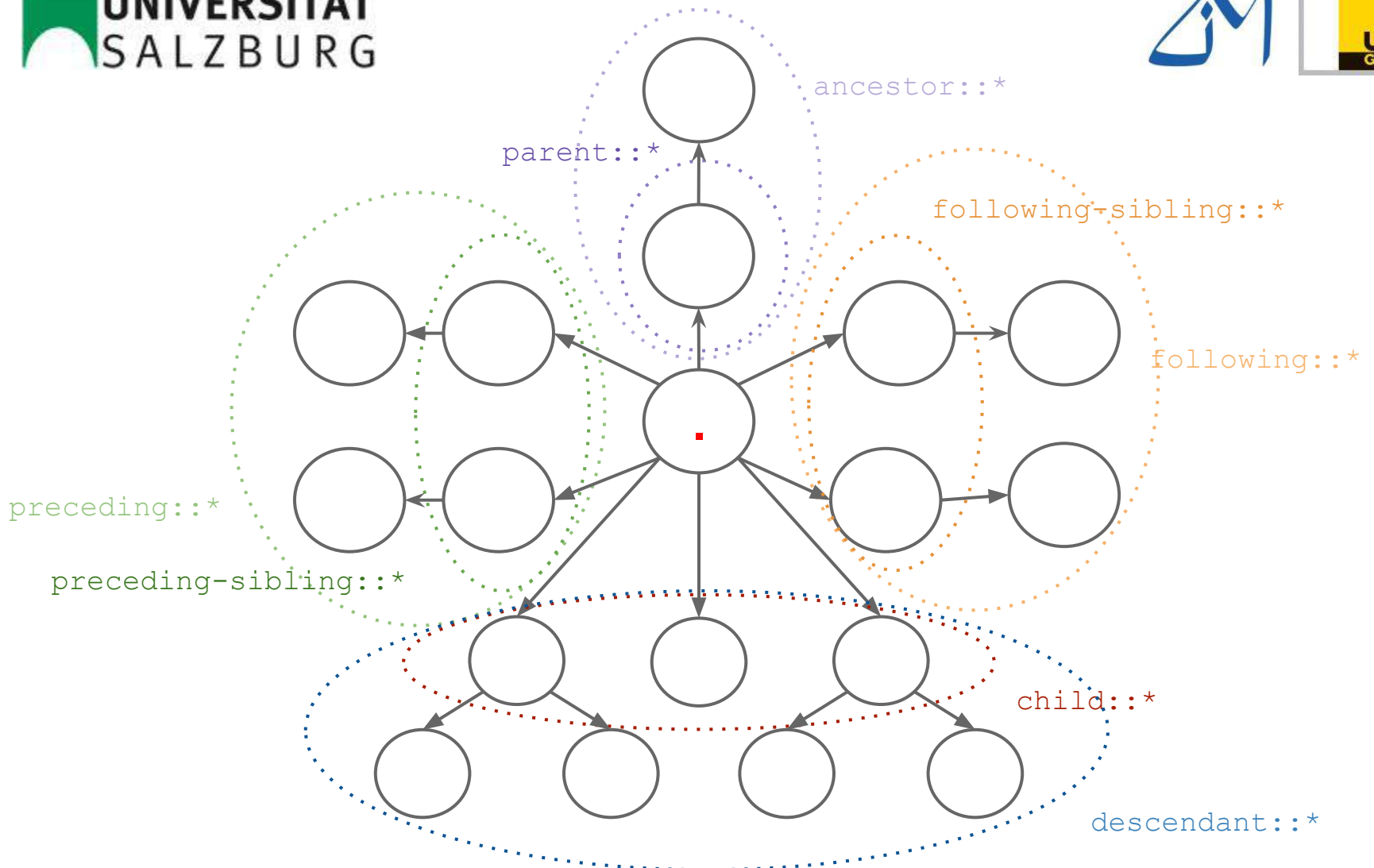


# Achsen

- **Eltern / Kind**  
/parent::\* bzw. /child::\*  
/.. bzw. /\*
- **Vorfahren / Nachfahren**  
ancestor::\* bzw. descendant::\*
- **Geschwister**  
preceding-sibling::\* bzw.  
following-sibling::\*
- **Selbst**  
self::  
.



