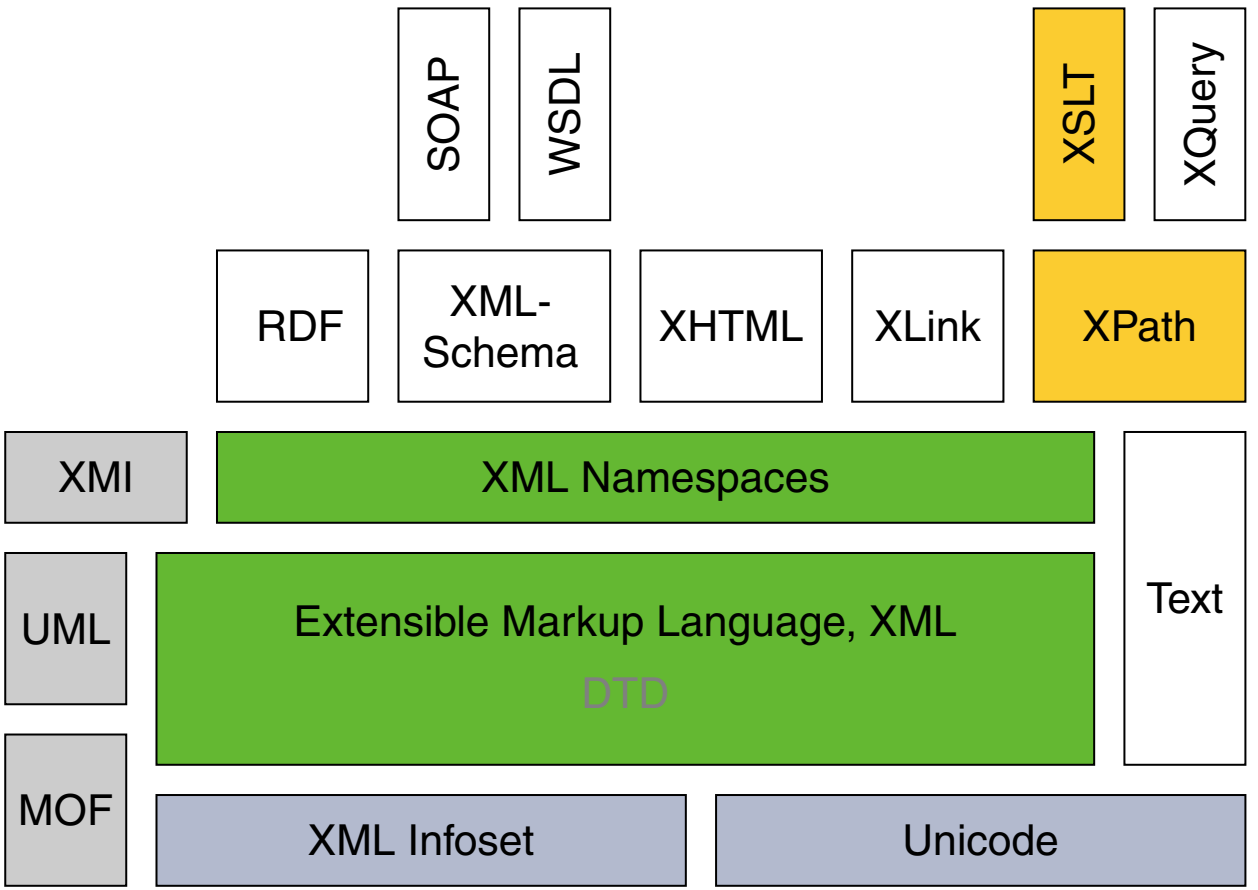
Kapitel WT:III (Fortsetzung)

III. Dokumentsprachen

- Auszeichnungssprachen
- HTML
- Cascading Stylesheets CSS
- XML-Grundlagen
- XML-Schema
- Die XSL-Familie
- Parse-Paradigmen und APIs für XML

Die XSL-Familie

Einordnung [Jeckle 2004]



Die XSL-Familie [W3C xsl home, reports]

Historie: zentrale XML-Spezifikationen

2006 Extensible Markup Language (XML) 1.1. Recommendation. [W3C REC, status]

2004 XML Schema Part 0: Primer. Recommendation. [W3C REC, status]

2012 XML Schema (XSD) 1.1 Part 1: Structures. [W3C REC]

2012 XML Schema (XSD) 1.1 Part 2: Datatypes. [W3C REC]

2021 XSL Transformations (XSLT) 2.0. Recommendation. [W3C REC, status]

2017 XML Path Language (XPath) 3.1. Recommendation. [W3C REC, status]

2017 XML Query Language (XQuery) 3.1. Recommendation. [W3C REC, status]

2012 XSL Formatting Objects (XSL-FO) 2.0. Working Draft. [W3C WD, Wiki]

Bemerkungen:

- “[The extensible stylesheet language family] XSL is a family of recommendations for defining XML document transformation and presentation. It consists of three parts:” XSLT, XPath, XSL-FO. [W3C]
- CSS versus XSL. Why two Style Sheet languages? [W3C 1, 2]
- Die Formatierungsmöglichkeiten von XSL-FO orientier(t)en sich an den Anforderungen von Print-Medien und ziel(t)en auf eine Ablösung von PDF. Schwerpunkte der Standards: CSS für HTML und XSL-FO für Print.

  Die Entwicklung des XSL-FO-Standards wurde 2013 gestoppt und wird durch die Entwicklung von CSS3-page fortgesetzt. [stackoverflow, readwritecode]

- Schritte eines XSL-Verarbeitungsprozesses [W3C]:

Die XSL-Familie

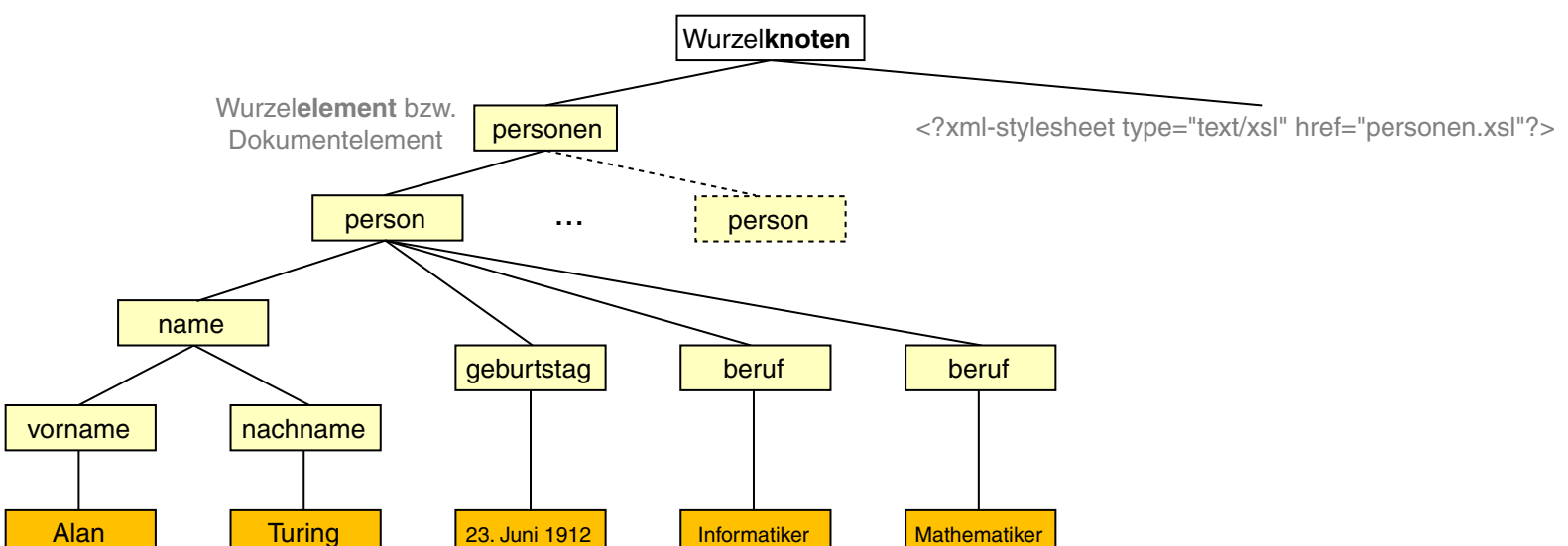
Verwendung von XPath

XSLT	Finden und Auswählen von Elementen im Eingabedokument, die in das Ausgabedokument kopiert/transformiert werden.
XQuery	Finden und Auswählen von Elementen.
XPointer	Identifikation einer Stelle im XML-Dokument, auf die ein XLink verweist.
XML-DOM-API	XPath-Interface zum Zugriff auf den DOM.
XML-Schema	Formulierung von Constraints hinsichtlich der Eindeutigkeit oder der Identität von Elementen.
XForm	Bindung von Formularsteuerungen an Instanzdaten; Formulierung von Werte-Constraints und Berechnungen.

Die XSL-Familie

XML-Knotentypen unter dem XPath-Modell

1. Wurzelknoten
2. Elementknoten
3. Textknoten
4. Attributknoten
5. Kommentarknoten
6. Verarbeitungsanweisungsknoten
7. Namensraumknoten



Bemerkungen:

- Der Wurzel*knoten* eines XML-Dokuments ist nicht identisch mit dem Wurzel*element*: Der Wurzelknoten entspricht dem *Document Information Item* des XML Information Sets. Das Wurzelelement hingegen ist das erste benannte Element des Dokuments und wird durch ein *Element Information Item* dargestellt.
- XPath dient zur Navigation in Dokumenten und der Auswahl von Dokumentbestandteilen; XPath ist keine Datenmanipulationssprache.
- XPath-Ausdrücke können zu einzelnen Knoten (XML-Element, XML-Attribut), zu Knotenmengen, zu Zeichenketten, zu Zahlen und zu Bool'schen Werten evaluieren. XPath stellt deshalb Funktionen zum Zugriff auf Knotenmengen und zur Manipulation verschiedener Datentypen zur Verfügung.
- Wiederholung. Das W3C hat mittlerweile drei Datenmodelle für XML-Dokumente definiert: XML Information Set, XPath, Document Object Model (DOM). Das XPath-Datenmodell basiert auf einer Baumstruktur, die bei der Abfrage eines XML-Dokuments durchlaufen wird und ist dem XML Information Set ähnlich; DOM ist der Vorläufer beider Datenmodelle. DOM und das XPath-Datenmodell können als Interpretationen des XML Information Sets betrachtet werden. [MSDN]

Die XSL-Familie

XPath-Lokalisierungspfade

- Ein Lokalisierungspfad spezifiziert eine eventuell leere Menge von Knoten in einem XML-Dokument.
- Ein Lokalisierungspfad setzt sich aus aufeinander folgenden Lokalisierungsschritten (*Location steps*) zusammen.
- Jeder Lokalisierungsschritt wird relativ zu einem bestimmten Knoten des XML-Dokuments ausgewertet, der dann als aktueller Knoten (*Current node*) oder Kontextknoten bezeichnet wird.

Die XSL-Familie

XPath-Lokalisierungspfade

- Ein Lokalisierungspfad spezifiziert eine eventuell leere Menge von Knoten in einem XML-Dokument.
- Ein Lokalisierungspfad setzt sich aus aufeinander folgenden Lokalisierungsschritten (*Location steps*) zusammen.
- Jeder Lokalisierungsschritt wird relativ zu einem bestimmten Knoten des XML-Dokuments ausgewertet, der dann als aktueller Knoten (*Current node*) oder Kontextknoten bezeichnet wird.

- Lokalisierungsschritte werden durch Schrägstriche (*Slashes*) getrennt:

  . . . / *Schritt_i* / *Schritt_i+1* / . . .

- Beginnt ein Lokalisierungspfad mit einem Schrägstrich, bezeichnet dieser den Wurzelknoten. Der Wurzelknoten ist dann Kontextknoten zum ersten Lokalisierungsschritt:

  / *Schritt_1*

Die XSL-Familie

XPath-Lokalisierungspfade (Fortsetzung)

```
<?xml version="1.0" ?>
<?xml-stylesheet type="text/xsl" ...?>

<personen>
  <person>
    <name>
      <vorname>Alan</vorname>
      <nachname>Turing</nachname>
    </name>
    <geburtstag>23. Juni 1912</geburtstag>
    <beruf>Mathematiker</beruf>
    <beruf>Informatiker</beruf>
  </person>

  <person>
    <name>
      <vorname>Judea</vorname>
      <nachname>Pearl</nachname>
    </name>
    <geburtstag>unknown</geburtstag>
    <beruf>Informatiker</beruf>
  </person>
</personen>
```

Beispiele für Lokalisierungspfade:

(a) `/personen/person`

Die XSL-Familie

XPath-Lokalisierungspfade (Fortsetzung)

```
<?xml version="1.0" ?>
<?xml-stylesheet type="text/xsl" ...?>

<personen>
  <person>
    <name>
      <vorname>Alan</vorname>
      <nachname>Turing</nachname>
    </name>
    <geburtstag>23. Juni 1912</geburtstag>
    <beruf>Mathematiker</beruf>
    <beruf>Informatiker</beruf>
  </person>

  <person>
    <name>
      <vorname>Judea</vorname>
      <nachname>Pearl</nachname>
    </name>
    <geburtstag>unknown</geburtstag>
    <beruf>Informatiker</beruf>
  </person>
</personen>
```

Beispiele für Lokalisierungspfade:

(a) `/personen/person`

Die XSL-Familie

XPath-Lokalisierungspfade (Fortsetzung)

```
<?xml version="1.0" ?>
<?xml-stylesheet type="text/xsl" ...?>

<personen>
  <person>
    <name>
      <vorname>Alan</vorname>
      <nachname>Turing</nachname>
    </name>
    <geburtstag>23. Juni 1912</geburtstag>
    <beruf>Mathematiker</beruf>
    <beruf>Informatiker</beruf>
  </person>

  <person>
    <name>
      <vorname>Judea</vorname>
      <nachname>Pearl</nachname>
    </name>
    <geburtstag>unknown</geburtstag>
    <beruf>Informatiker</beruf>
  </person>
</personen>
```

Beispiele für Lokalisierungspfade:

(a) `/personen/person`

(b) `/personen/person[1]/name/vorname`

Die XSL-Familie

XPath-Lokalisierungspfade (Fortsetzung)

```
<?xml version="1.0" ?>
<?xml-stylesheet type="text/xsl" ...?>

<personen>
  <person>
    <name>
      <vorname>Alan</vorname>
      <nachname>Turing</nachname>
    </name>
    <geburtstag>23. Juni 1912</geburtstag>
    <beruf>Mathematiker</beruf>
    <beruf>Informatiker</beruf>
  </person>

  <person>
    <name>
      <vorname>Judea</vorname>
      <nachname>Pearl</nachname>
    </name>
    <geburtstag>unknown</geburtstag>
    <beruf>Informatiker</beruf>
  </person>
</personen>
```

Beispiele für Lokalisierungspfade:

(a) `/personen/person`

(b) `/personen/person[1]/name/vorname`

Die XSL-Familie

XPath-Lokalisierungspfade (Fortsetzung)

```
<?xml version="1.0" ?>
<?xml-stylesheet type="text/xsl" ...?>

<personen>
  <person>
    <name>
      <vorname>Alan</vorname>
      <nachname>Turing</nachname>
    </name>
    <geburtstag>23. Juni 1912</geburtstag>
    <beruf>Mathematiker</beruf>
    <beruf>Informatiker</beruf>
  </person>

  <person>
    <name>
      <vorname>Judea</vorname>
      <nachname>Pearl</nachname>
    </name>
    <geburtstag>unknown</geburtstag>
    <beruf>Informatiker</beruf>
  </person>
</personen>
```

Beispiele für Lokalisierungspfade:

(a) `/personen/person`

(b) `/personen/person[1]/name/vorname`

(c) `/personen/person[1]/beruf`

Die XSL-Familie

XPath-Lokalisierungspfade (Fortsetzung)

```
<?xml version="1.0" ?>
<?xml-stylesheet type="text/xsl" ...?>

<personen>
  <person>
    <name>
      <vorname>Alan</vorname>
      <nachname>Turing</nachname>
    </name>
    <geburtstag>23. Juni 1912</geburtstag>
    <beruf>Mathematiker</beruf>
    <beruf>Informatiker</beruf>
  </person>

  <person>
    <name>
      <vorname>Judea</vorname>
      <nachname>Pearl</nachname>
    </name>
    <geburtstag>unknown</geburtstag>
    <beruf>Informatiker</beruf>
  </person>
</personen>
```

Beispiele für Lokalisierungspfade:

(a) `/personen/person`

(b) `/personen/person[1]/name/vorname`

(c) `/personen/person[1]/beruf`

Die XSL-Familie

XPath-Lokalisierungspfade (Fortsetzung)

Allgemeine Form eines Lokalisierungsschritts:

$$\ldots \; / \; \textit{Achse} :: \textit{Knotentest} \, [\textit{Prädikat}] \; / \; \ldots$$

1. Achse. Spezifiziert eine Knotenmenge relativ zum Kontextknoten. Es werden 13 Achsen unterschieden, `child::` ist die Defaultachse.

Die XSL-Familie

XPath-Lokalisierungspfade (Fortsetzung)

Allgemeine Form eines Lokalisierungsschritts:

$$\ldots \; / \; \textit{Achse} :: \textit{Knotentest} \; [\textit{Prädikat}] \; / \; \ldots$$

1. Achse. Spezifiziert eine Knotenmenge relativ zum Kontextknoten. Es werden 13 Achsen unterschieden, `child::` ist die Defaultachse.

2. Knotentest. Filtert die durch eine Achse (1.) spezifizierte Knotenmenge weiter. Hierzu gibt es für jeden Knotentyp ein Testschema.

