Università
della
Svizzera
italiana

Facoltà
di scienze
informatiche

XML Query and Transformation XPath - XSLT

Prof. Cesare Pautasso

http://www.pautasso.info

cesare.pautasso@unisi.ch

XML Technology Landscape

- Data Representation
  - **XML Syntax**
  - XML Information Set (InfoSet)
  - **XML Namespaces**
  - **XML Schema, DTD**
  - XLink, XPointer
- Data Processing
  - XPath
  - XSLT – Extensible Stylesheet Transformation Language
  - XQuery
  - XUpdate
- Data Processing API
  - DOM
  - SAX
  - JAXP
- Communication Protocols
  - XML Forms
  - XML Web Services (SOAP, WSDL, UDDI)
  - XML Encryption
  - XML Digital Signature

Contents

- XPath
- XML and CSS
- XSLT – Extensible Stylesheet Language Transformation

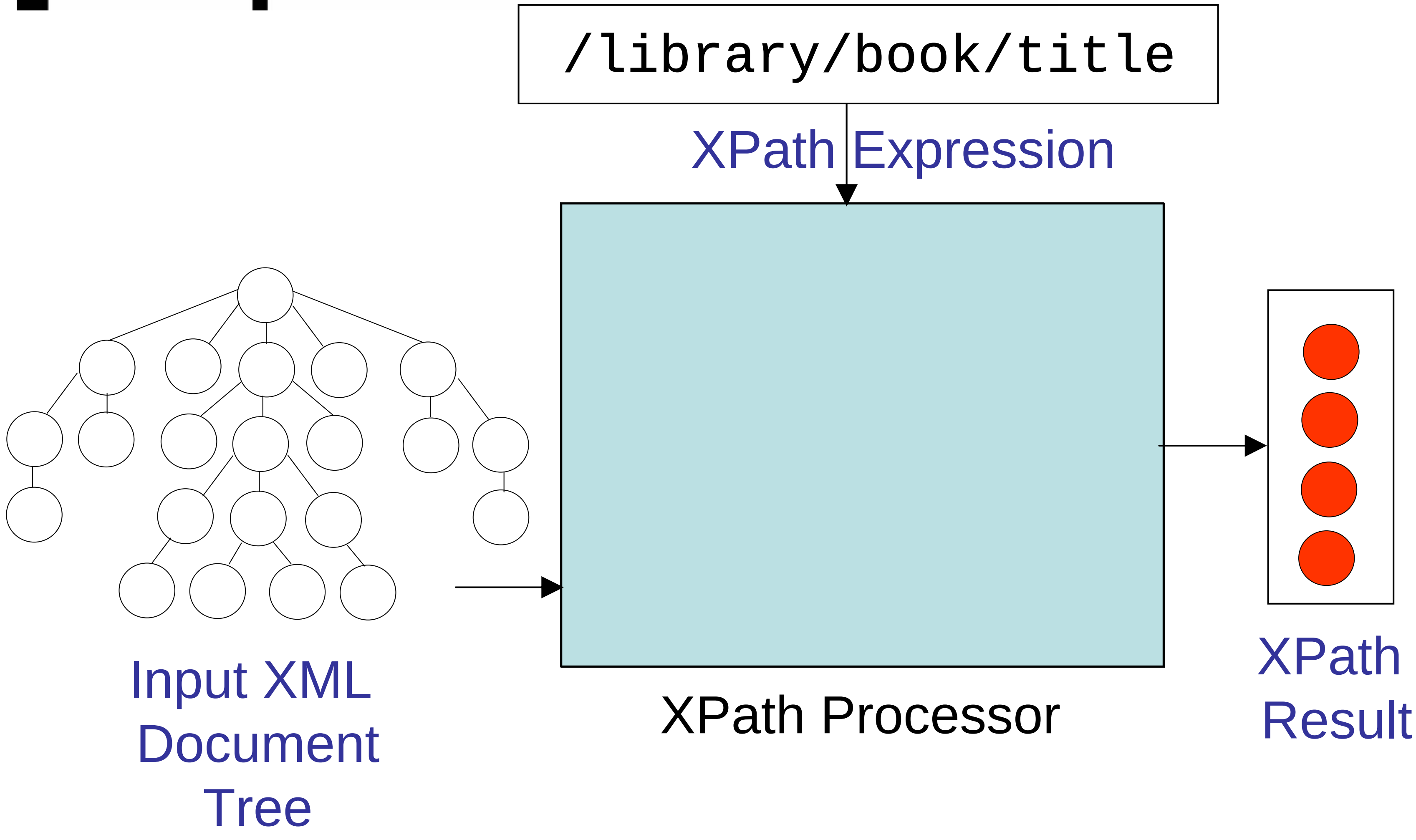
Università
della
Svizzera
italiana

Facoltà
di scienze
informatiche

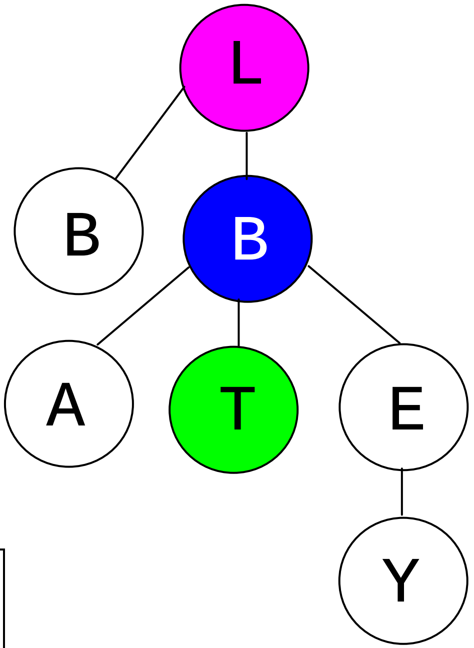
XPath

Università della Svizzera italiana | Facoltà di scienze informatiche

XPath Overview



XPath



- XPath specifies a navigation path along the XML tree in order to identify a subset of the nodes that are reachable using such path

```
/library/book/title
```

```
<library>
 <book isbn="0321269667"/>
 <book>
  <author>A</author>
  <title>XML for dummies</title>
  <edition>10th, <year>2008</year></edition>
 </book>
</library>
```

Location Paths

- A Location Path is a sequence of Location Steps (separated by `/`)
- Location Steps are used to test whether a node should be traversed

```
axis :: nodetest [ expression ]
```

Select the navigation direction:
- `child (default)`
- `attribute`
- `self`
- `parent`
- `descendant`
- `descendant-or-self`
- `ancestor`
- `following-sibling...`

