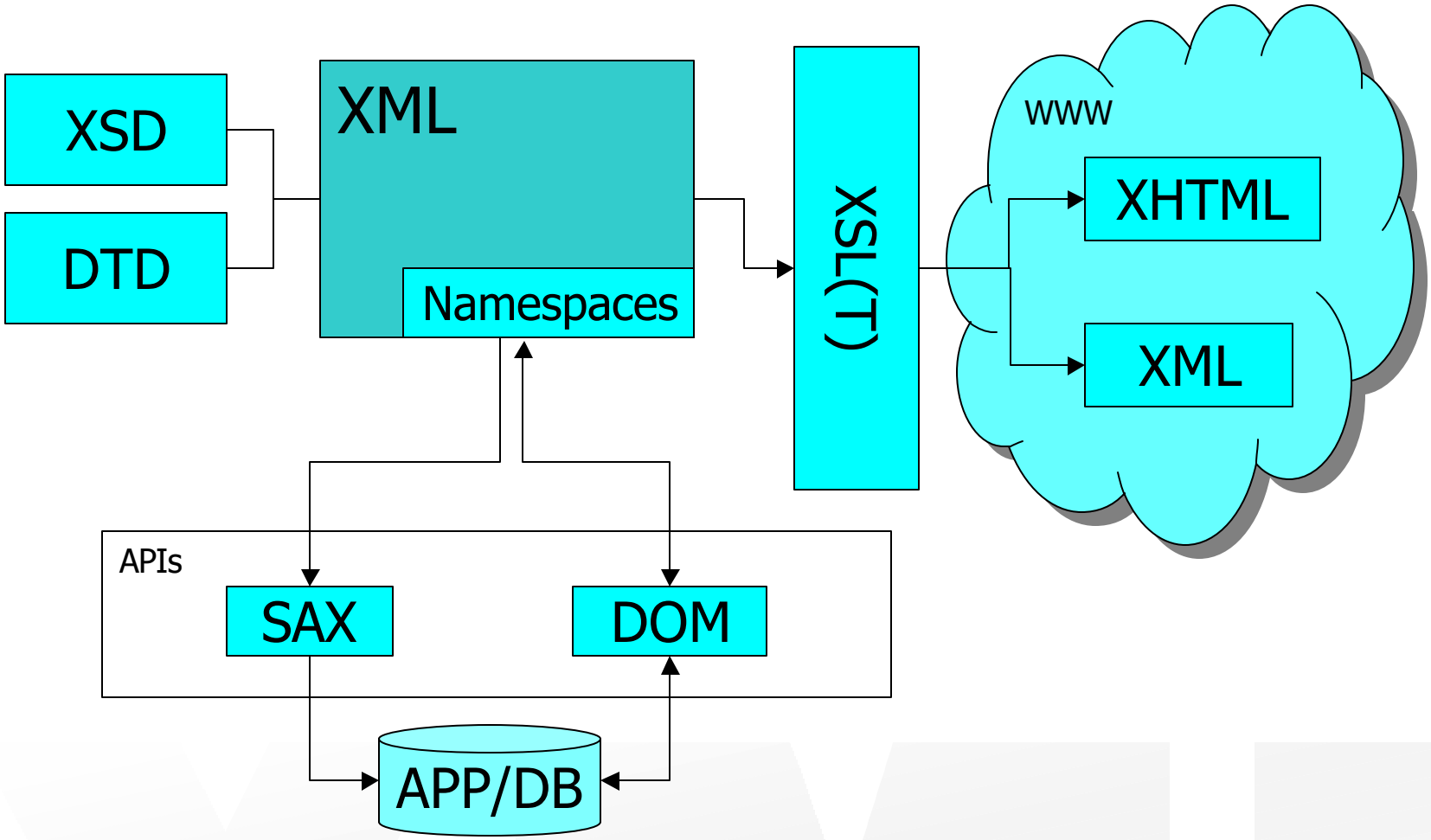
Session [3]

- **XML Rendering and Transformation with XSL**

- September 21, 2000
  Horst Rechner

Display



CSS - Example

- Links: http://www.w3.org/Style/CSS/

Address_Book_css.xml

```
<?xml version="1.0"?>
<?xml-stylesheet type="text/css"
href="Address_Book.css"?>
<Address_Book>
        <Address>
                <Name>Steve</Name>
                <Phone>123</Phone>
        </Address>

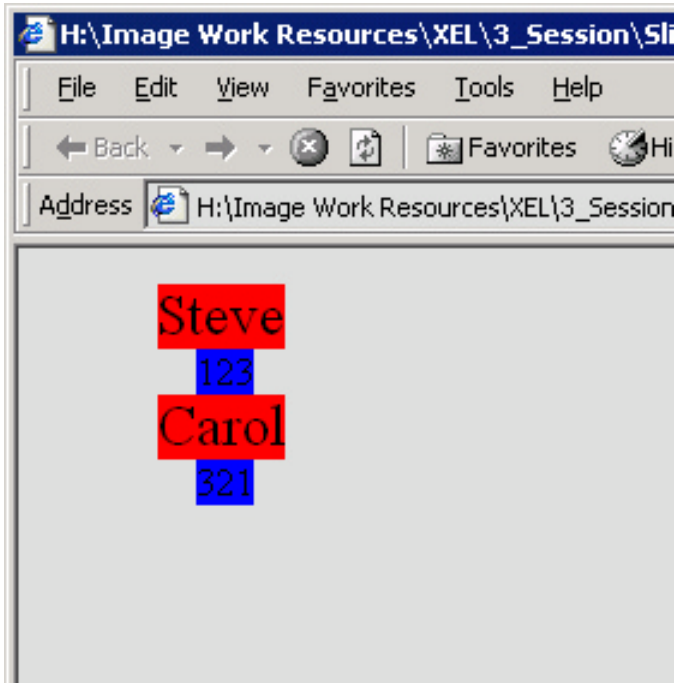
        <Address>
        …
```