   Beispiele:

   - ein qualifizierender Name (wie „Person“) $\widehat{=}$ Test auf Knoten mit diesem Namen
   - die Funktion `text()` $\widehat{=}$ Test auf den Typ „Textknoten“

Die XSL-Familie

XPath-Lokalisierungspfade (Fortsetzung)

Allgemeine Form eines Lokalisierungsschritts:

. . . / *Achse* : : *Knotentest* [*Prädikat*] / . . .

1. Achse. Spezifiziert eine Knotenmenge relativ zum Kontextknoten. Es werden 13 Achsen unterschieden, `child::` ist die Defaultachse.

2. Knotentest. Filtert die durch eine Achse (1.) spezifizierte Knotenmenge weiter. Hierzu gibt es für jeden Knotentyp ein Testschema.

   Beispiele:

   - ein qualifizierender Name (wie „Person“) $\widehat{=}$ Test auf Knoten mit diesem Namen
   - die Funktion `text()` $\widehat{=}$ Test auf den Typ „Textknoten“

3. Prädikat. Filtert die durch Achse (1.) und Knotentest (2.) spezifizierte Knotenmenge weiter. Jeder gültige XPath-Ausdruck kann Prädikat sein.

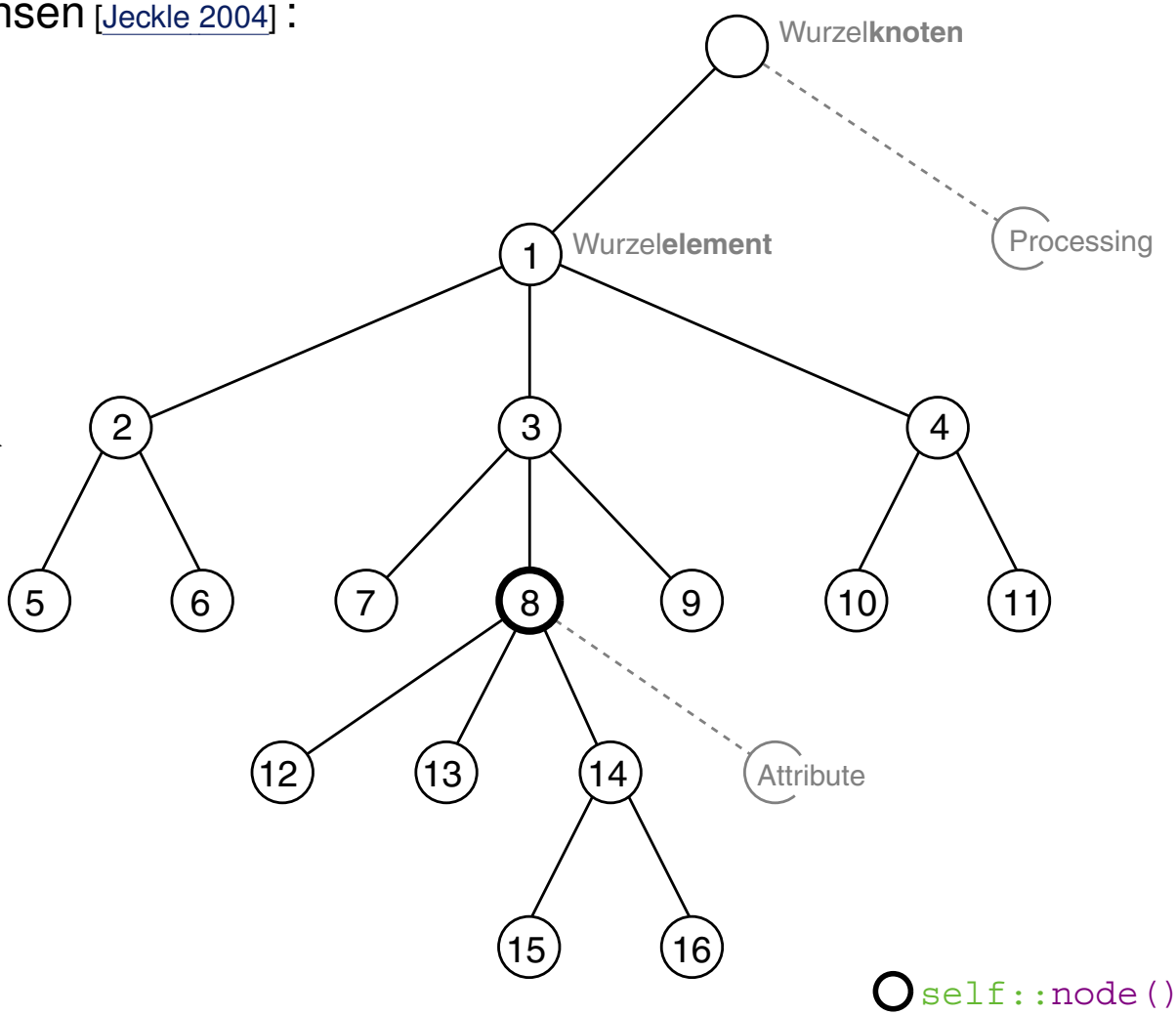
   Beispiele: Test auf den Typ „Kindknoten“ oder eine Position bzw. Index

Die XSL-Familie

XPath-Lokalisierungspfade (Fortsetzung)

Illustration wichtiger Achsen [Jeckle 2004] :

```
<?xml version="1.0">
<node1>
  <node2>
    <node5/>
    <node6/>
  </node2>
  <node3>
    <node7/>
    <node8 Attribute="42">
      <node12/>
      <node13/>
      <node14>
        <node15/>
        <node16/>
      </node14>
    </node8>
    <node9/>
  </node3>
  <node4>
    <node10/>
    <node11/>
  </node4>
</node1>
```

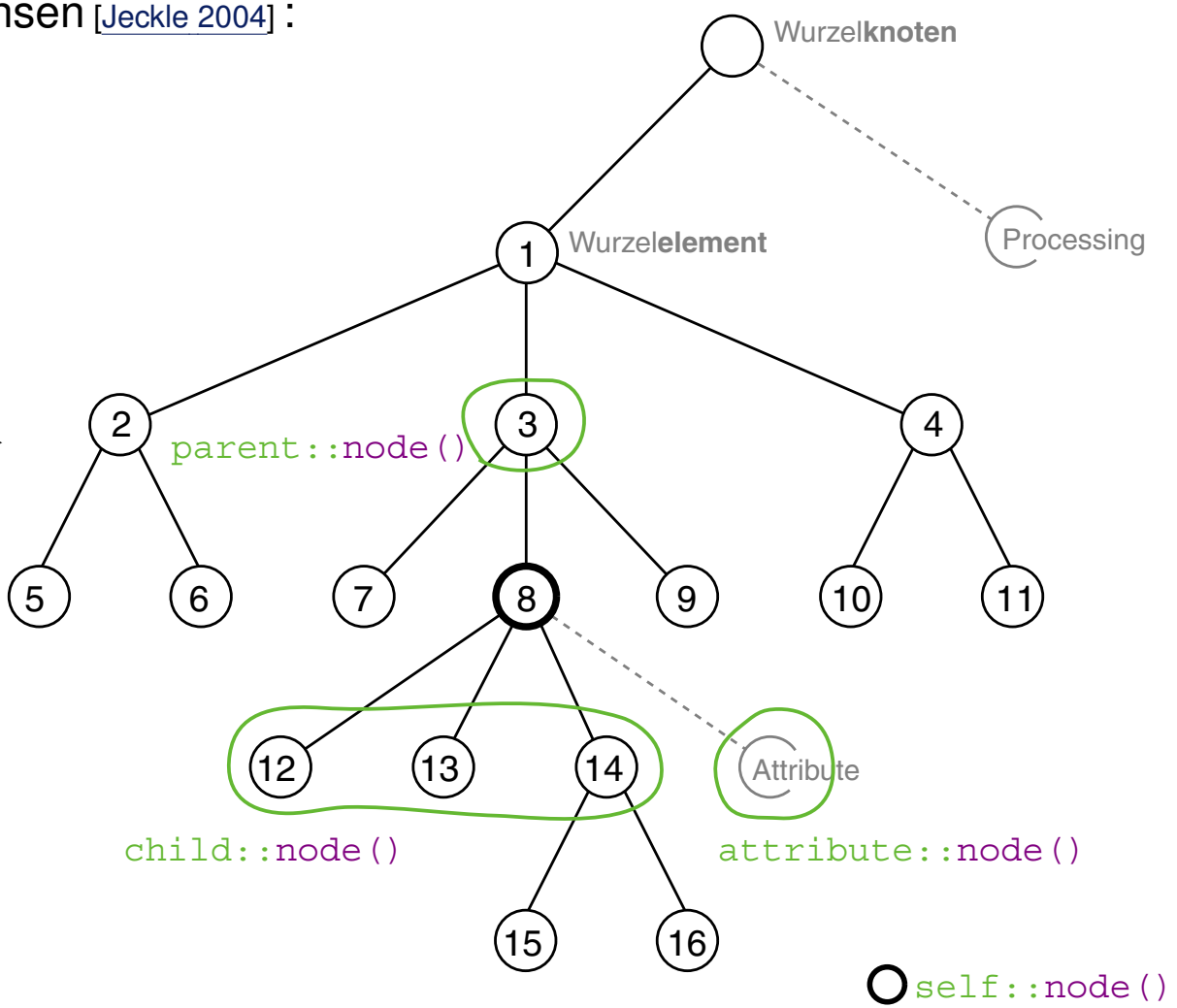


Die XSL-Familie

XPath-Lokalisierungspfade (Fortsetzung)

Illustration wichtiger Achsen [Jeckle 2004]:

```
<?xml version="1.0">
<node1>
  <node2>
    <node5/>
    <node6/>
  </node2>
  <node3>
    <node7/>
    <node8 Attribute="42">
      <node12/>
      <node13/>
      <node14>
        <node15/>
        <node16/>
      </node14>
    </node8>
    <node9/>
  </node3>
  <node4>
    <node10/>
    <node11/>
  </node4>
</node1>
```

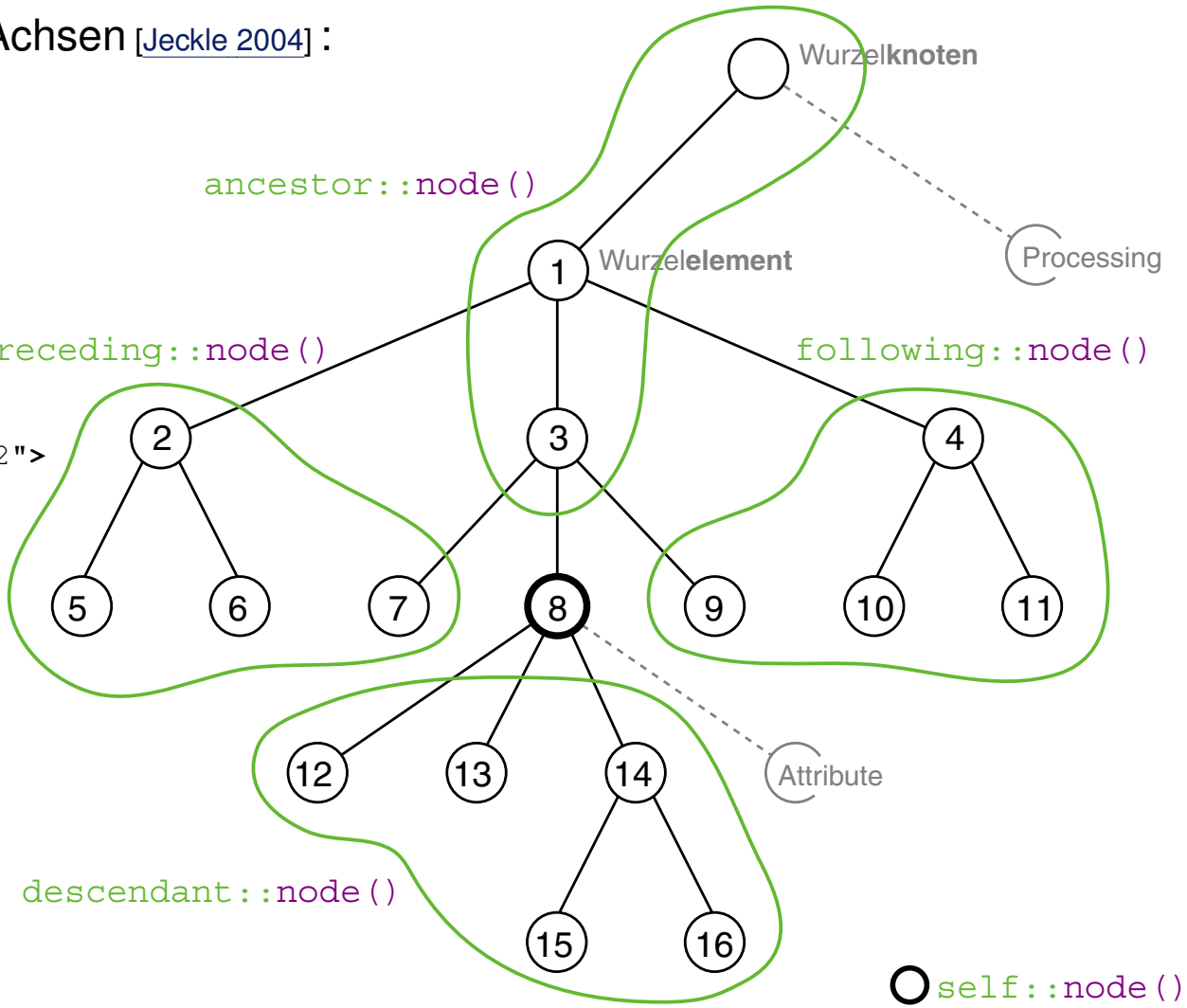


Die XSL-Familie

XPath-Lokalisierungspfade (Fortsetzung)

Illustration wichtiger Achsen [Jeckle 2004]:

```
<?xml version="1.0">
<node1>
  <node2>
    <node5/>
    <node6/>
  </node2>
  <node3>
    <node7/>
    <node8 Attribute="42">
      <node12/>
      <node13/>
      <node14>
        <node15/>
        <node16/>
      </node14>
    </node8>
    <node9/>
  </node3>
  <node4>
    <node10/>
    <node11/>
  </node4>
</node1>
```



Die XSL-Familie

XPath-Lokalisierungspfade (Fortsetzung)

- Schreibweisen häufig verwendeter Lokalisierungsschritte:

kurz	lang	Semantik
`.`	`self::node()`	Kontextknoten
`..`	`parent::node()`	Elternknoten des Kontextknotens
`//`	`/descendant-or-self::node()/`	Kontextknoten einschließlich aller seiner Nachkommen
`@*`	`attribute::*`	alle Attributknoten

- Wildcards für Knotentests:

`node()`	Knoten jedes Typs
`*`	Element- oder Attributknoten (achensabhängig)
`text()`	Textknoten
`comment()`	Kommentarknoten
`processing-instruction()`	Verarbeitungsanweisungsknoten

- Notation von alternativen Lokalisierungspfaden:

Pfad_1 | *Pfad_2* | ... | *Pfad_n*

Die XSL-Familie

XPath-Lokalisierungspfade (Fortsetzung)

- Schreibweisen häufig verwendeter Lokalisierungsschritte:

kurz	lang	Semantik
`.`	`self::node()`	Kontextknoten
`..`	`parent::node()`	Elternknoten des Kontextknotens
`//`	`/descendant-or-self::node()/`	Kontextknoten einschließlich aller seiner Nachkommen
`@*`	`attribute::*`	alle Attributknoten

- Wildcards für Knotentests:

`node()`	Knoten jedes Typs
`*`	Element- oder Attributknoten (achensabhängig)
`text()`	Textknoten
`comment()`	Kommentarknoten
`processing-instruction()`	Verarbeitungsanweisungsknoten

- Notation von alternativen Lokalisierungspfaden:

  Pfad_1 | *Pfad_2* | ... | *Pfad_n*

Die XSL-Familie

XPath-Lokalisierungspfade (Fortsetzung)

- Schreibweisen häufig verwendeter Lokalisierungsschritte:

kurz	lang	Semantik
`.`	`self::node()`	Kontextknoten
`..`	`parent::node()`	Elternknoten des Kontextknotens
`//`	`/descendant-or-self::node()/`	Kontextknoten einschließlich aller seiner Nachkommen
`@*`	`attribute::*`	alle Attributknoten

- Wildcards für Knotentests:

`node()`	Knoten jedes Typs
`*`	Element- oder Attributknoten (achensabhängig)
`text()`	Textknoten
`comment()`	Kommentarknoten
`processing-instruction()`	Verarbeitungsanweisungsknoten

- Notation von alternativen Lokalisierungspfaden:

  Pfad_1 | *Pfad_2* | ... | *Pfad_n*

Die XSL-Familie

XPath-Lokalisierungspfade (Fortsetzung)

```
<?xml version="1.0" ?>
<?xml-stylesheet type="text/xsl" ...?>

<personen>
  <person>
    <name>
      <vorname>Alan</vorname>
      <nachname>Turing</nachname>
    </name>
    <geburtstag>23. Juni 1912</geburtstag>
    <beruf>Mathematiker</beruf>
    <beruf>Informatiker</beruf>
  </person>

  <person>
    <name>
      <vorname>Judea</vorname>
      <nachname>Pearl</nachname>
    </name>
    <geburtstag>unknown</geburtstag>
    <beruf>Informatiker</beruf>
  </person>
</personen>
```

Beispiele für Lokalisierungspfade:

(a) `//person/name/descendant::*`

Die XSL-Familie

XPath-Lokalisierungspfade (Fortsetzung)

```
<?xml version="1.0" ?>
<?xml-stylesheet type="text/xsl" ...?>

<personen>
  <person>
    <name>
      <vorname>Alan</vorname>
      <nachname>Turing</nachname>
    </name>
    <geburtstag>23. Juni 1912</geburtstag>
    <beruf>Mathematiker</beruf>
    <beruf>Informatiker</beruf>
  </person>

  <person>
    <name>
      <vorname>Judea</vorname>
      <nachname>Pearl</nachname>
    </name>
    <geburtstag>unknown</geburtstag>
    <beruf>Informatiker</beruf>
  </person>
</personen>
```