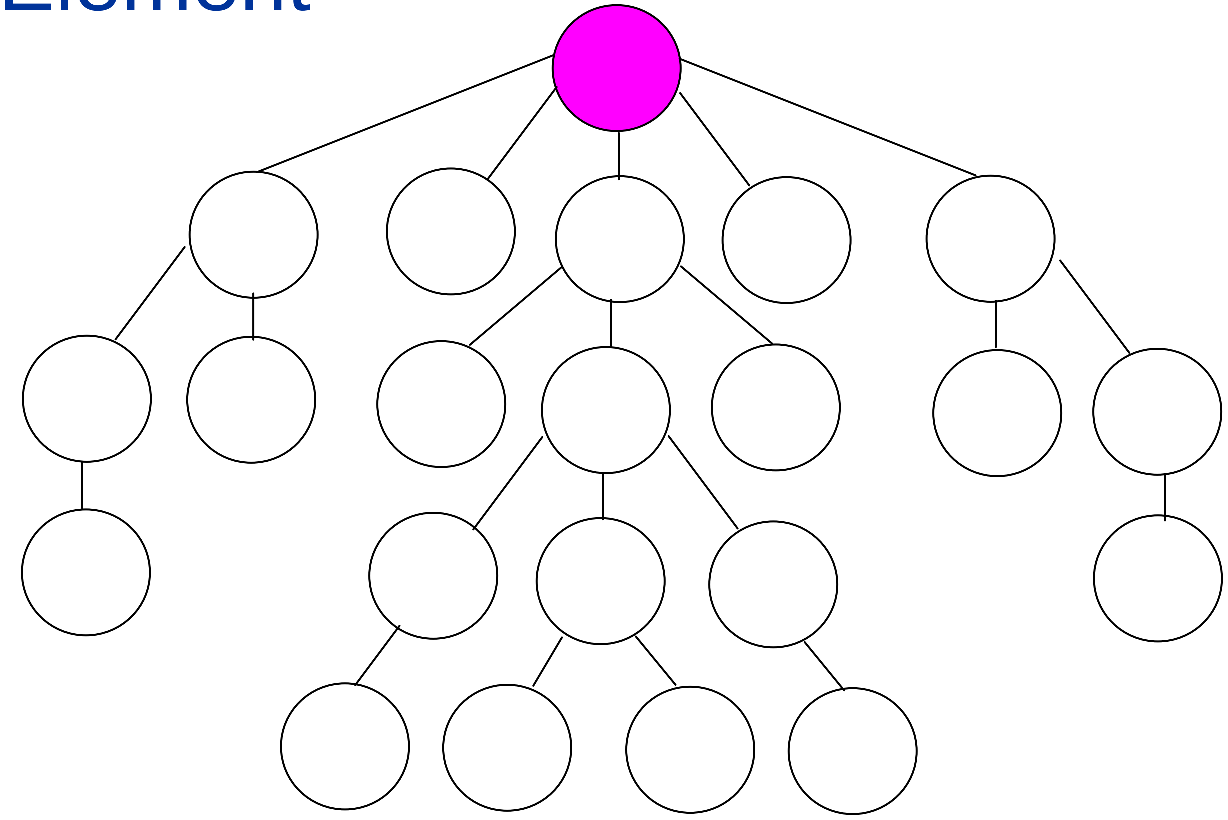
Check whether the node matches:
- name
- `*`
- `*:`localname
- prefix`:*`