[https://commons.wikimedia.org/wiki/File:Panak\\_4.jpg](https://commons.wikimedia.org/wiki/File:Panak_4.jpg)



```
<TEI>
  <teiHeader>
    <!-- etc. etc. -->
  </teiHeader>
  <text>
    <front>
      <titlePart>Begrüßung</titlePart>
    </front>
    <body>
      <p>
        <w rend="bold">Hallo</w>
        <w>Welt</w>
        <pc>!</pc>
      </p>
    </body>
    <back>
      <p>Auf Wiedersehen!</p>
    </back>
  </text>
</TEI>
```

**self::\*** oder .  
**child::\*** oder /\*  
**descendant::\***  
**descendant-or-self::\*** oder //

**preceding-sibling::\***  
**preceding::\***  
**following-sibling::\***  
**following::\***  
**parent::\***  
**ancestor::\***  
**attribute::\*** oder @\*



## XPath-Ausdrücke

**Achse::** Knotentest [Prädikat]

/Buch

/Buch/@Sprache

/Buch/Titel/Original

Titel/\*

/Buch/Titel/\*[1]

/Buch/Titel/\*[@Sprache="Englisch"]

# Operatoren

=	Gleichheit (String)
!=	Ungleichheit (String)
gt oder >	Größer als
lt oder <	Kleiner als
+ - * div	Addition, Subtraktion, Multiplikation, Division
and	Verknüpfung mit "UND"
or	Verknüpfung mit "ODER"
	Vereinigt zwei Knotenmengen in Dokumentreihenfolge, z.B. //a   //b
(..., ...)	Konstruiert eine Sequenz in der gegebenen Reihenfolge

## Resultate von XPath-Ausdrücken

1. Knotenmengen aus dem XML-Baum (Elemente, Attribute ...)
2. Zeichenketten ("Strings")
3. Wahrheitswerte ("Boolean")
4. Zahlen

XPath-Ausdrücke geben **Sequenzen** zurück:

Diese sind **geordnet** und können **Duplikate** enthalten.

2-4 sind "atomare Werte", die *nicht* Teil des XML-Baums sind

Wenn ein Ausdruck *kein* Resultat zurückliefert, ist das eine "leere Sequenz"  
(~ leere Menge)

# Übung



Öffnen Sie das Dokument **beispiel2.xml** und finden Sie ...

- ... alle Absätze `//p`
- ... nur Absätze innerhalb von `<text>` `//text//p` oder `//p[ancestor::text]`
- ... alle "Tokens" (sowohl `<w>` als auch `<pc>`-Elemente) `//w|//pc`
- ... den Token mit dem Wert "Hallo". `//w[. = "Hallo"]`  
`//text()[. = "Hallo"]/..`
- ... alle fettgedruckten Passagen `//*[@rend = "bold"]`

Klicken Sie in Oxygen auf unterschiedliche Knoten im Beispieldokument und experimentieren Sie mit Achsen und Knotentests. Z.B. Welche Kind-Knoten erhalten Sie am Kontext `<p>` im `<body>` mit dem Knotentest `node()` ?



# Übung



Öffnen Sie Dokument **beispiel1.xml**:

- Probieren Sie aus: Wodurch unterscheiden sich die Ausdrücke

`//Erscheinungsdatum | //Übersetzung`

und

`(//Erscheinungsdatum, //Übersetzung) ?`

- Wie viele Knoten enthält das Beispiel insgesamt?

`//root() | //node() | //@* | //processing-instruction() | //comment()`

# Funktionen

## Funktionsname(Argument 1, Argument2 ...)

- Funktionen verlangen unterschiedliche **Arten** und **Anzahlen** von Argumenten
- Argumente können **Knoten** oder **Werte** sein
- Funktionen “tun” Verschiedenes und liefern dementsprechend unterschiedliche Ergebnisse zurück:

```
contains("Beispiel", "spiel")
```

```
contains(//titleStmt/title, "Beispiel")
```

```
string-join(("Bei", "spiel"), "-")
```

## Funktionen: Aggregation

- Summiere alle Zahlen der Sequenz ... `sum((1, 2, 3))`
- Zähle alle Items in einer Sequenz `count("a", 2, //p)`
- Finde den höchsten Wert unter den Zahlen ... `max(5, 2, 21)`
- Finde den niedrigsten Wert unter den Zahlen ... `min(0, -1, 2)`
- Berechne den Durchschnitt der Zahlen... `avg(5, 2, 21)`

## Funktionen: Nodesets, Sequenzen u.a.

- Entferne gleiche Werte `distinct-values ("a", "b", "a")`
- Finde Position von "a" `index-of ("a", "b", "a"), "b"`
- Kehre Sequenz um `reverse (1 to 10)`
- Teilsequenz `subsequence (1 to 10, 5, 2)`
- Kehre boolschen Wert um `not (1 = 2)`
- Existiert Knotentyp x ? `exists (@type)`
- Lade Dokument **von Adresse**