Beispiele für Lokalisierungspfade:

(a) `//person/name/descendant::*`

Die XSL-Familie

XPath-Lokalisierungspfade (Fortsetzung)

```
<?xml version="1.0" ?>
<?xml-stylesheet type="text/xsl" ...?>

<personen>
  <person>
    <name>
      <vorname>Alan</vorname>
      <nachname>Turing</nachname>
    </name>
    <geburtstag>23. Juni 1912</geburtstag>
    <beruf>Mathematiker</beruf>
    <beruf>Informatiker</beruf>
  </person>

  <person>
    <name>
      <vorname>Judea</vorname>
      <nachname>Pearl</nachname>
    </name>
    <geburtstag>unknown</geburtstag>
    <beruf>Informatiker</beruf>
  </person>
</personen>
```

Beispiele für Lokalisierungspfade:

(a) `//person/name/descendant::*`

(b) `//geburtstag/parent::*/name`

Die XSL-Familie

XPath-Lokalisierungspfade (Fortsetzung)

```
<?xml version="1.0" ?>
<?xml-stylesheet type="text/xsl" ...?>

<personen>
  <person>
    <name>
      <vorname>Alan</vorname>
      <nachname>Turing</nachname>
    </name>
    <geburtstag>23. Juni 1912</geburtstag>
    <beruf>Mathematiker</beruf>
    <beruf>Informatiker</beruf>
  </person>

  <person>
    <name>
      <vorname>Judea</vorname>
      <nachname>Pearl</nachname>
    </name>
    <geburtstag>unknown</geburtstag>
    <beruf>Informatiker</beruf>
  </person>
</personen>
```

Beispiele für Lokalisierungspfade:

(a) `//person/name/descendant::*`

(b) `//geburtstag/parent::*/name`

Die XSL-Familie

XPath-Lokalisierungspfade (Fortsetzung)

```
<?xml version="1.0" ?>
<?xml-stylesheet type="text/xsl" ...?>

<personen>
  <person>
    <name>
      <vorname>Alan</vorname>
      <nachname>Turing</nachname>
    </name>
    <geburtstag>23. Juni 1912</geburtstag>
    <beruf>Mathematiker</beruf>
    <beruf>Informatiker</beruf>
  </person>

  <person>
    <name>
      <vorname>Judea</vorname>
      <nachname>Pearl</nachname>
    </name>
    <geburtstag>unknown</geburtstag>
    <beruf>Informatiker</beruf>
  </person>
</personen>
```

Beispiele für Lokalisierungspfade:

(a) `//person/name/descendant::*`

(b) `//geburtstag/parent::*/name`

(c) `/personen/child::name`

Die XSL-Familie

XPath-Lokalisierungspfade (Fortsetzung)

```
<?xml version="1.0" ?>
<?xml-stylesheet type="text/xsl" ...?>

<personen>
  <person>
    <name>
      <vorname>Alan</vorname>
      <nachname>Turing</nachname>
    </name>
    <geburtstag>23. Juni 1912</geburtstag>
    <beruf>Mathematiker</beruf>
    <beruf>Informatiker</beruf>
  </person>

  <person>
    <name>
      <vorname>Judea</vorname>
      <nachname>Pearl</nachname>
    </name>
    <geburtstag>unknown</geburtstag>
    <beruf>Informatiker</beruf>
  </person>
</personen>
```

Beispiele für Lokalisierungspfade:

(a) `//person/name/descendant::*`

(b) `//geburtstag/parent::*/name`

(c) `/personen/child::name`

Die XSL-Familie

XPath-Lokalisierungspfade (Fortsetzung)

```
<?xml version="1.0" ?>
<?xml-stylesheet type="text/xsl" ...?>

<personen>
  <person>
    <name>
      <vorname>Alan</vorname>
      <nachname>Turing</nachname>
    </name>
    <geburtstag>23. Juni 1912</geburtstag>
    <beruf>Mathematiker</beruf>
    <beruf>Informatiker</beruf>
  </person>

  <person>
    <name>
      <vorname>Judea</vorname>
      <nachname>Pearl</nachname>
    </name>
    <geburtstag>unknown</geburtstag>
    <beruf>Informatiker</beruf>
  </person>
</personen>
```

Beispiele für Lokalisierungspfade:

(a) `//person/name/descendant::*`

(b) `//geburtstag/parent::*/name`

(c) `/personen/child::name`

(d) `/personen/descendant::name`

Die XSL-Familie

XPath-Lokalisierungspfade (Fortsetzung)

```
<?xml version="1.0" ?>
<?xml-stylesheet type="text/xsl" ...?>

<personen>
  <person>
    <name>
      <vorname>Alan</vorname>
      <nachname>Turing</nachname>
    </name>
    <geburtstag>23. Juni 1912</geburtstag>
    <beruf>Mathematiker</beruf>
    <beruf>Informatiker</beruf>
  </person>

  <person>
    <name>
      <vorname>Judea</vorname>
      <nachname>Pearl</nachname>
    </name>
    <geburtstag>unknown</geburtstag>
    <beruf>Informatiker</beruf>
  </person>
</personen>
```

Beispiele für Lokalisierungspfade:

(a) `//person/name/descendant::*`

(b) `//geburtstag/parent::*/name`

(c) `/personen/child::name`

(d) `/personen/descendant::name`

Die XSL-Familie

XPath-Lokalisierungspfade (Fortsetzung)

```
<?xml version="1.0" ?>
<?xml-stylesheet type="text/xsl" ...?>

<personen>
  <person>
    <name>
      <vorname>Alan</vorname>
      <nachname>Turing</nachname>
    </name>
    <geburtstag>23. Juni 1912</geburtstag>
    <beruf>Mathematiker</beruf>
    <beruf>Informatiker</beruf>
  </person>

  <person>
    <name>
      <vorname>Judea</vorname>
      <nachname>Pearl</nachname>
    </name>
    <geburtstag>unknown</geburtstag>
    <beruf>Informatiker</beruf>
  </person>
</personen>
```

Beispiele für Lokalisierungspfade:

(a) `//person/name/descendant::*`

(b) `//geburtstag/parent::*/name`

(c) `/personen/child::name`

(d) `/personen/descendant::name`

(e) `//person[geburtstag!='unknown']`

Die XSL-Familie

XPath-Lokalisierungspfade (Fortsetzung)

```
<?xml version="1.0" ?>
<?xml-stylesheet type="text/xsl" ...?>

<personen>
  <person>
    <name>
      <vorname>Alan</vorname>
      <nachname>Turing</nachname>
    </name>
    <geburtstag>23. Juni 1912</geburtstag>
    <beruf>Mathematiker</beruf>
    <beruf>Informatiker</beruf>
  </person>

  <person>
    <name>
      <vorname>Judea</vorname>
      <nachname>Pearl</nachname>
    </name>
    <geburtstag>unknown</geburtstag>
    <beruf>Informatiker</beruf>
  </person>
</personen>
```

Beispiele für Lokalisierungspfade:

(a) `//person/name/descendant::*`

(b) `//geburtstag/parent::*/name`

(c) `/personen/child::name`

(d) `/personen/descendant::name`

(e) `//person[geburtstag!='unknown']`

Die XSL-Familie

XPath-Lokalisierungspfade (Fortsetzung)

Algorithmus zur Auswertung eines Lokalisierungspfades:

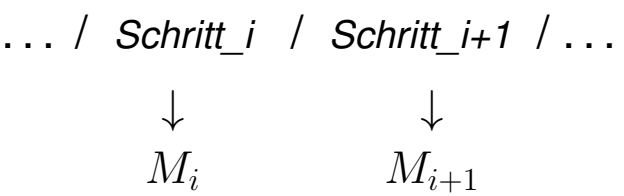
$$\ldots \;/\; \textit{Schritt\_i} \;/\; \textit{Schritt\_i+1} \;/\; \ldots$$
$$\downarrow \qquad\qquad \downarrow$$
$$M_i \qquad\qquad M_{i+1}$$

1. Die Auswertung der Lokalisierungsschritte geschieht von links nach rechts.
2. Jeder Lokalisierungsschritt spezifiziert eine Knotenmenge M.

Die XSL-Familie

XPath-Lokalisierungspfade (Fortsetzung)

Algorithmus zur Auswertung eines Lokalisierungspfades:



1. Die Auswertung der Lokalisierungsschritte geschieht von links nach rechts.
2. Jeder Lokalisierungsschritt spezifiziert eine Knotenmenge M.
3. Jeder Knoten n der Knotenmenge M_i des Lokalisierungsschritts i wird als Kontextknoten hinsichtlich des Lokalisierungsschritts $i+1$ interpretiert und spezifiziert im Lokalisierungsschritt $i+1$ die Knotenmenge $M_{i,n}$.
4. Die Vereinigung der Mengen $M_{i,n}$, $n \in M_i$, bildet die Knotenmenge M_{i+1} des Lokalisierungsschritts $i+1$.

Bemerkungen:

- Jeder der in irgendeinem Schritt spezifizierten Knoten kommt im Laufe der Auswertung in die Rolle des Kontextknotens.

- Vergleich verschiedener Lokalisierungspfade am Beispiel:

  1. Alle `<beruf>`-Elemente:
     `/descendant-or-self::node()/beruf` (bzw. `//beruf`)
     $\equiv$
     `/descendant-or-self::beruf`

  2. Von jedem Elementknoten das jeweils zweite `<beruf>`-Kindelement:
     `/descendant-or-self::node()/beruf[2]` (bzw. `//beruf[2]`)
     $\not\equiv$
     `/descendant-or-self::beruf[2]`
     (= das zweite `<beruf>`-Element aus dem Dokument)

Die XSL-Familie

Lokalisierungspfade (Fortsetzung)

Knoten

```
1
2  <?xml version="1.0" ?>
3  <?xml-stylesheet type="text/xsl" ...?>

4  <personen>
5    <person>
6      <name>
7        <vorname>Alan</vorname>
8        <nachname>Turing</nachname>
6      </name>
9      <geburtstag>23. Juni 1912</geburtstag>
10     <beruf>Mathematiker</beruf>
11     <beruf>Informatiker</beruf>
5    </person>

12   <person>
13     <name>
14       <vorname>Judea</vorname>
15       <nachname>Pearl</nachname>
13     </name>
16     <geburtstag>unknown</geburtstag>
17     <beruf>Informatiker</beruf>
12   </person>
4  </personen>
```

Illustration des Algorithmus:

```
//person/name/descendant::*
//person/name/descendant::*
```

$M_1 = \{1\}$

$M_2 = \{1, \ldots, 17\}$

$M_3 = \{5, 12\}$

$M_4 = \{6, 13\}$

$M_5 = \{7, 8, 14, 15\}$

Die XSL-Familie

Lokalisierungspfade (Fortsetzung)

Knoten

```
1
2  <?xml version="1.0" ?>
3  <?xml-stylesheet type="text/xsl" ...?>

4  <personen>
5    <person>
6      <name>
7        <vorname>Alan</vorname>
8        <nachname>Turing</nachname>
6      </name>
9      <geburtstag>23. Juni 1912</geburtstag>
10     <beruf>Mathematiker</beruf>
11     <beruf>Informatiker</beruf>
5    </person>

12   <person>
13     <name>
14       <vorname>Judea</vorname>
15       <nachname>Pearl</nachname>
13     </name>
16     <geburtstag>unknown</geburtstag>
17     <beruf>Informatiker</beruf>
12   </person>
4  </personen>
```

Illustration des Algorithmus:

`//person/name/descendant::*`

`/`/person/name/descendant::*

$M_1 = \{1\}$

$M_2 = \{1, \ldots, 17\}$

$M_3 = \{5, 12\}$

$M_4 = \{6, 13\}$

$M_5 = \{7, 8, 14, 15\}$

Die XSL-Familie

Lokalisierungspfade (Fortsetzung)

Knoten

```
 1
 2 <?xml version="1.0" ?>
 3 <?xml-stylesheet type="text/xsl" ...?>

 4 <personen>
 5   <person>
 6     <name>
 7       <vorname>Alan</vorname>
 8       <nachname>Turing</nachname>
 6     </name>
 9     <geburtstag>23. Juni 1912</geburtstag>
10     <beruf>Mathematiker</beruf>
11     <beruf>Informatiker</beruf>
 5   </person>

12   <person>
13     <name>
14       <vorname>Judea</vorname>
15       <nachname>Pearl</nachname>
13     </name>
16     <geburtstag>unknown</geburtstag>
17     <beruf>Informatiker</beruf>
12   </person>
 4 </personen>
```

Illustration des Algorithmus:

```
//person/name/descendant::*
//person/name/descendant::*
```

$M_1 = \{1\}$

$M_2 = \{1, \ldots, 17\}$

$M_3 = \{5, 12\}$

$M_4 = \{6, 13\}$

$M_5 = \{7, 8, 14, 15\}$

Die XSL-Familie

Lokalisierungspfade (Fortsetzung)

Knoten

```
1
2  <?xml version="1.0" ?>
3  <?xml-stylesheet type="text/xsl" ...?>

4  <personen>
5    <person>
6      <name>
7        <vorname>Alan</vorname>
8        <nachname>Turing</nachname>
6      </name>
9      <geburtstag>23. Juni 1912</geburtstag>
10     <beruf>Mathematiker</beruf>
11     <beruf>Informatiker</beruf>
5    </person>

12   <person>
13     <name>
14       <vorname>Judea</vorname>
15       <nachname>Pearl</nachname>
13     </name>
16     <geburtstag>unknown</geburtstag>
17     <beruf>Informatiker</beruf>
12   </person>
4  </personen>
```

Illustration des Algorithmus:

```
//person/name/descendant::*
//person/name/descendant::*
```

$M_1 = \{1\}$

$M_2 = \{1, \ldots, 17\}$

$M_3 = \{5, 12\}$

$M_4 = \{6, 13\}$

$M_5 = \{7, 8, 14, 15\}$

Die XSL-Familie

Lokalisierungspfade (Fortsetzung)

Knoten

```
1
2  <?xml version="1.0" ?>
3  <?xml-stylesheet type="text/xsl" ...?>

4  <personen>
5    <person>
6      <name>
7        <vorname>Alan</vorname>
8        <nachname>Turing</nachname>
6      </name>
9      <geburtstag>23. Juni 1912</geburtstag>
10     <beruf>Mathematiker</beruf>
11     <beruf>Informatiker</beruf>
5    </person>

12   <person>
13     <name>
14       <vorname>Judea</vorname>
15       <nachname>Pearl</nachname>
13     </name>
16     <geburtstag>unknown</geburtstag>
17     <beruf>Informatiker</beruf>
12   </person>
4  </personen>
```

Illustration des Algorithmus:

```
//person/name/descendant::*
//person/name/descendant::*
```

$M_1 = \{1\}$

$M_2 = \{1, \ldots, 17\}$

$M_3 = \{5, 12\}$

$M_4 = \{6, 13\}$

$M_5 = \{7, 8, 14, 15\}$

Die XSL-Familie

Lokalisierungspfade (Fortsetzung)

Knoten

```
1
2  <?xml version="1.0" ?>
3  <?xml-stylesheet type="text/xsl" ...?>

4  <personen>
5    <person>
6      <name>
7        <vorname>Alan</vorname>
8        <nachname>Turing</nachname>
6      </name>
9      <geburtstag>23. Juni 1912</geburtstag>
10     <beruf>Mathematiker</beruf>
11     <beruf>Informatiker</beruf>
5    </person>

12   <person>
13     <name>
14       <vorname>Judea</vorname>
15       <nachname>Pearl</nachname>
13     </name>
16     <geburtstag>unknown</geburtstag>
17     <beruf>Informatiker</beruf>
12   </person>
4  </personen>
```

Illustration des Algorithmus:

```
//person/name/descendant::*
//person/name/descendant::*
```

$$M_1 = \{1\}$$

$$M_2 = \{1, \ldots, 17\}$$

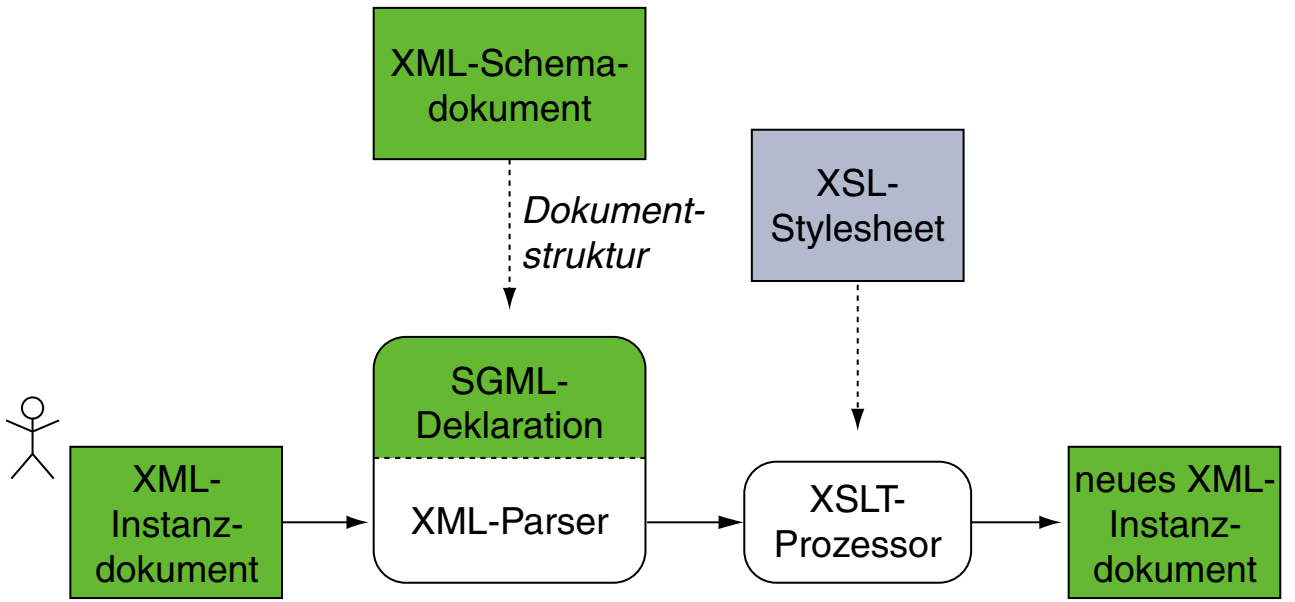
$$M_3 = \{5, 12\}$$

$$M_4 = \{6, 13\}$$

$$M_5 = \{7, 8, 14, 15\}$$

Die XSL-Familie

XSL Transformation



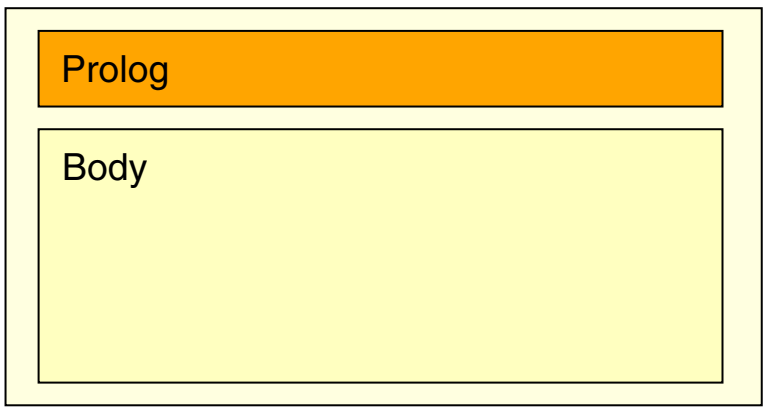
XSLT ist eine Turing-vollständige Programmiersprache zur Transformation wohlgeformter XML-Dokumente in andere XML-Dokumente. Ein XSLT-Programm liegt üblicherweise als XSL-Stylesheet vor.