Match nodes based on their kind:
- `text()`
- `comment()`
- `node()`

Università della Svizzera italiana | Facoltà di scienze informatiche

Working with Trees

- Root Element

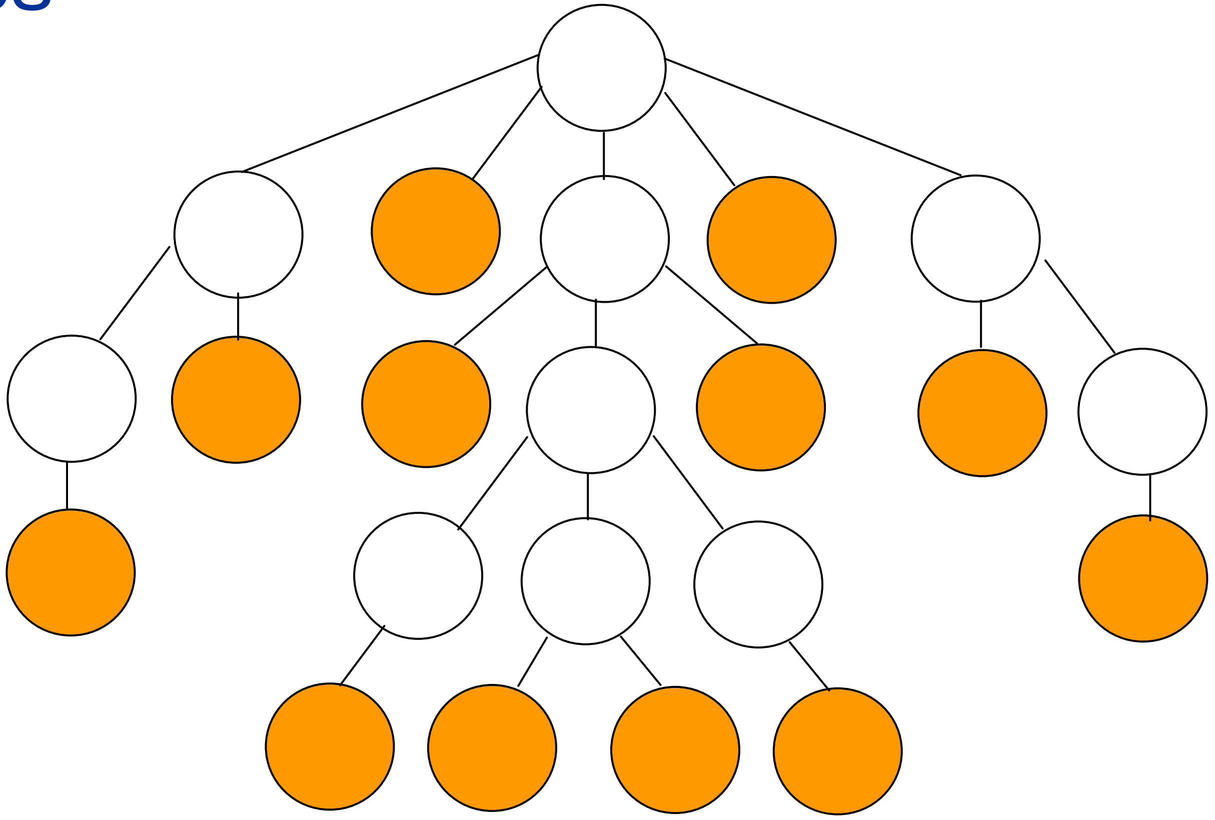


Università della Svizzera italiana

Facoltà di scienze informatiche

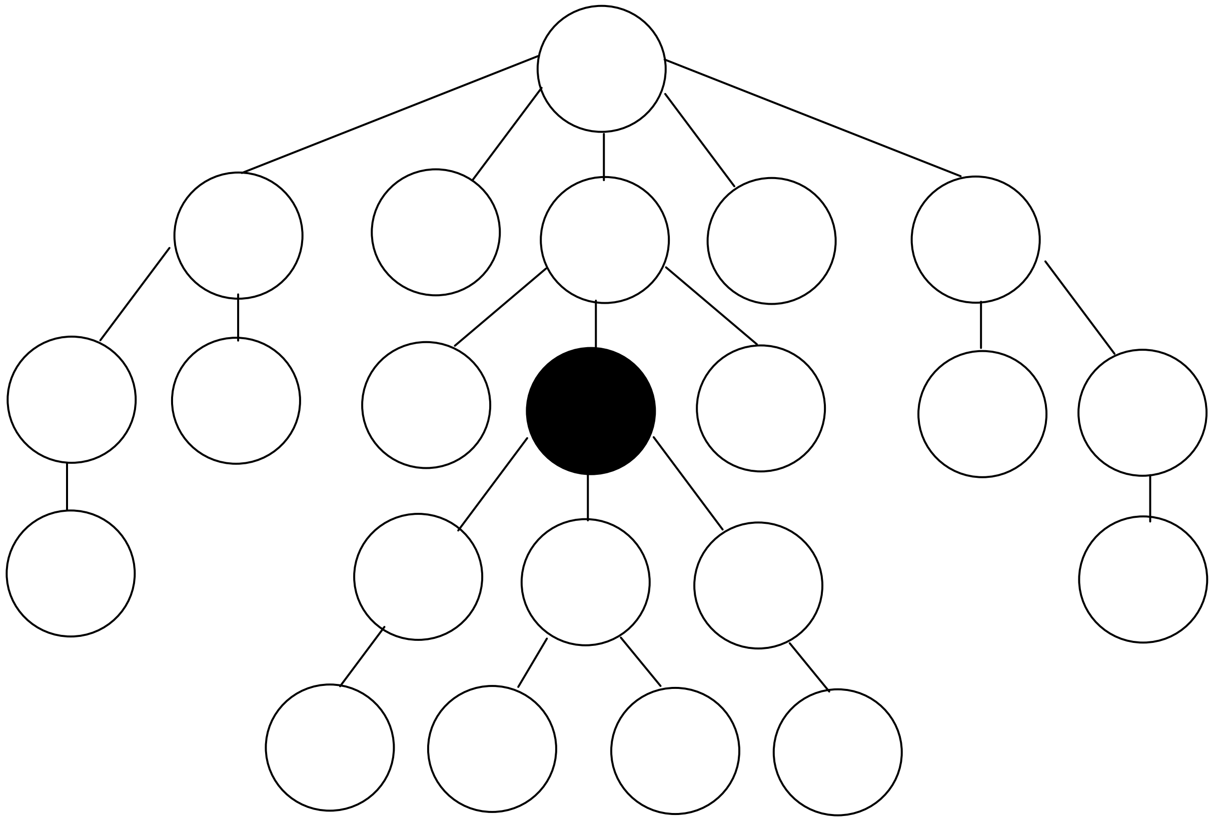
Working with Trees

- Leaves



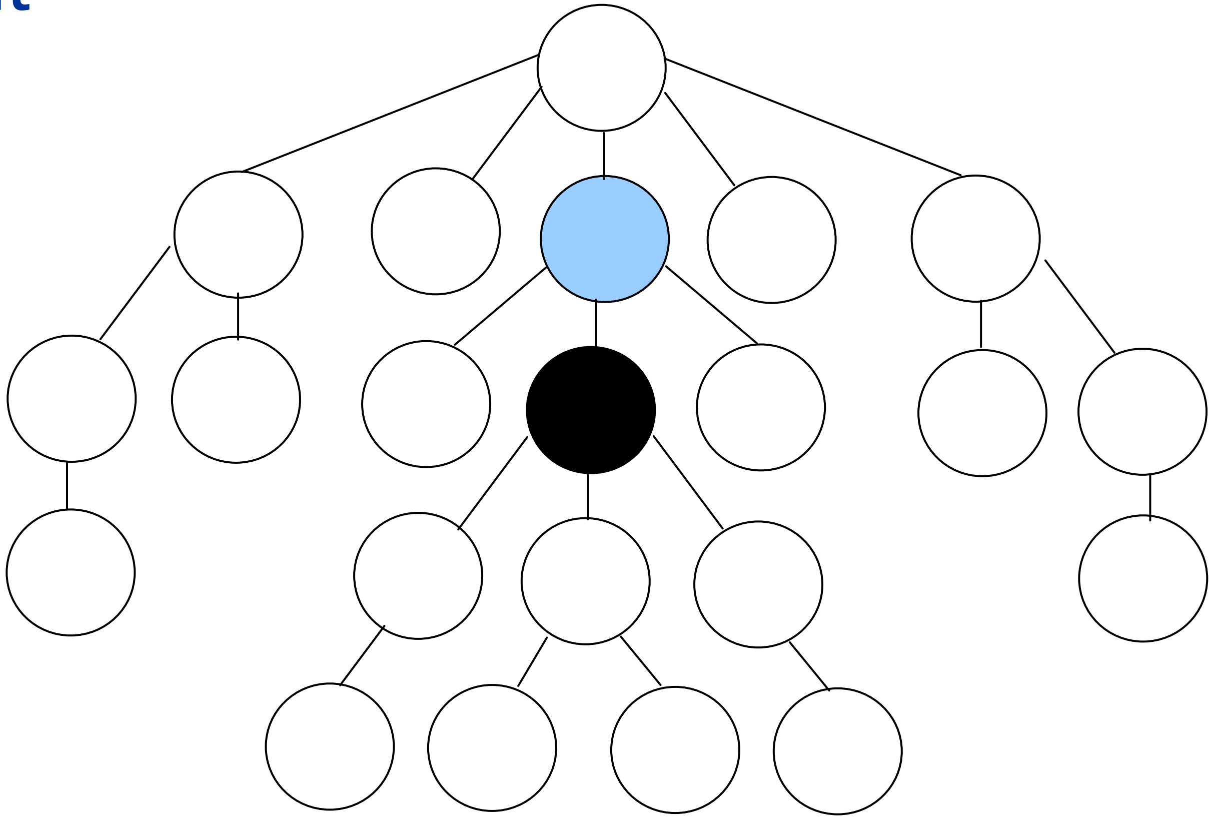
Working with Trees

- Self



Working with Trees

- Parent

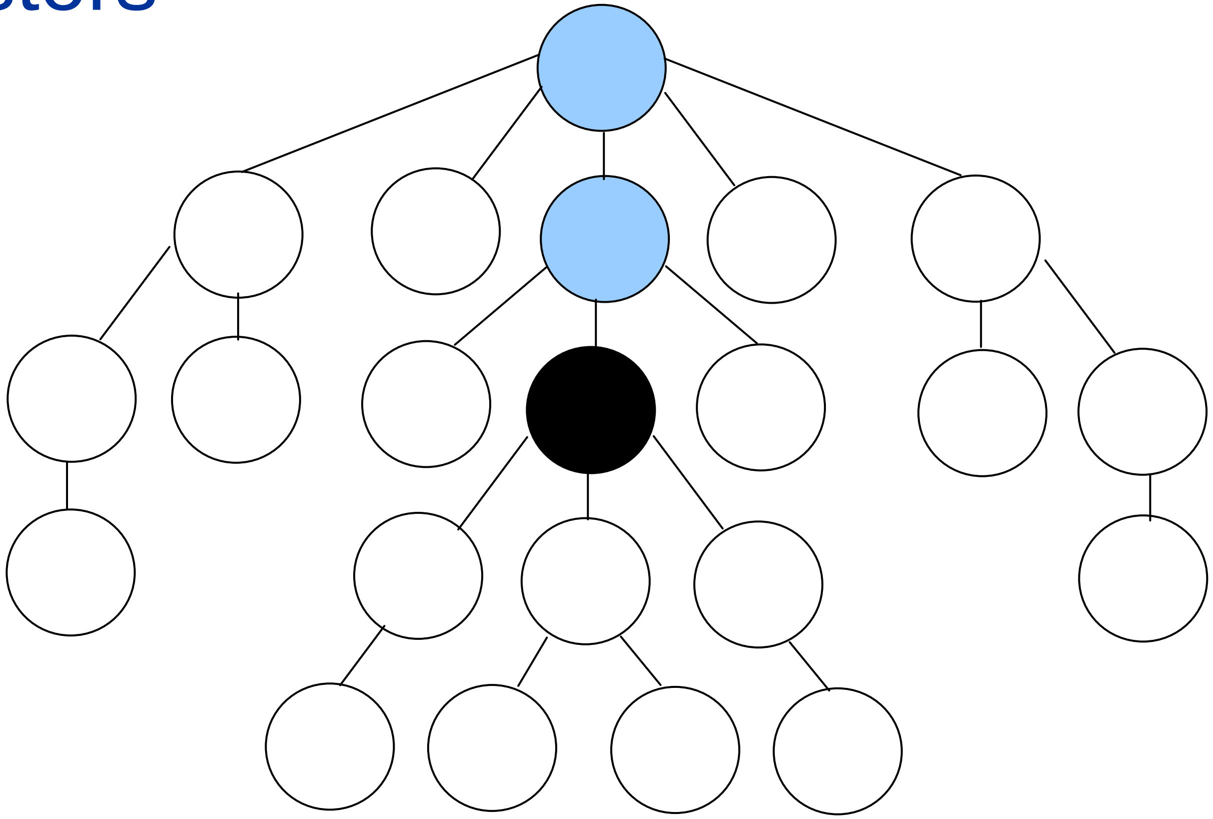


Università della Svizzera italiana

Facoltà di scienze informatiche

Working with Trees

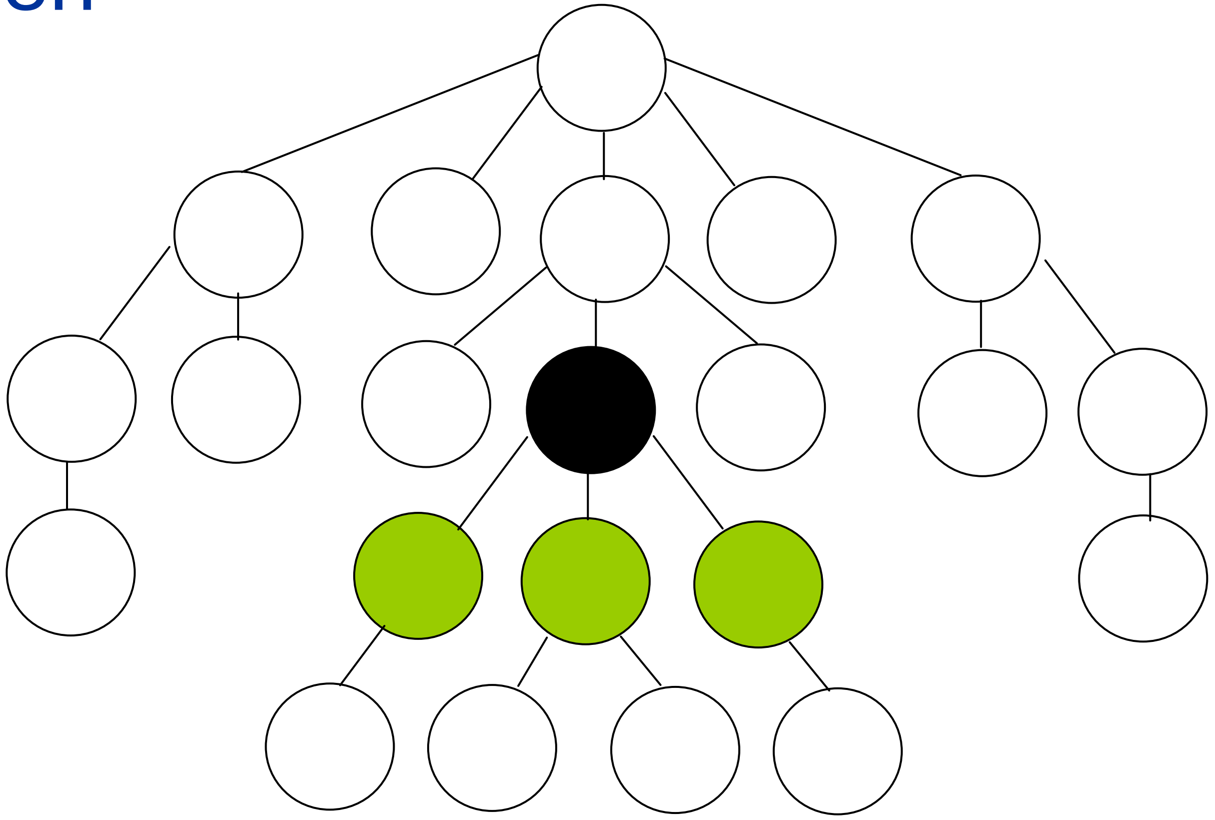