`doc ("https://www.w3.org/TR/xml/REC-xml-20081126.xml")`



## Funktionen: Strings

- Enthält x y ? `contains("Subtext", "text")`
- Beginnt x mit y ? `starts-with("text", "a")`
- Endet x mit y ? `ends-with("Subtext", "texte")`
- Umwandlung in Großbuchstaben `upper-case("apfel")`
- Verknüpfe x mit y `concat("Titel:", //title)`
- Extrahiere Zeichen 7-13 `substring("Summer School", 8, 6)`
- Zähle die Zeichen in einem String  
`string-length("donaudampfschiffahrtsgesell...")`

# Reguläre Ausdrücke

Reguläre Ausdrücke beschreiben **Muster von Zeichenfolgen**

Z.B. finde in einem Textknoten eine Datumsangabe, bestehend aus:

- 1-2 **Ziffern**, gefolgt von einem **Punkt** und **eventuell** einem **Spatium**
- 1-2 **Ziffern**, gefolgt von einem **Punkt** und **eventuell** einem **Spatium**
- genau 4 **Ziffern**

```
\d{1,2} \. \s? \d{1,2} \s? \d{4,4}
```

# Reguläre Ausdrücke: Zeichenklassen

.	ein beliebiges Zeichen
\s	Leerzeichen (inkl. Zeilenumbrüche und Tabs)
\d	Ziffern
\w	"Wörter-Zeichen" (~ alles außer Satzzeichen)
\IsGreek	Unicode-Block Griechisch
\p{Lu} bzw. \P{Lu}	Großbuchstaben bzw. alles <i>außer</i> Großbuchstaben
[d-g]	Zeichen "d" bis "g"

# Reguläre Ausdrücke: Quantifikatoren, Modifikatoren

.+	ein oder mehr Zeichen
.?	ein oder kein Zeichen
.*	kein oder mehr Zeichen
^.	ein Zeichen am Anfang
.\$	ein Zeichen am Ende
.{2,4}	2 - 4 Zeichen
.{2,}	2 oder mehr Zeichen
.{,4}	4 oder weniger Zeichen

# Reguläre Ausdrücke: Funktionen (1)

Reguläre Ausdrücke sind in XPath nützlich ...

- ... zur **Analyse** von Text, z.B. *Enthält ein Textknoten eine Datumsangabe?*

```
matches(text(), "\d{1,2}\.\d{1,2}\d{4,4}")
```

- ... zur **Manipulation**, z.B. Drehe Datumsangabe "10.12.1724" um zu "1724-12-10":

```
replace("10.12.1724",  
        "(\d{1,2})\.\d{1,2}\d{4,4}",  
        "$3-$2-$1")
```



## Reguläre Ausdrücke: Funktionen (2)

Reguläre Ausdrücke sind in XPath nützlich ...

- ... zur automatischen Wortsegmentierung eines Textes ("Tokenisierung"):

```
tokenize("Der grüne Heinrich", "\s+")
```



```
("Der", "grüne", "Heinrich")
```

# Übungen

Öffnen Sie Beispieldokument **beispiel3.xml**:

- Alle Personen, deren Nachname mit “A” beginnt  
`//surname[starts-with(., "A")]/ancestor::person`
- Kinos, die nicht “Kino” in ihrem Namen tragen  
`//orgName[@type="cinema"][not(contains(lower-case(.), 'kino'))]`
- Welche im Text genannten Personen emigrierten?  
`//person[birth//country != death//country]`
- Welche der im Text erwähnten Personen wurde im 20. Jahrhundert geboren?  
`//person[starts-with(birth/@when, '19')]/persName`
- Alle im Fließtext genannten Länder (ohne Duplikate)  
`//body//distinct-values(placeName[@type='country'])`

# Übungen

- Alle Absätze, die das Wort “Auge” enthalten  
`//body//p[contains(., 'Auge')]`
- Überschrift des längsten Artikels  
`//div[string-length(.) = max(//div/string-length())]/head`
- Erstellen Sie eine Übersicht mit Typ und Länge jedes Artikels pro Zeile:  
`//div/concat(@type, ": ", count(tokenize(., '\s')))`
- Wie viele distinkte Wortformen gibt es im Text?  
`count(distinct-values(//text//p/tokenize(., '\s')[. != '']))`
- Mit welchen Reimwörtern ist das Gedicht gereimt?  
`//1/tokenize(., '\s')[last()]`

# Referenzen

## Übersichten:

<https://www.data2type.de/xml-xslt-xslfo/xpath/referenz/>

[https://www.w3schools.com/xml/xsl\\_functions.asp](https://www.w3schools.com/xml/xsl_functions.asp)

[https://www.w3schools.com/xml/xpath\\_intro.asp](https://www.w3schools.com/xml/xpath_intro.asp)

## Spezifikationen:

<https://www.w3.org/TR/xpath-31/>

<https://www.w3.org/TR/xpath-functions-31/>

*Vielen Dank für Ihre Aufmerksamkeit!*  
daniel.schopper@oeaw.ac.at