Die Transformation umfasst die Selektion von Teilen des Eingabedokuments, deren Umordnung sowie die Generierung neuer Inhalte aus den bestehenden.

Die XSL-Familie

Aufbau XSL-Stylesheet

XSL-Stylesheets sind XML-Dokumente:



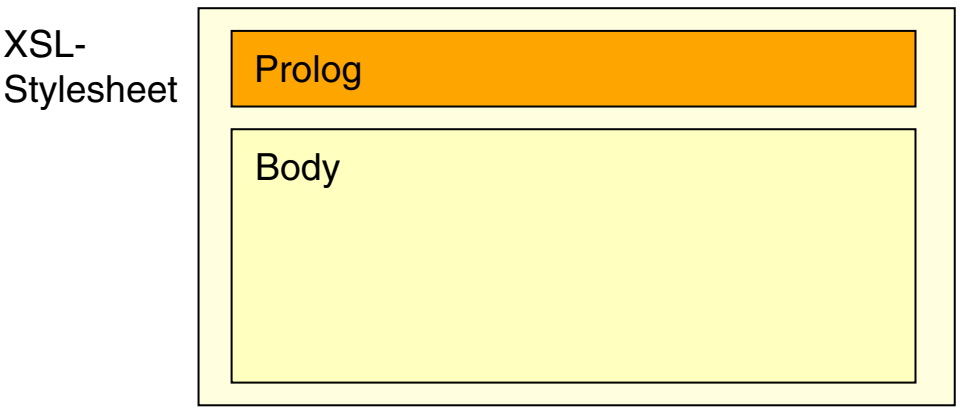

```
<?xml version="1.0"?>

<xsl:stylesheet xmlns:xsl=...>
  <xsl:template match=...>
      ...
  </xsl:template>
  ...
  <xsl:template match=...>
      ...
  </xsl:template>
</xsl:stylesheet>
```

- Wurzelelement jedes XSL-Schemas ist das Element `<xsl:stylesheet>` oder synonym `<xsl:transform>`.
- Die Kindelemente von `<xsl:stylesheet>` bzw. `<xsl:transform>` definieren Transformationsvorschriften in Form von Template-Regeln.
- Vergleiche hierzu die XML-Dokumentstruktur und die XML-Schema-Dokumentstruktur.

Die XSL-Familie

Aufbau XSL-Stylesheet

XSL-Stylesheets sind XML-Dokumente:




```
<?xml version="1.0"?>

<xsl:stylesheet xmlns:xsl=...>
  <xsl:template match=...>
      ...
  </xsl:template>
  ...
  <xsl:template match=...>
      ...
  </xsl:template>
</xsl:stylesheet>
```

- Wurzelelement jedes XSL-Schemas ist das Element `<xsl:stylesheet>` oder synonym `<xsl:transform>`.
- Die Kindelemente von `<xsl:stylesheet>` bzw. `<xsl:transform>` definieren Transformationsvorschriften in Form von Template-Regeln.
- Vergleiche hierzu die XML-Dokumentstruktur und die XML-Schema-Dokumentstruktur.

Bemerkungen:

- Das XSLT-Vokabular enthält die Namen für Elemente, die zur Erstellung von XSL-Stylesheets zur Verfügung stehen:

```
- xsl:accept                - xsl:assert              - xsl:character-map
- xsl:accumulator           - xsl:attribute           - xsl:choose>
- xsl:accumulator-rule      - xsl:attribute-set       - xsl:comment
- xsl:analyze-string        - xsl:break               - xsl:context-item
- xsl:apply-imports         - xsl:call-template         ⋮
- xsl:apply-templates       - xsl:catch
```

  Der zugehörige Namensraum heißt http://www.w3.org/1999/XSL/Transform. Das übliche Präfix bei der Namensraumdeklaration ist `xsl:`, es kann aber beliebig gewählt werden. Wird der offizielle Namensraum gebunden, ist auch das Attribut `version="1.0"` anzugeben.

- Die Dateiendung einer XSL-Stylesheet-Datei ist `.xsl`.

- Aufbau einer realen Turingmaschine. [youtube]

Die XSL-Familie

XML-Beispieldokument

```
<?xml version="1.0" ?>
<?xml-stylesheet type="text/xsl" href="personen2html.xsl"?>
<personen>
  <person>
    <name>
      <vorname>Alan</vorname>
      <nachname>Turing</nachname>
    </name>
    <geburtstag>23. Juni 1912</geburtstag>
    <beruf>Mathematiker</beruf>
    <beruf>Informatiker</beruf>
  </person>

  <person>
    <name>
      <vorname>Judea</vorname>
      <nachname>Pearl</nachname>
    </name>
    <geburtstag>unknown</geburtstag>
    <beruf>Informatiker</beruf>
  </person>
</personen>
```

Bemerkungen:

- Die Verknüpfung von XML-Dokument und XSL-Stylesheet kann explizit, in Form von Parametern für den XSLT-Prozessor, sowie auch implizit geschehen:

  Die Zeile `<?xml-stylesheet type="text/xsl" href="..."?>` im Prolog eines XML-Dokuments deklariert ein Stylesheet. Vergleiche hierzu die Stylesheet-Deklaration in HTML-Dokumenten.

- Beispiel: Verknüpfung von personen.xml (1) mit einem Stylesheet (2) zu einem HTML-Dokument (3). Der Quelltext (→ Seitenquelltext im Browser) der Dateien (1) und (3) unterscheidet sich nur um die Zeile, mit der das Stylesheet eingebunden wird.

- Aufruf des XSLT-Prozessors Xalan-J über die Kommandozeile:

  ```
  java org.apache.xalan.xslt.Process -in personen.xml -xsl tiny.xsl
  ```

  Hierfür muss die Xalan-Bibliothek heruntergeladen und der Ort der Bibliothek im Classpath spezifiziert sein. Alternativ der Aufruf mit expliziter Angabe der Xalan-Bibliothek:

  ```
  java -cp <path>/xalan.jar ...
  ```

- Aufruf des XSLT-Prozessor via Microsoft Visual Studio: Dokumentation

Die XSL-Familie

Elemente eines XSL-Stylesheets

Das einfachste (= leere) Stylesheet:

```
<?xml version="1.0"?>
<xsl:stylesheet version="1.0" xmlns:xsl="http://www.w3.org/1999/XSL/Transform">

</xsl:stylesheet>
```

Die XSL-Familie

Elemente eines XSL-Stylesheets

Das einfachste (= leere) Stylesheet:

```
<?xml version="1.0"?>
<xsl:stylesheet version="1.0" xmlns:xsl="http://www.w3.org/1999/XSL/Transform">

</xsl:stylesheet>
```

Angewandt auf das Beispieldokument:

```
<?xml version="1.0"?>

     Alan
     Turing

   23. Juni 1912
   Mathematiker
   Informatiker

     Judea
     Pearl

   unknown
   Informatiker
```

```
<?xml version="1.0" ?>
<?xml-stylesheet type="text/xsl" . . . ?>
<personen>
  <person>
    <name>
      <vorname>Alan</vorname>
      <nachname>Turing</nachname>
    </name>
    <geburtstag>23. Juni 1912</geburtstag>
    <beruf>Mathematiker</beruf>
    <beruf>Informatiker</beruf>
  </person>

  <person>
    <name>
      <vorname>Judea</vorname>
      <nachname>Pearl</nachname>
    </name>
    <geburtstag>unknown</geburtstag>
    <beruf>Informatiker</beruf>
  </person>
</personen>
```

Bemerkungen:

- Das leere Stylesheet in dem Bespiel enthält keine matchende Template-Regel. Die Ausgabe entsteht, weil in einer solchen Situation vom XSLT-Prozessor das Built-in-Template zur Ausgabe von Text- und Attributknoten angewandt wird.

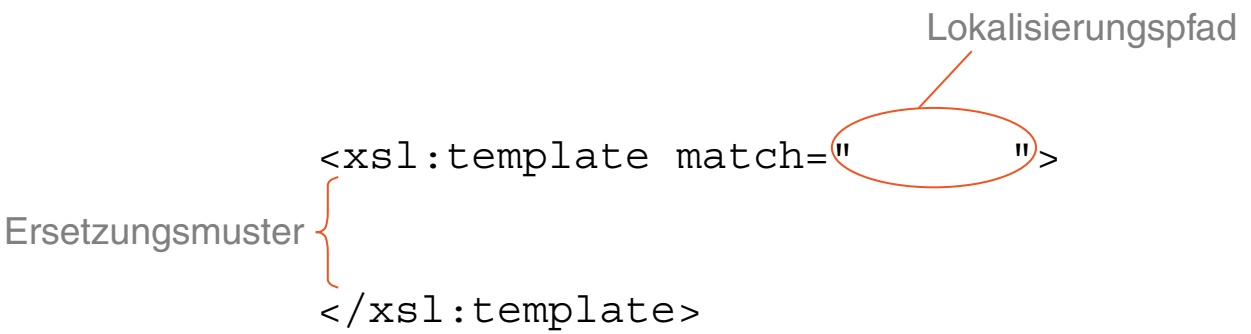
  In dem Beispiel handelt es sich um die Textknoten `Alan, Turing, 23. Juni 1912, Mathematiker, Informatiker, Judea, Pearl, unknown, Informatiker`. Diese Knoten entsprechen den *Character Information Item* des XML Information Sets.

- Diejenigen Konstrukte eines XML-Dokuments, die nicht zu einem der sieben Knotentypen des XPath-Modells gehören, werden unverändert übernommen. Hierzu zählt u.a. die `<?xml ...?>`-Deklaration.

Die XSL-Familie

Elemente eines XSL-Stylesheets (Fortsetzung)

Wichtigstes Stylesheet-Element ist die Template-Regel (*Template*):

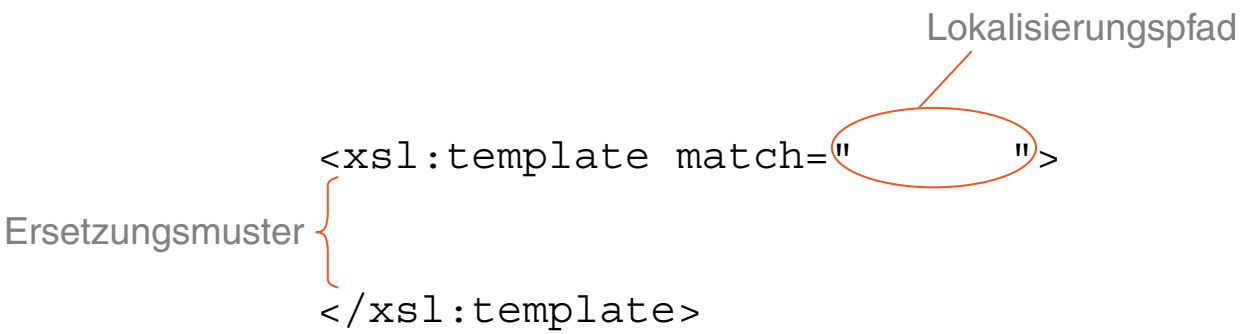


- Der Lokalisierungspfad des `match`-Attributs spezifiziert – ausgehend von dem Kontextknoten – eine Knotenmenge M.
- Wird während der Verarbeitung eines XML-Dokuments ein Knoten in M erreicht, dann matched die Template-Regel diesen Knoten.

Die XSL-Familie

Elemente eines XSL-Stylesheets (Fortsetzung)

Wichtigstes Stylesheet-Element ist die Template-Regel (*Template*):



- Der Lokalisierungspfad des `match`-Attributs spezifiziert – ausgehend von dem Kontextknoten – eine Knotenmenge M.
- Wird während der Verarbeitung eines XML-Dokuments ein Knoten in M erreicht, dann matched die Template-Regel diesen Knoten.
- Matched eine Template-Regel einen Knoten n, behandelt das Ersetzungsmuster den gesamten Teilbaum des XML-Dokuments, der Knoten n als Wurzel hat. Dieser Teilbaum gilt als abgearbeitet.

Bemerkungen:

- Der Wert des `match`-Attributes im `<xsl:template>`-Element ist ein Lokalisierungspfad in eingeschränkter XPath-Syntax.
- Um eine bestimmte Knotenmenge M zu spezifizieren, für deren Elemente eine Template-Regel matched, sind alternative Pfadangaben möglich. Beispielsweise spezifizieren die Ausdrücke `match`="`//`*Elementname*" und `match`="*Elementname*" dieselbe Knotenmenge.

  D.h., ein relativer Lokalisierungspfad des `<xsl:template>`-Elements kann wie der entsprechende absolute durch "`//`" eingeleitete Lokalisierungspfad aufgefasst werden – und umgekehrt.

Die XSL-Familie

Elemente eines XSL-Stylesheets (Fortsetzung)

Stylesheet mit konstanter Textausgabe:

```
<?xml version="1.0"?>
<xsl:stylesheet version="1.0" xmlns:xsl="http://www.w3.org/...">

  <xsl:template match="person">
    <xsl:text>Person found!</xsl:text>
  </xsl:template>

</xsl:stylesheet>
```

Angewandt auf das Beispieldokument:

```
<?xml version="1.0" ?>
<?xml-stylesheet type="text/xsl" ... ?>
<personen>
  <person>
    <name>
      <vorname>Alan</vorname>
      <nachname>Turing</nachname>
    </name>
    <geburtstag>23. Juni 1912</geburtstag>
    <beruf>Mathematiker</beruf>
    <beruf>Informatiker</beruf>
  </person>

  <person>
    <name>
      <vorname>Judea</vorname>
      <nachname>Pearl</nachname>
    </name>
    <geburtstag>unknown</geburtstag>
    <beruf>Informatiker</beruf>
  </person>
</personen>
```

Die XSL-Familie

Elemente eines XSL-Stylesheets (Fortsetzung)

Stylesheet mit konstanter Textausgabe:

```
<?xml version="1.0"?>
<xsl:stylesheet version="1.0" xmlns:xsl="http://www.w3.org/...">

  <xsl:template match="person">
    <xsl:text>Person found!</xsl:text>
  </xsl:template>

</xsl:stylesheet>
```

Angewandt auf das Beispieldokument:

```
Person found!

Person found!
```

```
<?xml version="1.0" ?>
<?xml-stylesheet type="text/xsl" ... ?>
<personen>
  <person>
    <name>
      <vorname>Alan</vorname>
      <nachname>Turing</nachname>
    </name>
    <geburtstag>23. Juni 1912</geburtstag>
    <beruf>Mathematiker</beruf>
    <beruf>Informatiker</beruf>
  </person>

  <person>
    <name>
      <vorname>Judea</vorname>
      <nachname>Pearl</nachname>
    </name>
    <geburtstag>unknown</geburtstag>
    <beruf>Informatiker</beruf>
  </person>
</personen>
```

Die XSL-Familie

Elemente eines XSL-Stylesheets (Fortsetzung)

Stylesheet zum Kopieren der Elemente:

```
<?xml version="1.0"?>
<xsl:stylesheet version="1.0" xmlns:xsl="http://www.w3.org/...">

  <xsl:template match="person">
    <xsl:copy-of select="self::*"/>
  </xsl:template>

</xsl:stylesheet>
```

Angewandt auf das Beispieldokument:

```
<?xml version="1.0" ?>
<?xml-stylesheet type="text/xsl" ... ?>
<personen>
  <person>
    <name>
      <vorname>Alan</vorname>
      <nachname>Turing</nachname>
    </name>
    <geburtstag>23. Juni 1912</geburtstag>
    <beruf>Mathematiker</beruf>
    <beruf>Informatiker</beruf>
  </person>

  <person>
    <name>
      <vorname>Judea</vorname>
      <nachname>Pearl</nachname>
    </name>
    <geburtstag>unknown</geburtstag>
    <beruf>Informatiker</beruf>
  </person>
</personen>
```

Die XSL-Familie

Elemente eines XSL-Stylesheets (Fortsetzung)