- Ancestors



Università della Svizzera italiana | Facoltà di scienze informatiche

Working with Trees

- Children

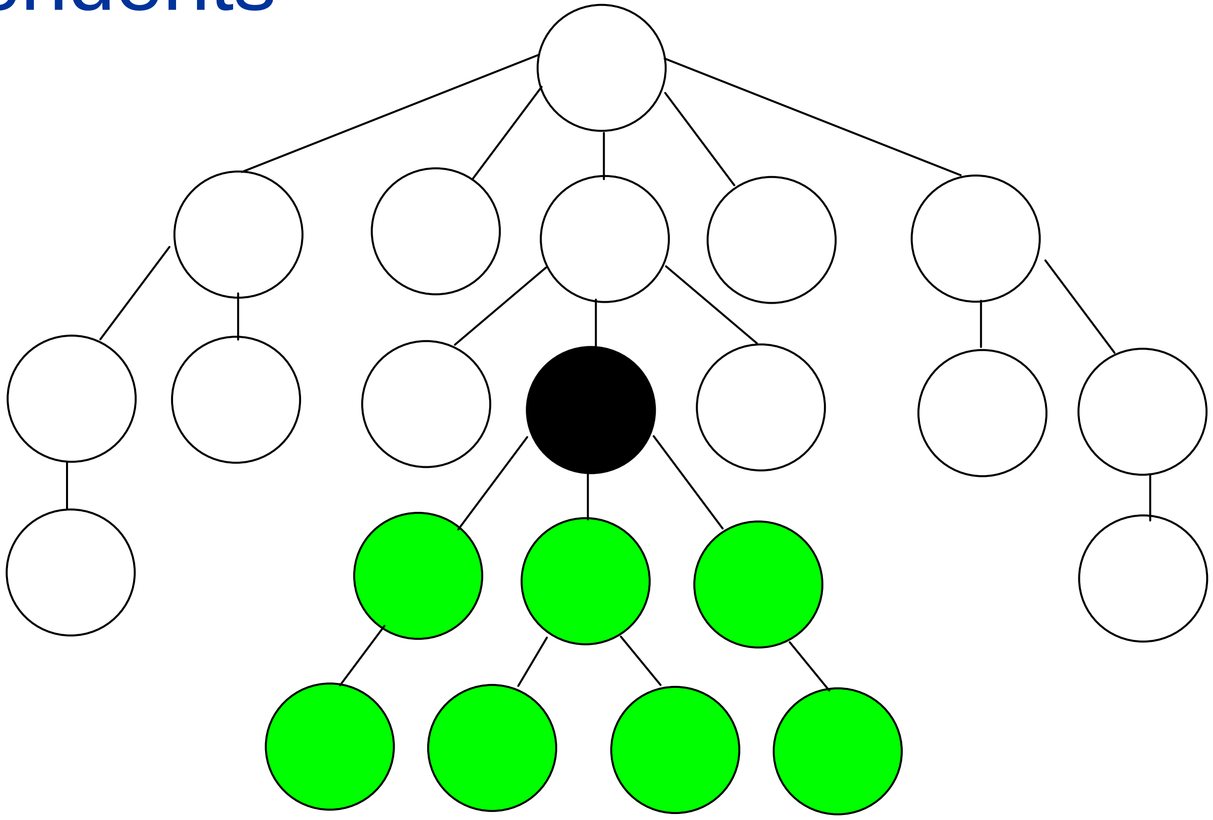


Università della Svizzera italiana

Facoltà di scienze informatiche

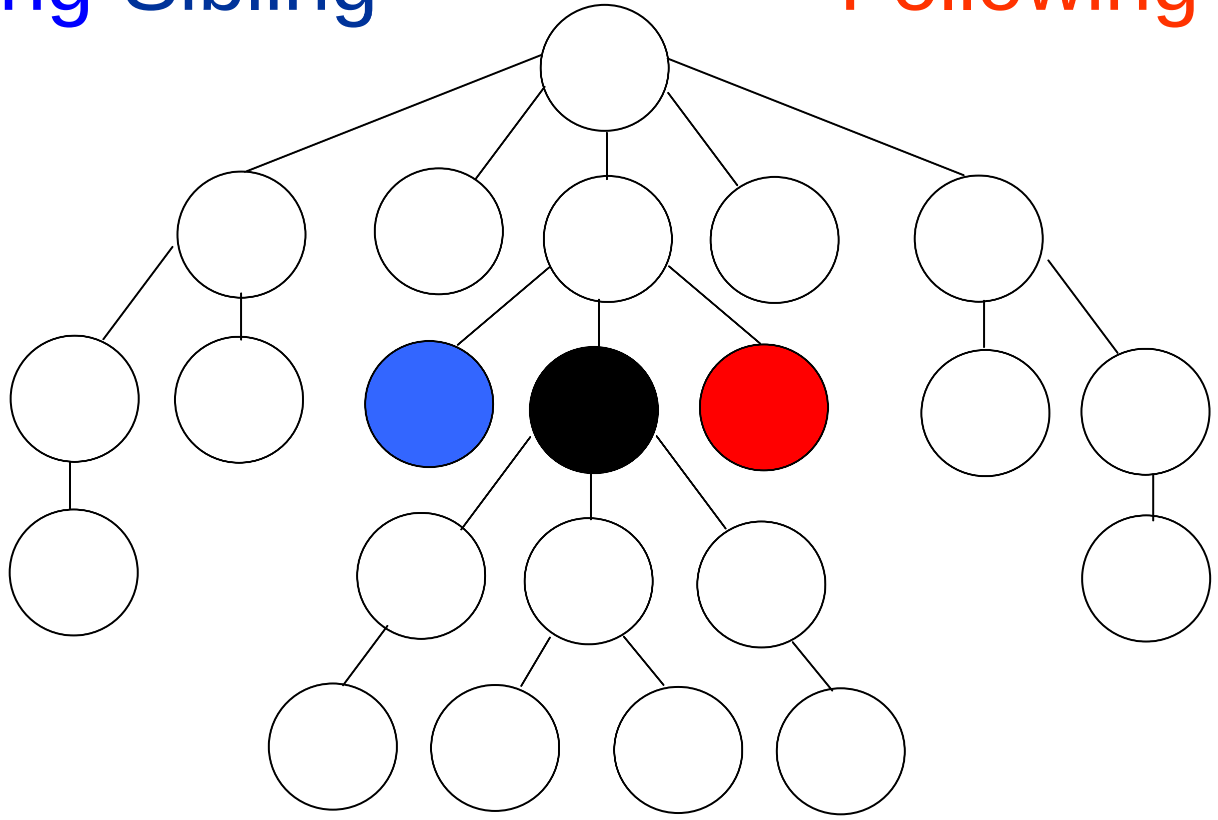
Working with Trees

- Descendents



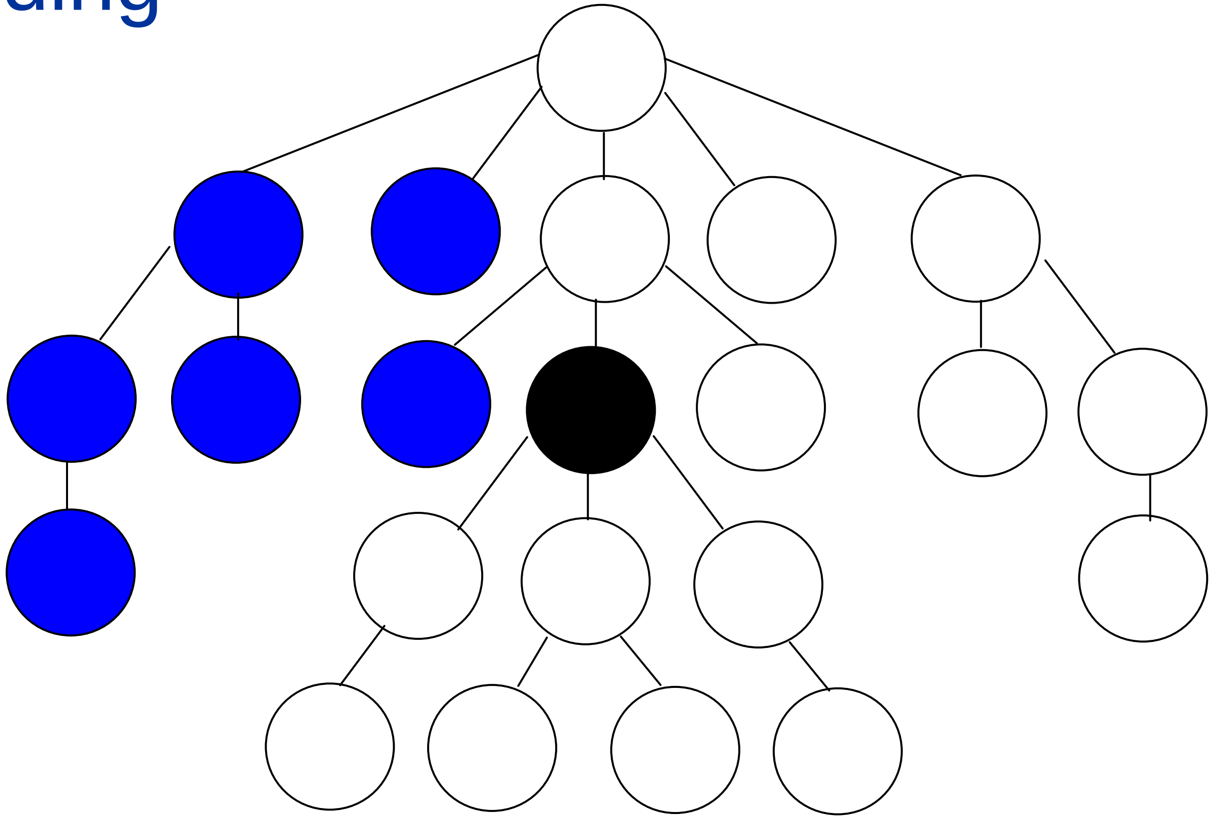
Working with Trees

Preceding-Sibling   Following-Sibling



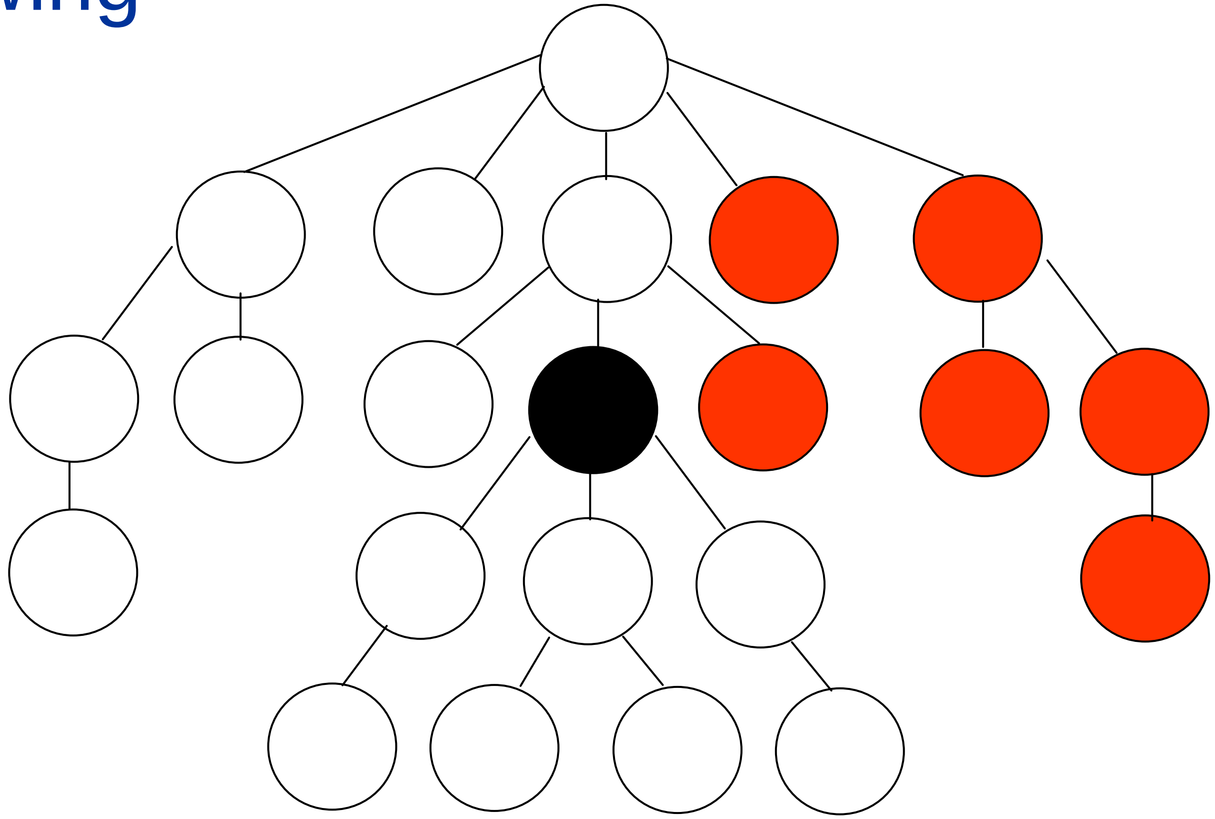
Working with Trees

- Preceding



Working with Trees

- Following



XPath by Example

- Select All Book Titles

```
/books/book/title/text()
//title/text()
```

- Return the ISBN of the books