Stylesheet zum Kopieren der Elemente:

```
<?xml version="1.0"?>
<xsl:stylesheet version="1.0" xmlns:xsl="http://www.w3.org/...">

  <xsl:template match="person">
    <xsl:copy-of select="self::*"/>
  </xsl:template>

</xsl:stylesheet>
```

Angewandt auf das Beispieldokument:

```
<person>
  <name>
    <vorname>Alan</vorname>
    <nachname>Turing</nachname>
  </name>
  <geburtstag>23. Juni 1912</geburtstag>
  <beruf>Mathematiker</beruf>
  <beruf>Informatiker</beruf>
</person>
...
```

```
<?xml version="1.0" ?>
<?xml-stylesheet type="text/xsl" . . . ?>
<personen>
  <person>
    <name>
      <vorname>Alan</vorname>
      <nachname>Turing</nachname>
    </name>
    <geburtstag>23. Juni 1912</geburtstag>
    <beruf>Mathematiker</beruf>
    <beruf>Informatiker</beruf>
  </person>

  <person>
    <name>
      <vorname>Judea</vorname>
      <nachname>Pearl</nachname>
    </name>
    <geburtstag>unknown</geburtstag>
    <beruf>Informatiker</beruf>
  </person>
</personen>
```

Die XSL-Familie

Elemente eines XSL-Stylesheets (Fortsetzung)

Stylesheet zum Kopieren der Element*inhalte*:

```
<?xml version="1.0"?>
<xsl:stylesheet version="1.0" xmlns:xsl="http://www.w3.org/...">

  <xsl:template match="person">
    <xsl:value-of select="self::*"/>
  </xsl:template>

</xsl:stylesheet>
```

Angewandt auf das Beispieldokument:

```
<?xml version="1.0" ?>
<?xml-stylesheet type="text/xsl" ... ?>
<personen>
  <person>
    <name>
      <vorname>Alan</vorname>
      <nachname>Turing</nachname>
    </name>
    <geburtstag>23. Juni 1912</geburtstag>
    <beruf>Mathematiker</beruf>
    <beruf>Informatiker</beruf>
  </person>

  <person>
    <name>
      <vorname>Judea</vorname>
      <nachname>Pearl</nachname>
    </name>
    <geburtstag>unknown</geburtstag>
    <beruf>Informatiker</beruf>
  </person>
</personen>
```

Die XSL-Familie

Elemente eines XSL-Stylesheets (Fortsetzung)

Stylesheet zum Kopieren der Element*inhalte*:

```
<?xml version="1.0"?>
<xsl:stylesheet version="1.0" xmlns:xsl="http://www.w3.org/...">

  <xsl:template match="person">
    <xsl:value-of select="self::*"/>
  </xsl:template>

</xsl:stylesheet>
```

Angewandt auf das Beispieldokument:

```
    Alan
    Turing

  23. Juni 1912
  Mathematiker
  Informatiker

  ...
```

```
<?xml version="1.0" ?>
<?xml-stylesheet type="text/xsl" . . . ?>
<personen>
  <person>
    <name>
      <vorname>Alan</vorname>
      <nachname>Turing</nachname>
    </name>
    <geburtstag>23. Juni 1912</geburtstag>
    <beruf>Mathematiker</beruf>
    <beruf>Informatiker</beruf>
  </person>

  <person>
    <name>
      <vorname>Judea</vorname>
      <nachname>Pearl</nachname>
    </name>
    <geburtstag>unknown</geburtstag>
    <beruf>Informatiker</beruf>
  </person>
</personen>
```

Die XSL-Familie

Elemente eines XSL-Stylesheets (Fortsetzung)

Stylesheet zur Elementselektion mittels leerer Template-Regeln:

```
<?xml version="1.0"?>
<xsl:stylesheet version="1.0" xmlns:xsl="http://www.w3.org/...">

  <xsl:template match="name">
    <xsl:value-of select="nachname"/>
    <xsl:text>, </xsl:text>
    <xsl:value-of select="vorname"/>
  </xsl:template>

  <xsl:template match="geburtstag"/>
  <xsl:template match="beruf"/>

</xsl:stylesheet>
```

Angewandt auf das Beispieldokument:

```
<?xml version="1.0" ?>
<?xml-stylesheet type="text/xsl" ... ?>
<personen>
  <person>
    <name>
      <vorname>Alan</vorname>
      <nachname>Turing</nachname>
    </name>
    <geburtstag>23. Juni 1912</geburtstag>
    <beruf>Mathematiker</beruf>
    <beruf>Informatiker</beruf>
  </person>

  <person>
    <name>
      <vorname>Judea</vorname>
      <nachname>Pearl</nachname>
    </name>
    <geburtstag>unknown</geburtstag>
    <beruf>Informatiker</beruf>
  </person>
</personen>
```

Die XSL-Familie

Elemente eines XSL-Stylesheets (Fortsetzung)

Stylesheet zur Elementselektion mittels leerer Template-Regeln:

```
<?xml version="1.0"?>
<xsl:stylesheet version="1.0" xmlns:xsl="http://www.w3.org/...">

  <xsl:template match="name">
    <xsl:value-of select="nachname"/>
    <xsl:text>, </xsl:text>
    <xsl:value-of select="vorname"/>
  </xsl:template>

  <xsl:template match="geburtstag"/>
  <xsl:template match="beruf"/>

</xsl:stylesheet>
```

```
<?xml version="1.0" ?>
<?xml-stylesheet type="text/xsl" ... ?>
<personen>
  <person>
    <name>
      <vorname>Alan</vorname>
      <nachname>Turing</nachname>
    </name>
    <geburtstag>23. Juni 1912</geburtstag>
    <beruf>Mathematiker</beruf>
    <beruf>Informatiker</beruf>
  </person>

  <person>
    <name>
      <vorname>Judea</vorname>
      <nachname>Pearl</nachname>
    </name>
    <geburtstag>unknown</geburtstag>
    <beruf>Informatiker</beruf>
  </person>
</personen>
```

Angewandt auf das Beispieldokument:

```
Turing, Alan
Pearl, Judea
```

Vergleiche die Elementselektion durch explizite Verarbeitungssteuerung.

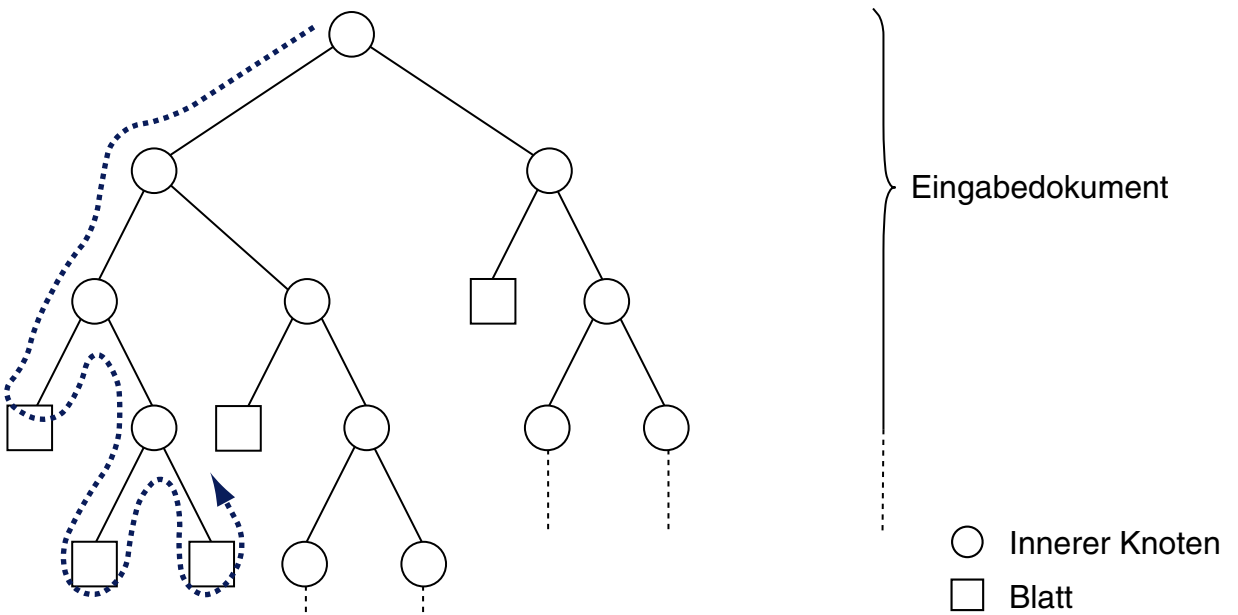
Bemerkungen: (Wiederholung)

- Matched eine Template-Regel einen Knoten im XML-Dokument, so gilt der Knoten einschließlich des zugehörigen Teilbaums als abgearbeitet.
- Mit leeren Template-Regeln kann man Knoten und Teilbäume filtern, die nicht in der Ausgabe erscheinen sollen.
- Matched keine Template-Regel des Stylesheets einen Knoten im XML-Dokument, wird vom XSLT-Prozessor das Built-in-Template zur Ausgabe von Text- und Attributknoten angewandt.

Die XSL-Familie

XSLT-Prozessor: Verarbeitungsstrategie

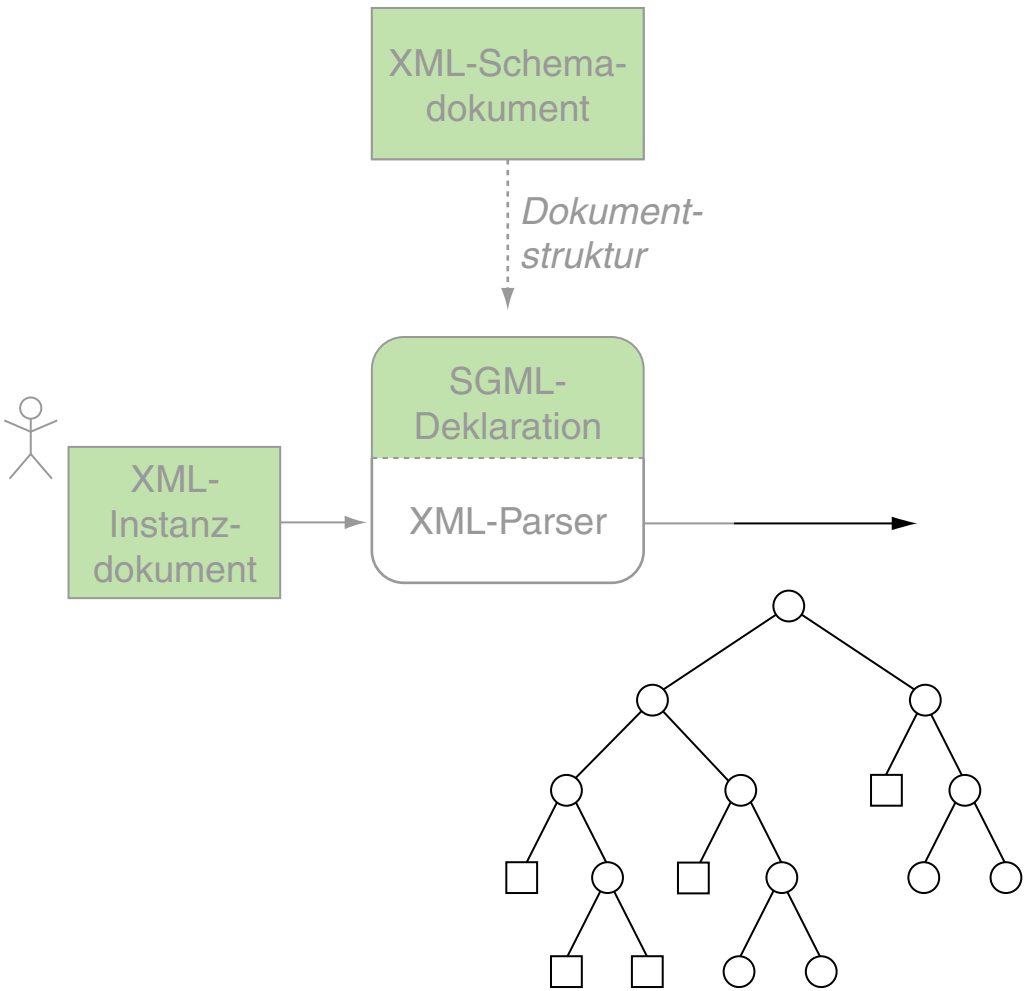
Standardmäßig durchläuft der XSLT-Prozessor den aus dem Eingabedokument erzeugten Baum ausgehend vom Wurzelknoten in Pre-Order-Reihenfolge.



Während des Traversierungsvorgangs wird für jeden besuchten Knoten das speziellste, matchende Template gesucht und angewandt. So transformiert der XSLT-Prozessor einen XML-Quellbaum in einen XML-Zielbaum.

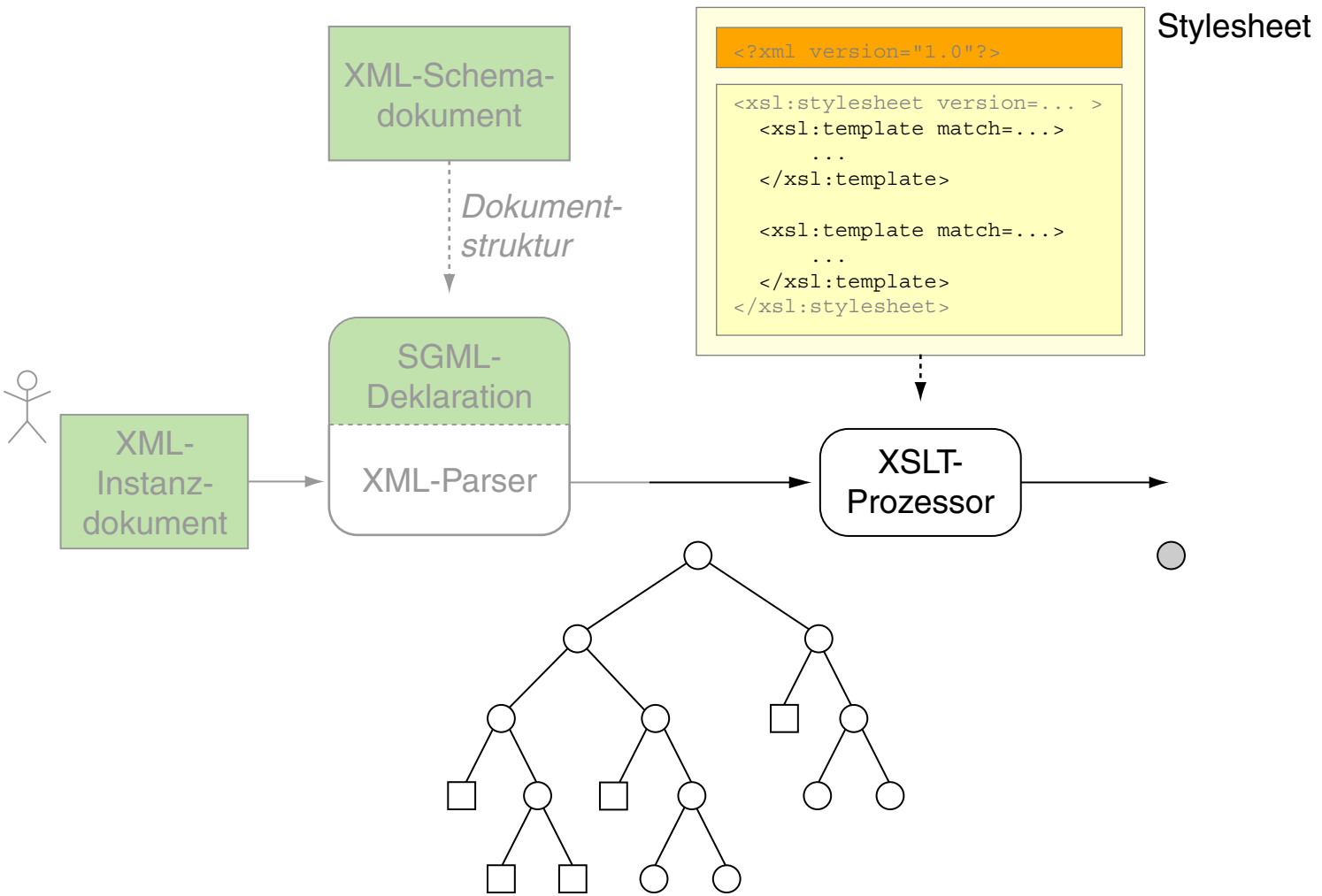
Die XSL-Familie

XSLT-Prozessor: Verarbeitungsstrategie (Fortsetzung)



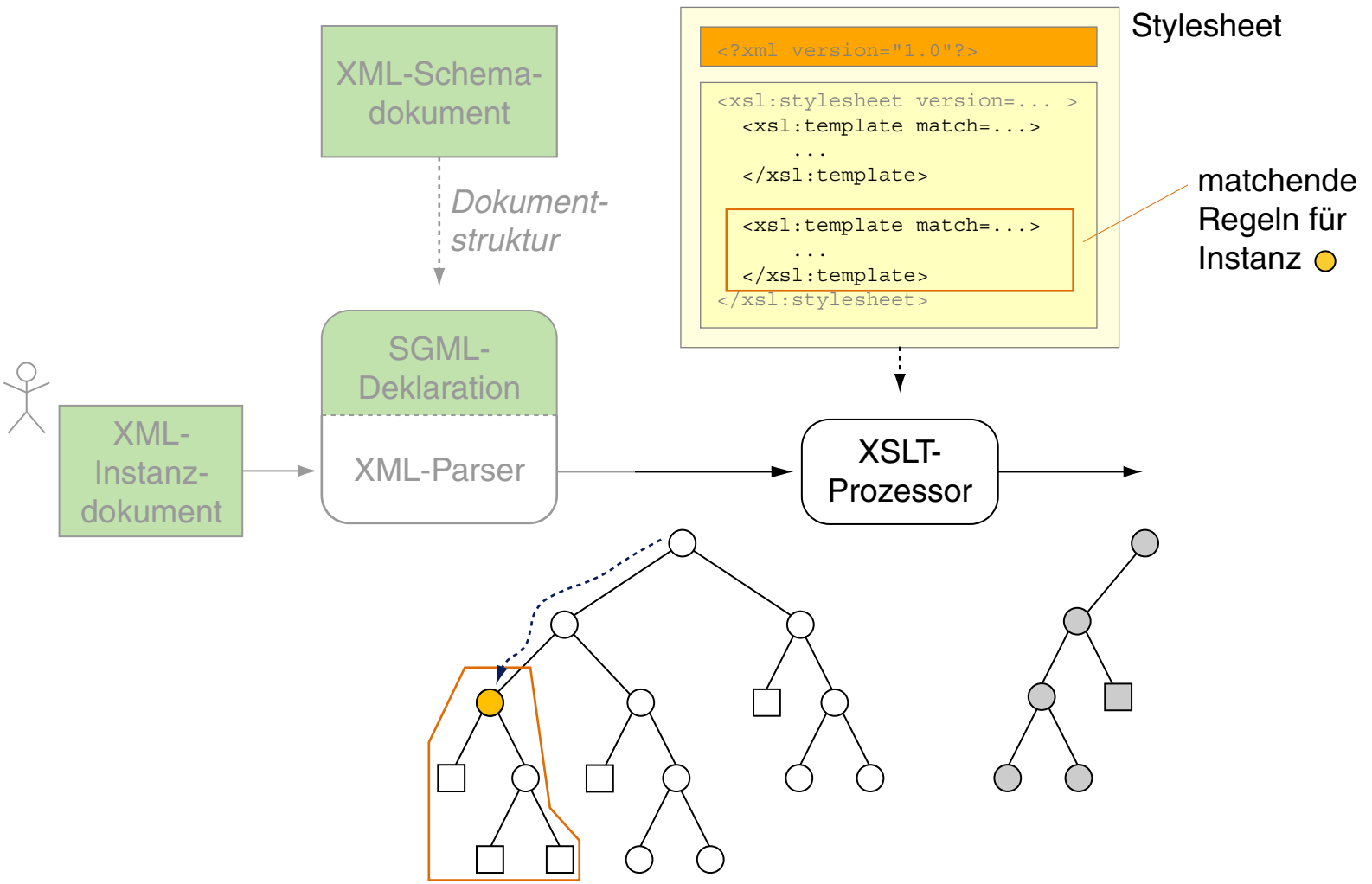
Die XSL-Familie

XSLT-Prozessor: Verarbeitungsstrategie (Fortsetzung)



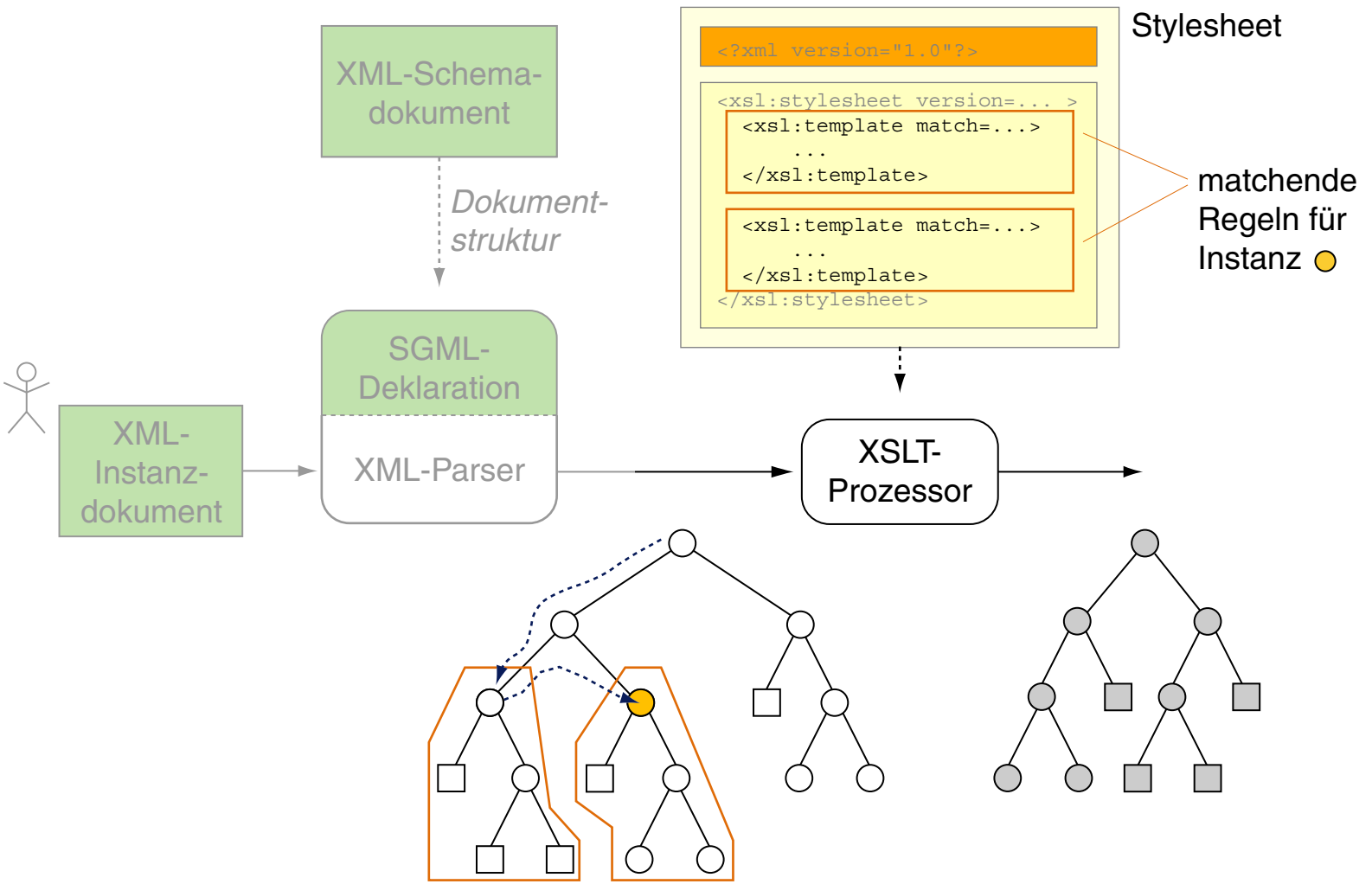
Die XSL-Familie

XSLT-Prozessor: Verarbeitungsstrategie (Fortsetzung)



Die XSL-Familie

XSLT-Prozessor: Verarbeitungsstrategie (Fortsetzung) [WT:III CSS-Verarbeitung]



Bemerkungen:

- Aus Verarbeitungssicht spielt somit die Reihenfolge der Template-Regeln in einem XSL-Stylesheet keine Rolle: die Verarbeitung wird ausschließlich durch die *Reihenfolge der Elemente im Eingabedokument* bestimmt.
- Ein Anwendungskonflikt zur Verarbeitung eines Knoten liegt vor, wenn Lokalisierungspfade von mehreren Template-Regeln diesen Knoten in ihrer spezifizierten Knotenmengen enthalten. In diesem Fall kommt das spezifischste Template zur Anwendung – also das Template, das die kleinste Knotenmenge spezifiziert.

Die XSL-Familie

XSLT-Prozessor: Verarbeitungsstrategie (Fortsetzung)

Stylesheet zur Elementselektion mit expliziter Verarbeitungssteuerung:

```
<?xml version="1.0"?>
<xsl:stylesheet version="1.0" xmlns:xsl="http://www.w3.org/...">

  <xsl:template match="person">
    <xsl:apply-templates select="name"/>
  </xsl:template>

  <xsl:template match="name">
    <xsl:value-of select="nachname"/>
    <xsl:text>, </xsl:text>
    <xsl:value-of select="vorname"/>
  </xsl:template>

</xsl:stylesheet>
```

Angewandt auf das Beispieldokument:

```
<?xml version="1.0" ?>
<?xml-stylesheet type="text/xsl" ... ?>
<personen>
  <person>
    <name>
      <vorname>Alan</vorname>
      <nachname>Turing</nachname>
    </name>
    <geburtstag>23. Juni 1912</geburtstag>
    <beruf>Mathematiker</beruf>
    <beruf>Informatiker</beruf>
  </person>

  <person>
    <name>
      <vorname>Judea</vorname>
      <nachname>Pearl</nachname>
    </name>
    <geburtstag>unknown</geburtstag>
    <beruf>Informatiker</beruf>
  </person>
</personen>
```

Die XSL-Familie

XSLT-Prozessor: Verarbeitungsstrategie (Fortsetzung)

Stylesheet zur Elementselektion mit expliziter Verarbeitungssteuerung:

```
<?xml version="1.0"?>
<xsl:stylesheet version="1.0" xmlns:xsl="http://www.w3.org/...">

  <xsl:template match="person">
    <xsl:apply-templates select="name"/>
  </xsl:template>

  <xsl:template match="name">
    <xsl:value-of select="nachname"/>
    <xsl:text>, </xsl:text>
    <xsl:value-of select="vorname"/>
  </xsl:template>

</xsl:stylesheet>
```

```
<?xml version="1.0" ?>
<?xml-stylesheet type="text/xsl" ... ?>
<personen>
  <person>
    <name>
      <vorname>Alan</vorname>
      <nachname>Turing</nachname>
    </name>
    <geburtstag>23. Juni 1912</geburtstag>
    <beruf>Mathematiker</beruf>
    <beruf>Informatiker</beruf>
  </person>

  <person>
    <name>
      <vorname>Judea</vorname>
      <nachname>Pearl</nachname>
    </name>
    <geburtstag>unknown</geburtstag>
    <beruf>Informatiker</beruf>
  </person>
</personen>
```

Angewandt auf das Beispieldokument:

```
Turing, Alan
Pearl, Judea
```

Vergleiche die Elementselektion mittels leerer Template-Regeln.

Bemerkungen:

- Das `<xsl:apply-templates>`-Element startet für die mit dem `select`-Attribut spezifizierte Knotenmenge erneut einen Pre-Order-Durchlauf zur Anwendung der Template-Regeln des Stylesheets.
- Der Wert des `select`-Attributes im `<xsl:apply-templates>`-Element ist ein Lokalisierungspfad in eingeschränkter XPath-Syntax. Weil sich so beliebige Knoten im Dokument spezifizieren lassen, ermöglicht das `<xsl:apply-templates>`-Element die mehrmalige Verarbeitung von Knoten, also auch die Erzeugung von Endlosschleifen.
- Falls keine andere Achse angegeben ist, setzt der Lokalisierungspfad des `<xsl:apply-templates>`-Elements den Pfad des matchenden Knoten fort. Das heißt, die Ausdrücke `select=“./`*Elementname*” und `select=“`*Elementname*” spezifizieren dieselbe Knotenmenge.
- Enthält das `<xsl:apply-templates>`-Element kein `select`-Attribut, so gelten per Default die Kindknoten (`child::`-Achse) des matchenden Knoten als spezifiziert.

Die XSL-Familie

XSLT-Prozessor: Beispiele zur Verarbeitungsstrategie

Stylesheet mit zweifacher Verarbeitung der `<name>`-Kindelemente:

```
<?xml version="1.0"?>
<xsl:stylesheet version="1.0" xmlns:xsl="http://www.w3.org/...">

  <xsl:template match="person">
    <xsl:apply-templates select="name"/>
    <xsl:apply-templates select="name"/>
  </xsl:template>

  <xsl:template match="name">
    <xsl:value-of select="nachname"/>
    <xsl:text>, </xsl:text>
    <xsl:value-of select="vorname"/>
  </xsl:template>

</xsl:stylesheet>
```

Angewandt auf das Beispieldokument:

```
<?xml version="1.0" ?>
<?xml-stylesheet type="text/xsl" ... ?>
<personen>
  <person>
    <name>
      <vorname>Alan</vorname>
      <nachname>Turing</nachname>
    </name>
    <geburtstag>23. Juni 1912</geburtstag>
    <beruf>Mathematiker</beruf>
    <beruf>Informatiker</beruf>
  </person>

  <person>
    <name>
      <vorname>Judea</vorname>
      <nachname>Pearl</nachname>
    </name>
    <geburtstag>unknown</geburtstag>
    <beruf>Informatiker</beruf>
  </person>
</personen>
```

Die XSL-Familie

XSLT-Prozessor: Beispiele zur Verarbeitungsstrategie

Stylesheet mit zweifacher Verarbeitung der `<name>`-Kindelemente:

```
<?xml version="1.0"?>
<xsl:stylesheet version="1.0" xmlns:xsl="http://www.w3.org/...">

  <xsl:template match="person">
    <xsl:apply-templates select="name"/>
    <xsl:apply-templates select="name"/>
  </xsl:template>

  <xsl:template match="name">
    <xsl:value-of select="nachname"/>
    <xsl:text>, </xsl:text>
    <xsl:value-of select="vorname"/>
  </xsl:template>

</xsl:stylesheet>
```

```
<?xml version="1.0" ?>
<?xml-stylesheet type="text/xsl" ... ?>
<personen>
  <person>
    <name>
      <vorname>Alan</vorname>
      <nachname>Turing</nachname>
    </name>
    <geburtstag>23. Juni 1912</geburtstag>
    <beruf>Mathematiker</beruf>
    <beruf>Informatiker</beruf>
  </person>

  <person>
    <name>
      <vorname>Judea</vorname>
      <nachname>Pearl</nachname>
    </name>
    <geburtstag>unknown</geburtstag>
    <beruf>Informatiker</beruf>
  </person>
</personen>
```

Angewandt auf das Beispieldokument:

```
Turing, AlanTuring, Alan
Pearl, JudeaPearl, Judea
```

Die XSL-Familie

XSLT-Prozessor: Beispiele zur Verarbeitungsstrategie (Fortsetzung)

Stylesheet zur wiederholten Verarbeitung *aller* `<name>`-Elemente:

```
<?xml version="1.0"?>
<xsl:stylesheet version="1.0" xmlns:xsl="https://www.w3.org/...">

  <xsl:template match="person">
    <xsl:apply-templates select="//name"/>
  </xsl:template>

  <xsl:template match="name">
    <xsl:value-of select="nachname"/>
    <xsl:text>, </xsl:text>
    <xsl:value-of select="vorname"/>
  </xsl:template>

</xsl:stylesheet>
```

Angewandt auf das Beispieldokument:

```
<?xml version="1.0" ?>
<?xml-stylesheet type="text/xsl" ... ?>
<personen>
  <person>
    <name>
      <vorname>Alan</vorname>
      <nachname>Turing</nachname>
    </name>
    <geburtstag>23. Juni 1912</geburtstag>
    <beruf>Mathematiker</beruf>
    <beruf>Informatiker</beruf>
  </person>

  <person>
    <name>
      <vorname>Judea</vorname>
      <nachname>Pearl</nachname>
    </name>
    <geburtstag>unknown</geburtstag>
    <beruf>Informatiker</beruf>
  </person>
</personen>
```

Die XSL-Familie

XSLT-Prozessor: Beispiele zur Verarbeitungsstrategie (Fortsetzung)

Stylesheet zur wiederholten Verarbeitung *aller* `<name>`-Elemente:

```
<?xml version="1.0"?>
<xsl:stylesheet version="1.0" xmlns:xsl="https://www.w3.org/...">

  <xsl:template match="person">
    <xsl:apply-templates select="//name"/>
  </xsl:template>

  <xsl:template match="name">
    <xsl:value-of select="nachname"/>
    <xsl:text>, </xsl:text>
    <xsl:value-of select="vorname"/>
  </xsl:template>

</xsl:stylesheet>
```

```
<?xml version="1.0" ?>
<?xml-stylesheet type="text/xsl" ... ?>
<personen>
  <person>
    <name>
      <vorname>Alan</vorname>
      <nachname>Turing</nachname>
    </name>
    <geburtstag>23. Juni 1912</geburtstag>
    <beruf>Mathematiker</beruf>
    <beruf>Informatiker</beruf>
  </person>

  <person>
    <name>
      <vorname>Judea</vorname>
      <nachname>Pearl</nachname>
    </name>
    <geburtstag>unknown</geburtstag>
    <beruf>Informatiker</beruf>
  </person>
</personen>
```

Angewandt auf das Beispieldokument:

```
Turing, AlanPearl, Judea
Turing, AlanPearl, Judea
```

Die XSL-Familie

XSLT-Prozessor: Beispiele zur Verarbeitungsstrategie (Fortsetzung)

Stylesheet, dessen matchende Template-Regel die leere Knotenmenge liefert:

```
<?xml version="1.0"?>
<xsl:stylesheet version="1.0" xmlns:xsl="http://www.w3.org/...">

  <xsl:template match="person">
    <xsl:apply-templates select="nachname"/>
  </xsl:template>

  <xsl:template match="name">
    <xsl:value-of select="nachname"/>
    <xsl:text>, </xsl:text>
    <xsl:value-of select="vorname"/>
  </xsl:template>

</xsl:stylesheet>
```

```
<?xml version="1.0" ?>
<?xml-stylesheet type="text/xsl" ... ?>
<personen>
  <person>
    <name>
      <vorname>Alan</vorname>
      <nachname>Turing</nachname>
    </name>
    <geburtstag>23. Juni 1912</geburtstag>
    <beruf>Mathematiker</beruf>
    <beruf>Informatiker</beruf>
  </person>

  <person>
    <name>
      <vorname>Judea</vorname>
      <nachname>Pearl</nachname>
    </name>
    <geburtstag>unknown</geburtstag>
    <beruf>Informatiker</beruf>
  </person>
</personen>
```

Die XSL-Familie

XSLT-Prozessor: Beispiele zur Verarbeitungsstrategie (Fortsetzung)

Stylesheet, dessen Verarbeitung in eine Endlosschleife führt:

```
<?xml version="1.0"?>
<xsl:stylesheet version="1.0" xmlns:xsl="https://www.w3.org/...">

  <xsl:template match="person">
    <xsl:apply-templates select="/personen/person"/>
  </xsl:template>

  <xsl:template match="name">
    <xsl:value-of select="nachname"/>
    <xsl:text>, </xsl:text>
    <xsl:value-of select="vorname"/>
  </xsl:template>

</xsl:stylesheet>
```

Angewandt auf das Beispieldokument:

```
(Location of error unknown) XSLT Error
(java.lang.StackOverflowError): null
```

```
<?xml version="1.0" ?>
<?xml-stylesheet type="text/xsl" ... ?>
<personen>
  <person>
    <name>
      <vorname>Alan</vorname>
      <nachname>Turing</nachname>
    </name>
    <geburtstag>23. Juni 1912</geburtstag>
    <beruf>Mathematiker</beruf>
    <beruf>Informatiker</beruf>
  </person>

  <person>
    <name>
      <vorname>Judea</vorname>
      <nachname>Pearl</nachname>
    </name>
    <geburtstag>unknown</geburtstag>
    <beruf>Informatiker</beruf>
  </person>
</personen>
```

Die XSL-Familie

XSLT-Prozessor: Built-in-Templates [xpath notation]

1. Built-in-Template, das die rekursive Verarbeitung garantiert, falls kein matchendes Template im Stylesheet existiert:

```
<xsl:template match="*|/">
  <xsl:apply-templates/>
</xsl:template>
```

2. Built-in-Template zur Ausgabe von Text- und Attributknoten:

```
<xsl:template match="text()|@*">
  <xsl:value-of select="."/>
</xsl:template>
```

3. Built-in-Template, das die Kommentare matched und ignoriert:

```
<xsl:template match="processing-instruction()|comment()"/>
```

Vergleiche die Elementselektion mittels leerer Template-Regeln.