```
/books/book/@isbn
```

- Return the year of the books that have a publisher

```
/books/book/publisher/../year
```

- Count how many book elements are in the document

```
count(//books)
```

Compact Notation

Default axis (Child)

```
/child::books/child::book
/books/book
```

Attribute axis replaced by @

```
//book/attribute::isbn
//book/@isbn
```

Navigating to the parent node shortened like in the file system

```
publisher/parent::node()/child::year
publisher/../year
```

Match any subtree

```
/descendant-or-self:node()/author
//author
```

XPath Predicate Examples

- Select the 2nd Book
```
/books/book[2]
```
- Return the author of the books that have an ISBN
```
/books/book[@isbn]/author
```
- Return the books that have a single author
```
/books/book[count(author)=1]
```
- Sum the prices (in EUR) of all books
```
sum(//price[@currency="EUR"])
```
- Select all books at even positions
```
//book[position() mod 2 = 0]
```

XPath 1.0 Functions

- Node Set Functions
  - `last`
  - `position`
  - `count`
  - `id`
  - `name`
- Math Functions
  - `sum`
  - `floor`
  - `ceiling`
  - `round`
- String Manipulation Functions
  - `concat`
  - `starts-with`
  - `contains`
  - `normalize-space`
  - `string-length`
  - `substring`
  - `translate`