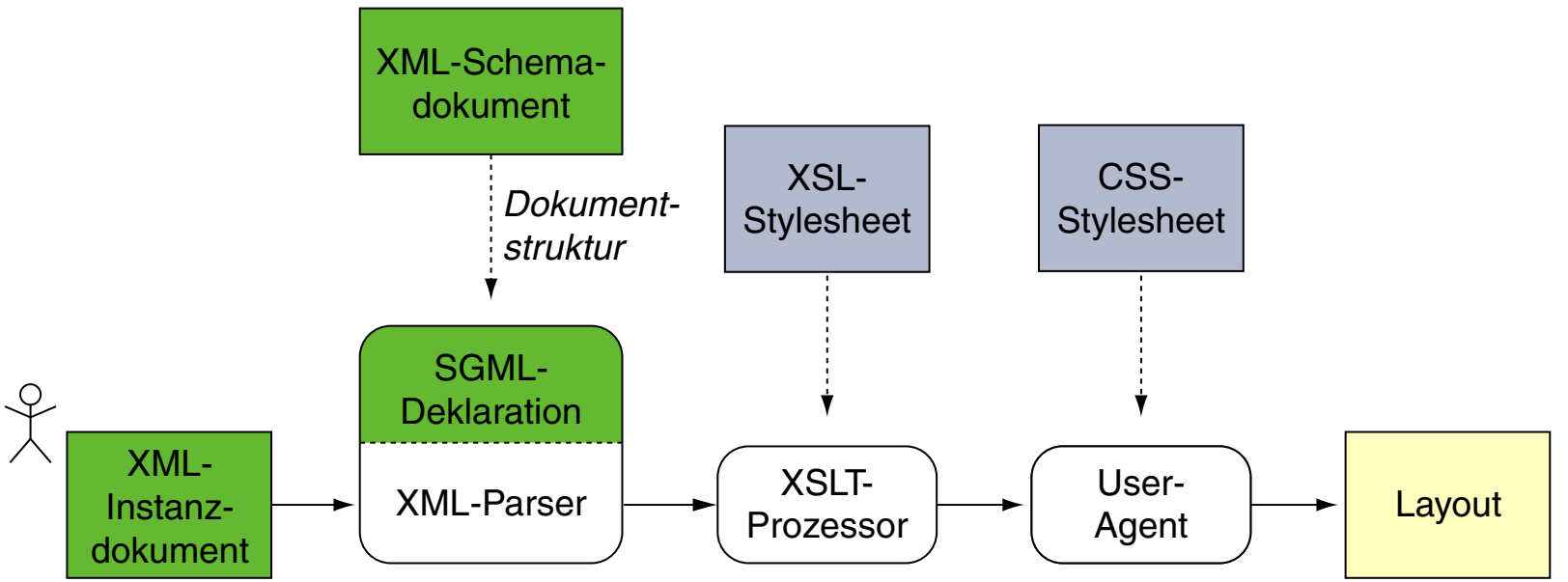
Die XSL-Familie

Weitere XSLT-Konzepte

- Template-Modi zur Charakterisierung von Verarbeitungsphasen
- benannte Templates zur Realisierung direkter Aufrufe
- Nummerierung und Sortierung von Ausgabeelementen
- bedingte Verarbeitung und Schleifen
- Import anderer Stylesheets

Die XSL-Familie

XML-Dokumentenverarbeitung: Erzeugung von HTML-Dokumenten



Vergleiche hierzu den Standardprozess der XSL Transformation.

Die XSL-Familie

Erzeugung von HTML-Dokumenten (Fortsetzung)

```
<?xml version="1.0" ?>
<?xml-stylesheet type="text/xsl" href="personen2html.xsl"?>
<personen>
  <person>
    <name>
      <vorname>Alan</vorname>
      <nachname>Turing</nachname>
    </name>
    <geburtstag>23. Juni 1912</geburtstag>
    <beruf>Mathematiker</beruf>
    <beruf>Informatiker</beruf>
  </person>

  <person>
    <name>
      <vorname>Judea</vorname>
      <nachname>Pearl</nachname>
    </name>
    <geburtstag>unknown</geburtstag>
    <beruf>Informatiker</beruf>
  </person>
</personen>
```

Die XSL-Familie

Erzeugung von HTML-Dokumenten (Fortsetzung) [Eingangsbeispiel]

```
<xsl:template match="personen">
  <html>
    <head>
      <title>
        <xsl:text>Personen</xsl:text>
      </title>
    </head>
    <body>
      <xsl:apply-templates/>
    </body>
  </html>
</xsl:template>
```

Die XSL-Familie

Erzeugung von HTML-Dokumenten (Fortsetzung) [Eingangsbeispiel]

```
<xsl:template match="personen">
  <html>
    <head>
      <title>
        <xsl:text>Personen</xsl:text>
      </title>
    </head>
    <body>
      <xsl:apply-templates/>
    </body>
  </html>
</xsl:template>
```

Die XSL-Familie

Erzeugung von HTML-Dokumenten (Fortsetzung) [Eingangsbeispiel]

```
<xsl:template match="personen">
  <html>
    <head>
      <title>
        <xsl:text>Personen</xsl:text>
      </title>
    </head>
    <body>
      <xsl:apply-templates/>
    </body>
  </html>
</xsl:template>
```

Die XSL-Familie

Erzeugung von HTML-Dokumenten (Fortsetzung) [Eingangsbeispiel]

```
<xsl:template match="personen">
  <html>
    <head>
      <title>
        <xsl:text>Personen</xsl:text>
      </title>
    </head>
    <body>
      <xsl:apply-templates/>
    </body>
  </html>
</xsl:template>

<xsl:template match="name">
  <p/>
  <div>
    <xsl:text>Name: </xsl:text>
    <span style="font-weight:bold">
      <xsl:value-of select="self::*"/>
    </span>
  </div>
</xsl:template>
...
```

Die XSL-Familie

Erzeugung von HTML-Dokumenten (Fortsetzung) [Eingangsbeispiel]

```
<xsl:template match="personen">
  <html>
    <head>
      <title>
        <xsl:text>Personen</xsl:text>
      </title>
    </head>
    <body>
      <xsl:apply-templates/>
    </body>
  </html>
</xsl:template>

<xsl:template match="name">
  <p/>
  <div>
    <xsl:text>Name: </xsl:text>
    <span style="font-weight:bold">
      <xsl:value-of select="self::*"/>
    </span>
  </div>
</xsl:template>
...
```

```
<?xml version="1.0" ?>
<?xml-stylesheet type="text/xsl" ... ?>
<personen>
  <person>
    <name>
      <vorname>Alan</vorname>
      <nachname>Turing</nachname>
    </name>
    <geburtstag>23. Juni 1912</geburtstag>
    <beruf>Mathematiker</beruf>
    <beruf>Informatiker</beruf>
  </person>

  <person>
    <name>
      <vorname>Judea</vorname>
      <nachname>Pearl</nachname>
    </name>
    <geburtstag>unknown</geburtstag>
    <beruf>Informatiker</beruf>
  </person>
</personen>
```

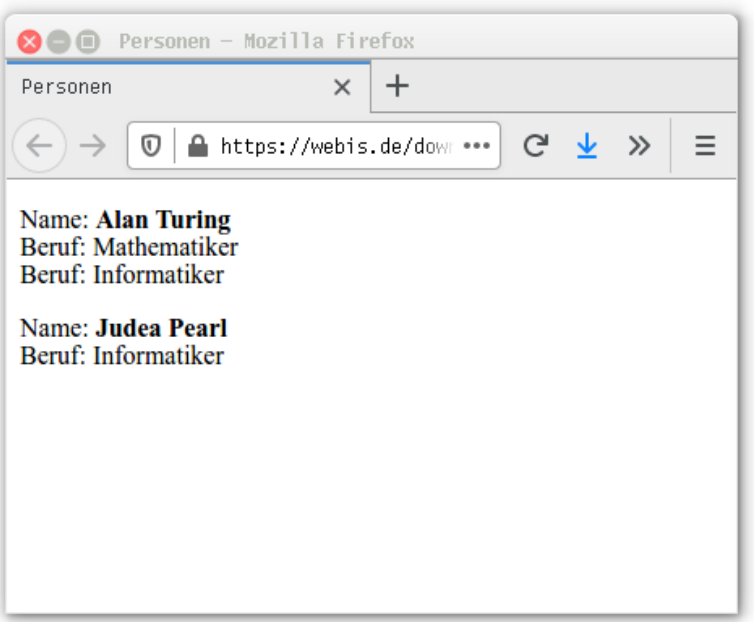
Die XSL-Familie

Erzeugung von HTML-Dokumenten (Fortsetzung) [Eingangsbeispiel]

```
<xsl:template match="personen">
  <html>
    <head>
      <title>
        <xsl:text>Personen</xsl:text>
      </title>
    </head>
    <body>
      <xsl:apply-templates/>
    </body>
  </html>
</xsl:template>

<xsl:template match="name">
  <p/>
  <div>
    <xsl:text>Name: </xsl:text>
    <span style="font-weight:bold">
      <xsl:value-of select="self::*"/>
    </span>
  </div>
</xsl:template>
...
```

[ohne / mit Stylesheet]

```
<?xml version="1.0" ?>
<?xml-stylesheet type="text/xsl" ... ?>
<personen>
  <person>
    <name>
      <vorname>Alan</vorname>
      <nachname>Turing</nachname>
    </name>
    <geburtstag>23. Juni 1912</geburtstag>
    <beruf>Mathematiker</beruf>
    <beruf>Informatiker</beruf>
  </person>

  <person>
    <name>
      <vorname>Judea</vorname>
```



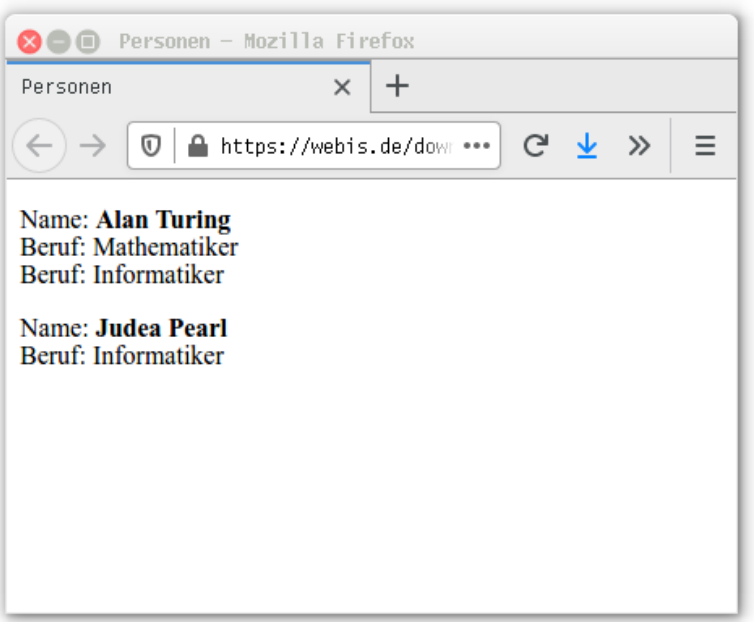
Die XSL-Familie

Erzeugung von HTML-Dokumenten (Fortsetzung) [Eingangsbeispiel]

```
<xsl:template match="personen">
  <html>
    <head>
      <title>
        <xsl:text>Personen</xsl:text>
      </title>
    </head>
    <body>
      <xsl:apply-templates/>
    </body>
  </html>
</xsl:template>

<xsl:template match="name">
  <p/>
  <div>
    <xsl:text>Name: </xsl:text>
    <span style="font-weight:bold">
      <xsl:value-of select="self::*"/>
    </span>
  </div>
</xsl:template>
...
```

[ohne / mit Stylesheet]

```
<?xml version="1.0" ?>
<?xml-stylesheet type="text/xsl" ... ?>
<personen>
  <person>
    <name>
      <vorname>Alan</vorname>
      <nachname>Turing</nachname>
    </name>
    <geburtstag>23. Juni 1912</geburtstag>
    <beruf>Mathematiker</beruf>
    <beruf>Informatiker</beruf>
  </person>

  <person>
    <name>
      <vorname>Judea</vorname>
```



Die XSL-Familie

Erzeugung von HTML-Dokumenten (Fortsetzung)

```
<xsl:template match="personen">
  <html>
    <head>
      <title>
        <xsl:text>Personen</xsl:text>
      </title>
      <link rel="stylesheet" type="text/css" href="personen.css"/>
    </head>
    <body>
      <xsl:apply-templates/>
    </body>
  </html>
</xsl:template>

<xsl:template match="name">
  <p/>
  <div>
    <xsl:text>Name: </xsl:text>
      <xsl:value-of select="self::*"/>
  </div>
</xsl:template>
...
```

Bemerkungen:

- Eine Anwendung nach diesem Schema sind die FAQs des W3C: Aus der XML-Source faq.xml gemäß der DTD faq.dtd wird mittels des Stylesheets faqxsl.xsl das HTML-Dokument faq.html erzeugt.

  Weil in faq.xml das Stylesheet faq.css verlinkt ist:

```
<?xml version="1.0"?>
<!DOCTYPE faq SYSTEM "faq.dtd">
<?xml-stylesheet href="faq.css" type="text/css"?>
<faq>
  <head>
    <title>Document Object Model FAQ</title>
    ...
```

  zeigt der Browser nicht den XML-Dokumentenbaum, sondern ein HTML-Dokument an.

Die XSL-Familie

Erzeugung von HTML-Dokumenten: Datenaufbereitung

CD-Datenbank als XML-Dokument [w3schools] :

```
<?xml version="1.0"?>
<?xml-stylesheet type="text/xsl" href="cdcatalog.xsl"?>
<catalog>
  <cd>
    <title>Empire Burlesque</title>
    <artist>Bob Dylan</artist>
    <company>Columbia</company>
    <price>10.90</price>
    <year>1985</year>
  </cd>
  ...
  <cd>
    <title>Unchain my heart</title>
    <artist>Joe Cocker</artist>
    <company>EMI</company>
    <price>8.20</price>
    <year>1987</year>
  </cd>
</catalog>
```

Die XSL-Familie

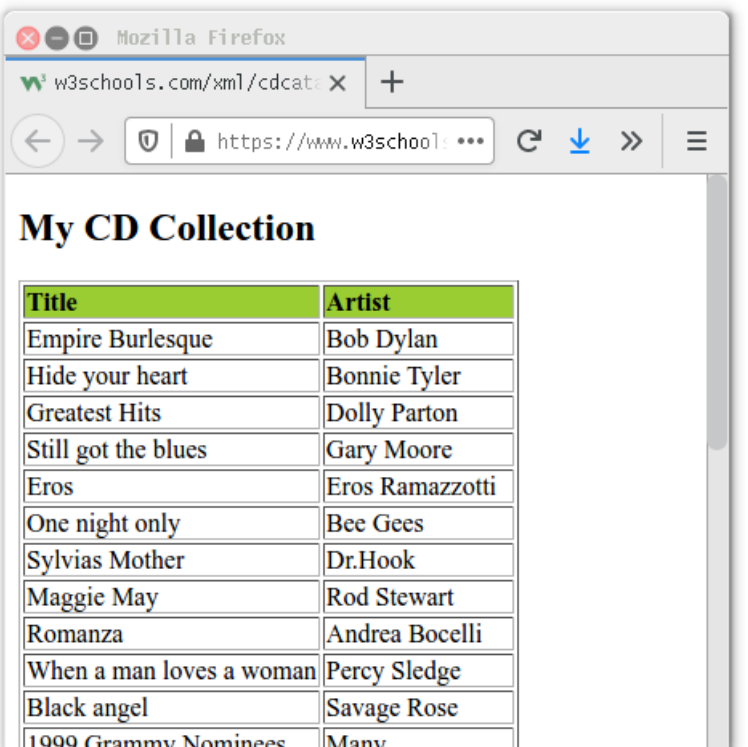
Erzeugung von HTML-Dokumenten: Datenaufbereitung (Fortsetzung)

```
<?xml version="1.0"?>
<xsl:stylesheet version="1.0" xmlns:xsl="http://www.w3.org/...">

<xsl:template match="/">
  <html>
  <body>
    <h2>My CD Collection</h2>
    <table border="1">
      <tr bgcolor="#9acd32">
        <th>Title</th>
        <th>Artist</th>
      </tr>
      <xsl:for-each select="catalog/cd">
      <tr>
        <td><xsl:value-of select="title"/></td>
        <td><xsl:value-of select="artist"/></td>
      </tr>
      </xsl:for-each>
    </table>
  </body>
  </html>
</xsl:template>
</xsl:stylesheet>
```

Die XSL-Familie

Erzeugung von HTML-Dokumenten: Datenaufbereitung (Fortsetzung)

```
<?xml version="1.0"?>
<xsl:stylesheet version="1.0" xmlns:xsl="http://www.w3.org/...">

<xsl:template match="/">
  <html>
  <body>
    <h2>My CD Collection</h2>
    <table border="1">
      <tr bgcolor="#9acd32">
        <th>Title</th>
        <th>Artist</th>
      </tr>
      <xsl:for-each select="catalog/cd">
      <tr>
        <td><xsl:value-of select="title"/>
        <td><xsl:value-of select="artist"/>
      </tr>
      </xsl:for-each>
    </table>
  </body>
  </html>
</xsl:template>
</xsl:stylesheet>
```



My CD Collection

Title	Artist
Empire Burlesque	Bob Dylan
Hide your heart	Bonnie Tyler
Greatest Hits	Dolly Parton
Still got the blues	Gary Moore
Eros	Eros Ramazzotti
One night only	Bee Gees
Sylvias Mother	Dr.Hook
Maggie May	Rod Stewart
Romanza	Andrea Bocelli
When a man loves a woman	Percy Sledge
Black angel	Savage Rose
1999 Grammy Nominees	Many

[w3schools xml, xsl, editor]

Die XSL-Familie

Erzeugung von HTML-Dokumenten: Datenaufbereitung (Fortsetzung)

Filtern mit XPath:

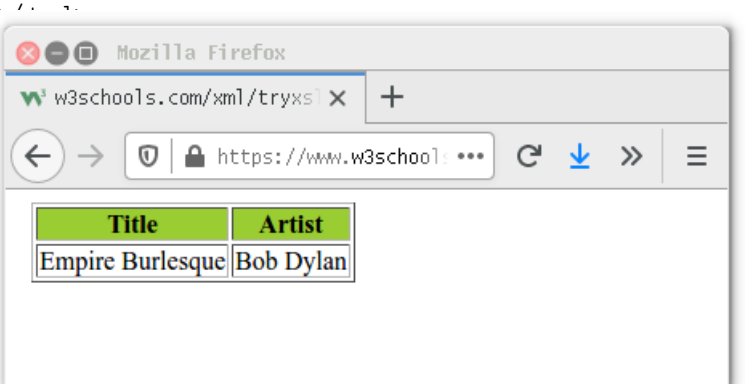
```
<xsl:template match="/">
  <html>
  <body>
    <h2>My CD Collection</h2>
    <table border="1">
      <tr bgcolor="#9acd32">
        <th>Title</th>
        <th>Artist</th>
      </tr>
      <xsl:for-each select="catalog/cd[artist='Bob Dylan']">
      <tr>
        <td><xsl:value-of select="title"/></td>
        <td><xsl:value-of select="artist"/></td>
      </tr>
      </xsl:for-each>
    </table>
  </body>
  </html>
</xsl:template>
```

Die XSL-Familie

Erzeugung von HTML-Dokumenten: Datenaufbereitung (Fortsetzung)

Filtern mit XPath:

```
<xsl:template match="/">
  <html>
  <body>
    <h2>My CD Collection</h2>
    <table border="1">
      <tr bgcolor="#9acd32">
        <th>Title</th>
        <th>Artist</th>
      </tr>
      <xsl:for-each select="catalog/cd[artist='Bob Dylan']">
      <tr>
        <td><xsl:value-of select="title"/></td>
        <td><xsl:value-of select="artist"/></td>
      </tr>
      </xsl:for-each>
    </table>
  </body>
  </html>
</xsl:template>
```



[w3schools]

Die XSL-Familie

Erzeugung von HTML-Dokumenten: Datenaufbereitung (Fortsetzung)

Sortierte Ausgabe:

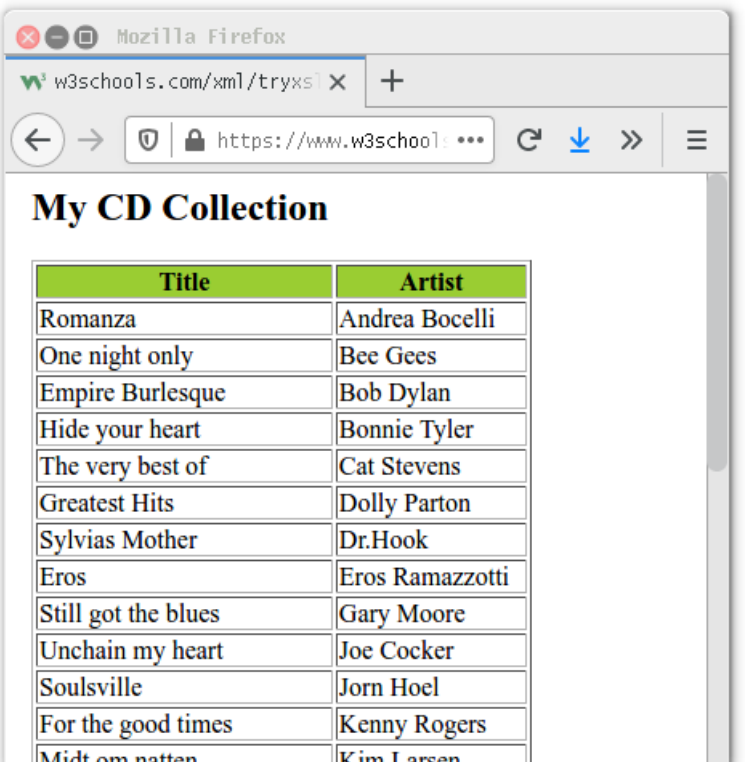
```
<xsl:template match="/">
  <html>
  <body>
    <h2>My CD Collection</h2>
    <table border="1">
      <tr bgcolor="#9acd32">
        <th>Title</th>
        <th>Artist</th>
      </tr>
      <xsl:for-each select="catalog/cd">
      <xsl:sort select="artist"/>
      <tr>
        <td><xsl:value-of select="title"/></td>
        <td><xsl:value-of select="artist"/></td>
      </tr>
      </xsl:for-each>
    </table>
  </body>
  </html>
</xsl:template>
```

Die XSL-Familie

Erzeugung von HTML-Dokumenten: Datenaufbereitung (Fortsetzung)

Sortierte Ausgabe:

```
<xsl:template match="/">
  <html>
  <body>
    <h2>My CD Collection</h2>
    <table border="1">
      <tr bgcolor="#9acd32">
        <th>Title</th>
        <th>Artist</th>
      </tr>
      <xsl:for-each select="catalog/cd">
      <xsl:sort select="artist"/>
      <tr>
        <td><xsl:value-of select="title"/>
        <td><xsl:value-of select="artist"/
      </tr>
      </xsl:for-each>
    </table>
  </body>
  </html>
</xsl:template>
```



My CD Collection

Title	Artist
Romanza	Andrea Bocelli
One night only	Bee Gees
Empire Burlesque	Bob Dylan
Hide your heart	Bonnie Tyler
The very best of	Cat Stevens
Greatest Hits	Dolly Parton
Sylvias Mother	Dr.Hook
Eros	Eros Ramazzotti
Still got the blues	Gary Moore
Unchain my heart	Joe Cocker
Soulsville	Jorn Hoel
For the good times	Kenny Rogers
Midt om natten	Kim Larsen

[w3schools]

Die XSL-Familie

Erzeugung von HTML-Dokumenten: Datenaufbereitung (Fortsetzung)

Bedingte Ausgabe:

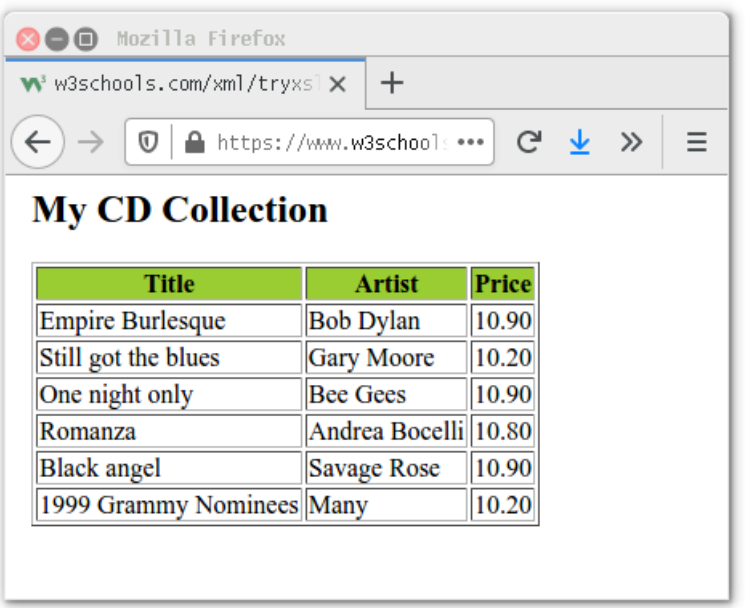
```
<xsl:template match="/">
  <html>
  <body>
    <h2>My CD Collection</h2>
    <table border="1">
      <tr bgcolor="#9acd32">
        <th>Title</th>
        <th>Artist</th>
      </tr>
      <xsl:for-each select="catalog/cd">
      <xsl:if test="price &gt; 10">
        <tr>
          <td><xsl:value-of select="title"/></td>
          <td><xsl:value-of select="artist"/></td>
        </tr>
      </xsl:if>
      </xsl:for-each>
    </table>
  </body>
  </html>
</xsl:template>
```

Die XSL-Familie

Erzeugung von HTML-Dokumenten: Datenaufbereitung (Fortsetzung)

Bedingte Ausgabe:

```
<xsl:template match="/">
  <html>
  <body>
    <h2>My CD Collection</h2>
    <table border="1">
      <tr bgcolor="#9acd32">
        <th>Title</th>
        <th>Artist</th>
      </tr>
      <xsl:for-each select="catalog/cd">
      <xsl:if test="price &gt; 10">
        <tr>
          <td><xsl:value-of select="title"/
          <td><xsl:value-of select="artist"
        </tr>
      </xsl:if>
      </xsl:for-each>
    </table>
  </body>
  </html>
</xsl:template>
```



[w3schools]

Die XSL-Familie

XML-Dokumentenverarbeitung: Elementinhalte anpassen [WT:III DOM-API]

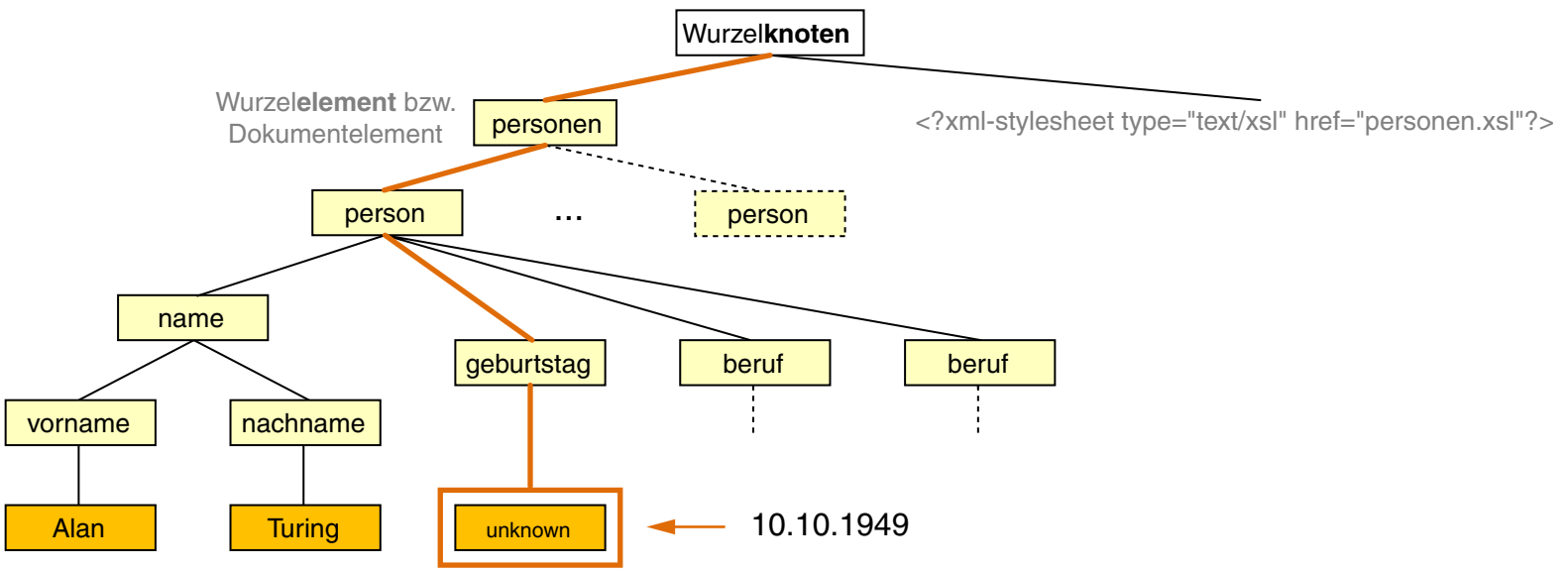
```
<?xml version="1.0" ?>
<?xml-stylesheet type="text/xsl" href="personen2html.xsl"?>
<personen>
  <person>
    <name>
      <vorname>Alan</vorname>
      <nachname>Turing</nachname>
    </name>
    <geburtstag>23. Juni 1912</geburtstag>
    <beruf>Mathematiker</beruf>
    <beruf>Informatiker</beruf>
  </person>

  <person>
    <name>
      <vorname>Judea</vorname>
      <nachname>Pearl</nachname>
    </name>
    <geburtstag>unknown</geburtstag>
    <beruf>Informatiker</beruf>
  </person>
</personen>
```

Die XSL-Familie

XML-Dokumentenverarbeitung: Elementinhalte anpassen (Fortsetzung)

Aufgabe:

1. Die Person „Judea Pearl“ finden.
2. Seinen Geburtstag auf einen bestimmten Wert setzen.



Die XSL-Familie

XML-Dokumentenverarbeitung: Elementinhalte anpassen (Fortsetzung)

Stylesheet:

```
<?xml version="1.0"?>
<xsl:stylesheet version="1.0" xmlns:xsl="http://www.w3.org/...">

  <xsl:template match="@*|node()"> [xpath notation]
    <xsl:copy><xsl:apply-templates select="@*|node()|"/></xsl:copy> [W3C]
  </xsl:template>




</xsl:stylesheet>
```

Die XSL-Familie

XML-Dokumentenverarbeitung: Elementinhalte anpassen (Fortsetzung)

Stylesheet:

```
<?xml version="1.0"?>
<xsl:stylesheet version="1.0" xmlns:xsl="http://www.w3.org/...">

  <xsl:template match="@*|node()"> [xpath notation]
    <xsl:copy><xsl:apply-templates select="@*|node()|"/></xsl:copy> [W3C]
  </xsl:template>

  <xsl:template match= "person[name/nachname='Pearl' and
       name/vorname='Judea']/geburtstag/text()">
    <xsl:text>10.10.1949</xsl:text>
  </xsl:template>

</xsl:stylesheet>
```

Die XSL-Familie

XML-Dokumentenverarbeitung: Elementinhalte anpassen (Fortsetzung)

Stylesheet:

```
<?xml version="1.0"?>
<xsl:stylesheet version="1.0" xmlns:xsl="http://www.w3.org/...">

  <xsl:template match="@*|node()"> [xpath notation]
    <xsl:copy><xsl:apply-templates select="@*|node()|"/></xsl:copy> [W3C]
  </xsl:template>

  <xsl:template match= "person[name/nachname='Pearl' and
       name/vorname='Judea']/geburtstag/text()">
    <xsl:text>10.10.1949</xsl:text>
  </xsl:template>

</xsl:stylesheet>
```

Angewandt auf das Beispieldokument:

```
...
  <name>
    <vorname>Judea</vorname>
    <nachname>Pearl</nachname>
  </name>
  <geburtstag>10.10.1949</geburtstag>
...
```

```
<?xml version="1.0" ?>
<?xml-stylesheet type="text/xsl" ... ?>
<personen>
  <person>
    <name>
      <vorname>Alan</vorname>
      <nachname>Turing</nachname>
    </name>
    <geburtstag>23. Juni 1912</geburtstag>
    <beruf>Mathematiker</beruf>
    <beruf>Informatiker</beruf>
  </person>

  <person>
    <name>
      <vorname>Judea</vorname>
      <nachname>Pearl</nachname>
    </name>
    <geburtstag>unknown</geburtstag>
    <beruf>Informatiker</beruf>
  </person>
</personen>
```

Die XSL-Familie

Quellen zum Nachlernen und Nachschlagen im Web: Referenz

- W3C. *XSL Transformations (XSLT) 2.0.*
  www.w3.org/TR/xslt20
- W3C *XML Path Language (XPath) 3.1.*
  www.w3.org/TR/xpath-31
- W3C *XML Query Language (XQuery) 3.1*
  www.w3.org/TR/xquery-31

Die XSL-Familie

Quellen zum Nachlernen und Nachschlagen im Web: Usage

- Apache. *Xalan Project.*
  xalan.apache.org
- MDN. *XSLTProcessor.*
  developer.mozilla.org/en-US/docs/Web/API/XSLTProcessor
- Saxonica.com. *XSLT and XQuery Processing.*
  www.saxonica.com
- W3 Schools. *XSLT.*
  www.w3schools.com/xml/xsl_intro.asp