XPath and JavaScript

```
function runxpath(xml, xpath) {

   //setup an empty document tree
   xmlDom=document.implementation.createDocument("","",null);
   xmlDom.onload = function() {

       //run the xpath query
       var nodes=xmlDom.evaluate(xpath, xmlDom, null,
                                 XPathResult.ANY_TYPE, null);
       //process the result nodes

   }
   //load and parse the XML document into the dom
   xmlDom.load(xml);

}
```

Università
della
Svizzera
italiana

Facoltà
di scienze
informatiche

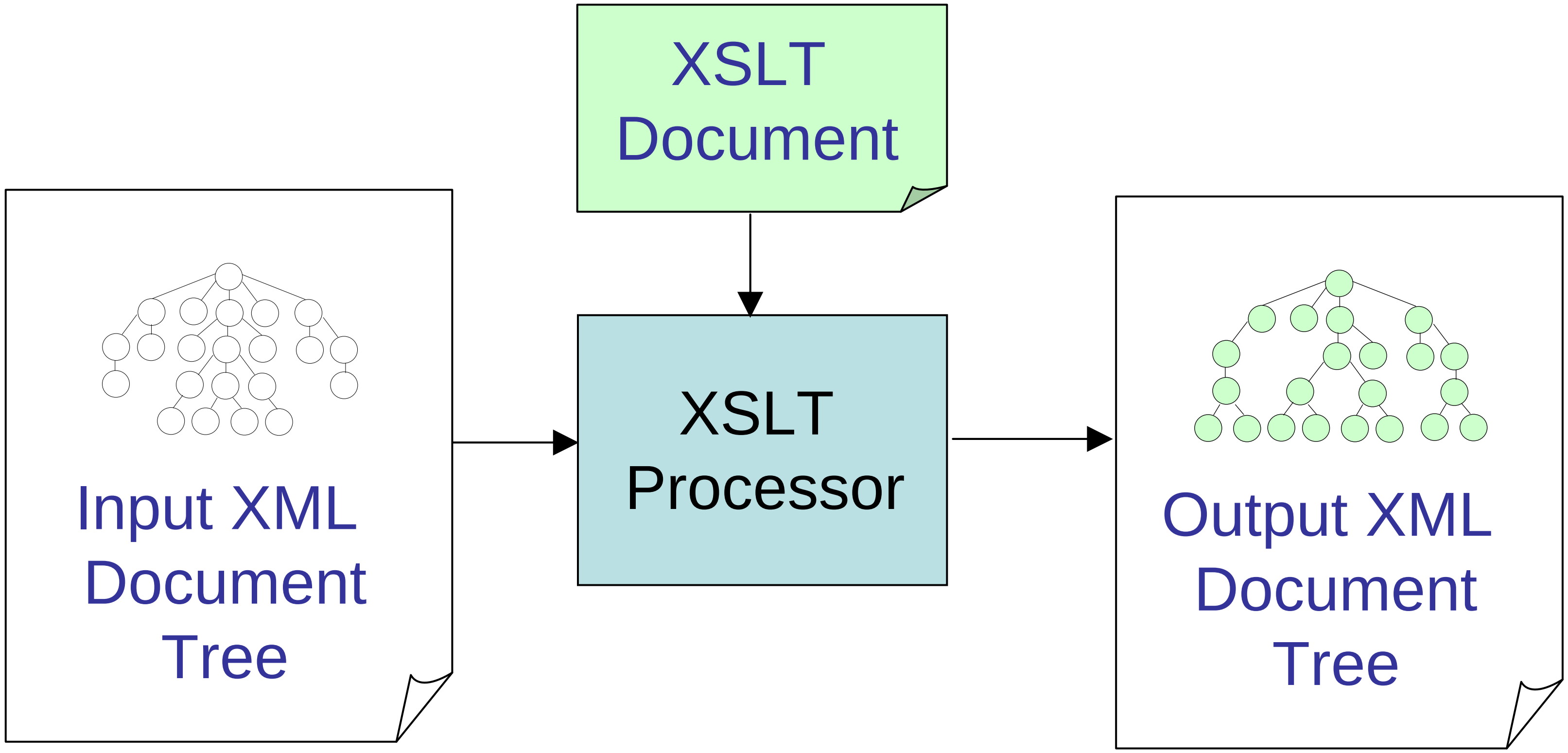
XSLT

eXtensible Stylesheet Language Transformation

Format XML with CSS

- Add this processing instruction to your XML document and open it in a browser:
  `<?xml-stylesheet type="text/css" href=“style.css"?>`
- The CSS style sheet can use the same formatting properties as for HTML pages.
- The only difference are the selectors, which should refer to the XML element tags as defined by the DTD/Schema of the XML document
- Limitations:
  - Only XML text elements are visualized (attributes remain hidden)
  - Cannot sort, manipulate and filter the information of the XML
  - Cannot introduce additional HTML page elements (images, tables, forms…)
- Solution: use XSLT instead.
  `<?xml-stylesheet type="text/xsl“ href="style.xsl"?>`

XSLT Overview



http://saxon.sourceforge.net/
http://eclipsexslt.sourceforge.net/

XSLT

- eXtensible Stylesheet Language Transformation (W3C, 1.0 1999 – 2.0 2007)
- XSLT transforms an XML document into another XML document (so the output can also be in XHTML)
- XSLT is a **declarative language** based on rules that match elements of the input XML document and transform them based on templates.
- XSLT uses a simplified version of XPath to navigate and access the input XML document nodes
- XSLT uses Namespaces to mix XSLT processing tags with the XML templates that specify the structure of the output document

Università della Svizzera italiana

Facoltà di scienze informatiche

XSLT Example

```
<?xml version="1.0" encoding="UTF-8"?>
<xsl:stylesheet version="1.0“
xmlns:xsl="http://www.w3.org/1999/XSL/Transform">

<xsl:template match="/books">
  <html><body>
    <h2>My Library</h2>
    <table border="1">
      <tr>
        <th>ISBN</th> <th>Author</th> <th>Title</th>
      </tr>
      <xsl:for-each select="book[@isbn]">
      <tr>
        <td class="isbn"><xsl:value-of select="@isbn" /></td>
        <td><xsl:value-of select="author" /></td>
        <td><xsl:value-of select="title" /></td>
      </tr>
      </xsl:for-each>
    </table>
  </body></html>
</xsl:template>

</xsl:stylesheet>
```

XPath Expressions

Template Rule

XSLT Template

```
<xsl:template match="Pattern">
Output
</xsl:template>
```

Example Patterns
`Name`
`Name/Name`
`Name[@a="…"]`
`/`
`*`
`Name|Name`

- The Template element defines the structure of the Output that is produced when the Pattern rule (XPath) matches the Input document
- If more than one pattern matches, the most specific is executed (like CSS selectivity)
- The template contains a set of "constructor elements" instructions and XML elements that will be literally copied in the output
- Templates can be associated with Modes (2.0)

Reading the input

```
<xsl:value-of
  select="Pattern"/>
{Pattern}
```

- The value-of instruction is replaced with the value of the XPath pattern evaluated in the context of the input elements matched by the template
- The {} notation is only used to generate attribute values

```
<xsl:for-each
  select="Pattern">
```

Iterate over all elements returned by the XPath pattern

```
<xsl:apply-templates
  select="Pattern">
<xsl:call-template
  name="…">
```

- Call another template rule (Helps to modularize the stylesheet)

Generating the output

- Apart from literal values, you can use explicit instructions to construct the output document:

```
<xsl:element name="…">
  <xsl:attribute name="…" select="Pattern"/>
  <xsl:text>…</xsl:text>
</xsl:element>
```

These are more verbose, but can be useful if you do not want to mix the XSLT instructions with the literal elements of the output.

```
<xsl:text>2+2 = </xsl:text>
<xsl:value-of select="2+2"/>
```

Explicit Constructors Example

```
<xsl:for-each select="book">
  <xsl:element name="tr">
    <xsl:element name="td">
    <xsl:attribute name="class" select="'isbn'">
      <xsl:value-of select="@isbn" />
    </xsl:element>
    <xsl:element name="td">
      <xsl:value-of select="author" />
    </xsl:element>
    <xsl:element name="td">
       <xsl:value-of select="title" />
    </xsl:element>
  </xsl:element>
</xsl:for-each>
```

Variables and Parameters

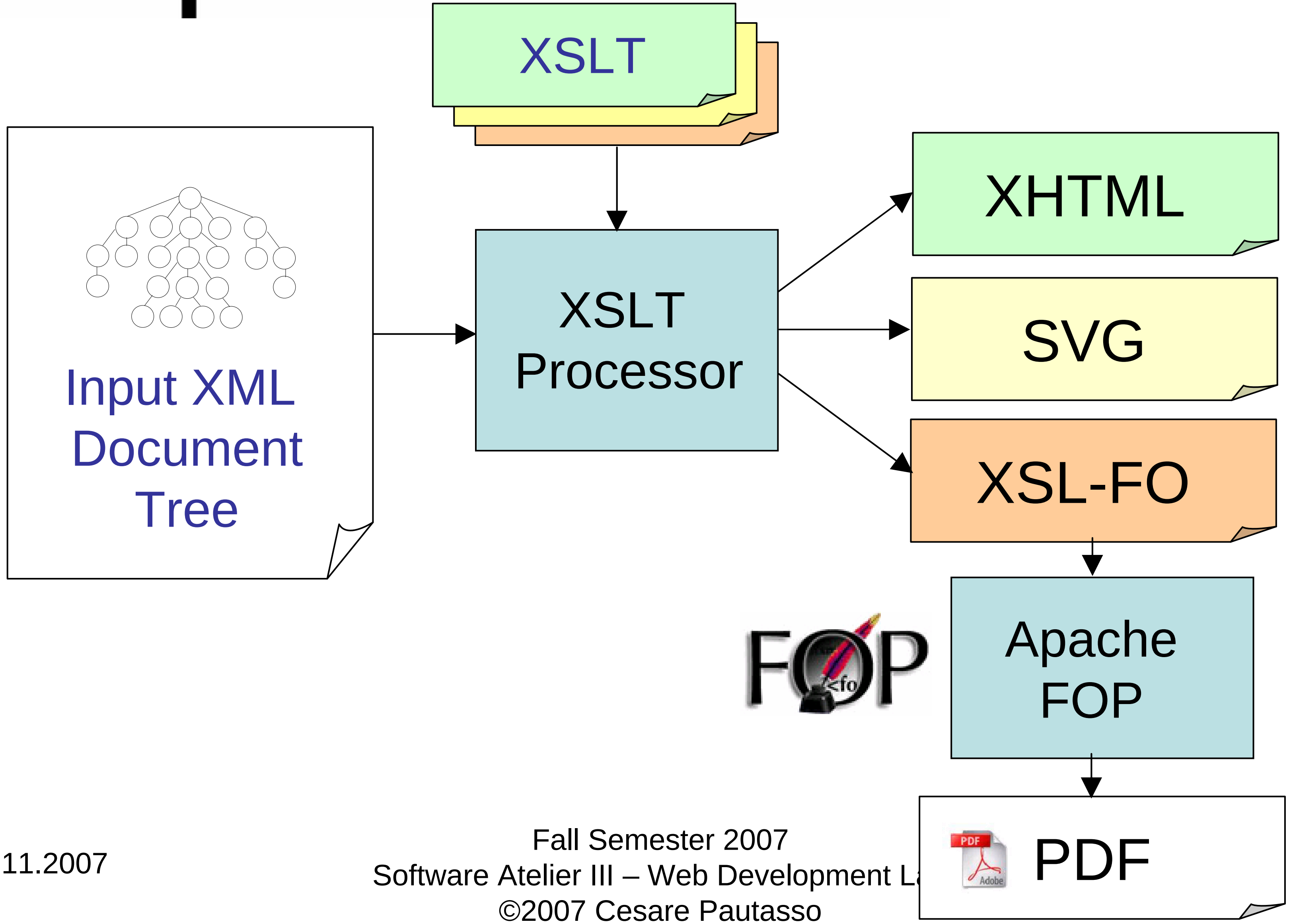
```
<xsl:template name="fib">
 <xsl:param name="n"/>
 <xsl:choose>
  <xsl:when test="$n le 1">
   <xsl:value-of select="1"/>
  </xsl:when>
  <xsl:otherwise>
   <xsl:variable name="f1">
    <xsl:call-template name="fib">
     <xsl:with-param name="n" select="$n - 1"/>
    </xsl:call-template>
   </xsl:variable>
   <xsl:variable name="f2">
    <xsl:call-template name="fib">
     <xsl:with-param name="n" select="$n - 2"/>
    </xsl:call-template>
   </xsl:variable>
   <xsl:value-of select="$f1 + $f2"/>
  </xsl:otherwise>
 </xsl:choose>
</xsl:template>
```

```
<xsl:template match="/">
 <xsl:call-template name="fib">
  <xsl:with-param name="n“ select="10"/>
 </xsl:call-template>
</xsl:template>
```

- Templates (and entire stylesheets) can be parametric.
- Two forms of variable declaration:

```
<xsl:variable name=“…”
  select=“Pattern”/>
<xsl:variable name=“…”>
Value
</xsl:variable>
```

XSLT Applications



Università della Svizzera italiana

Facoltà di scienze informatiche

References

- Anders Moller and Michael Schwartzbach, **An Introduction to XML and Web Technologies**, Addison-Wesley, 2006
- Elliotte Rusty Harold and W. Scott Means, **XML in a Nutshell**, O’Reilly, 3rd Ed. 2004
- Sal Mangano, **XSLT Cookbook**, O’Reilly, 2005