Adress_Book.css

```
Address_Book Address Name
{
        display: block;
        margin-left: 2em;
        margin-right: 100%;
        font-size: 18pt;
        background-color:red;
}
Address_Book Address Phone
…
```

CSS - Example output

- Internet Explorer 5:



CSS - Selectors

- Group selectors:
  `x, y, z {rule;}`
- Universal selector:
  `* {rule;}`
- Type selector:
  `x {rule;}`
- Descendant selector:
  `x y {rule;}`
- Attribute selector:
  `x[attr] {rule;}`
  `x[attr="value"] {rule;}`
- Class selector:
  `.my {rule;}`
- ID selectors:
  `#value {rule;}`

Triggers:

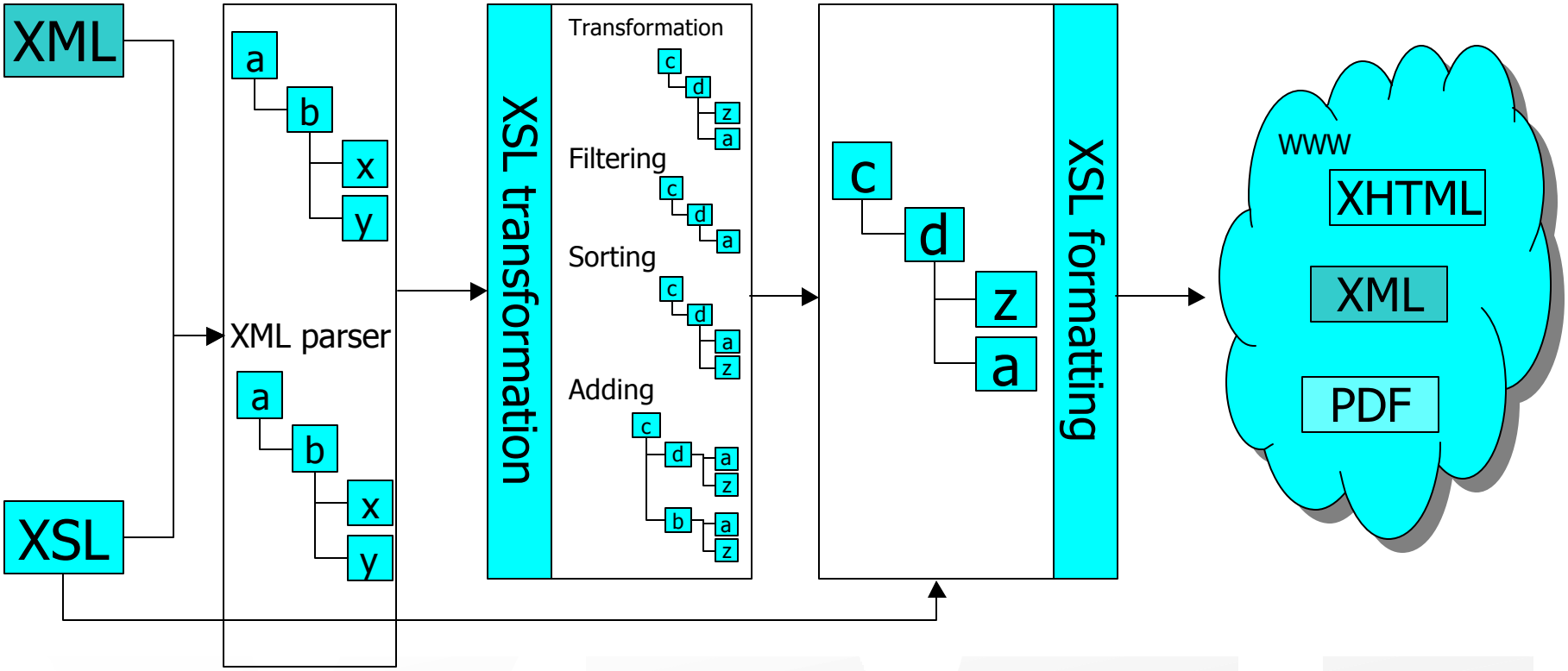
```
<x>abc</x> <y>def</y>
<anytag>abc</anytag>
<x>abc</x>
<x><y>abc</x></y>
<x attr="any">abc</x>
<x attr="value">abc</x>
<x class="my">abc</x>
<x id="value">abc</x>
```

CSS - Why not use that?

- CSS can style XML documents, but CSS can not…
  - Transform the document
  - Sort nodes
  - Reorder nodes
  - Add, remove nodes

CSS can also (or still) be used to render XHTML

XSL can...



XML XSL(T) → XML XHTML ...

XSL - Flexibility

- x to x
  - **One style sheet** can be used to transform/render **multiple XML documents**
  - **Multiple style sheets** can be used to transform/render one XML document into **multiple output files**
    i.e. XML to… XML, XHTML, PDF, …
- This can be done **server or client side**

XSL or XSLT?

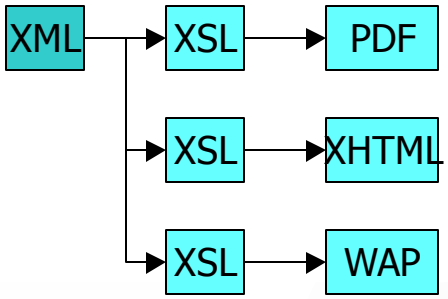
- XSL consists of two parts (W3C def.):
  - A language for **transforming** XML documents (XSLT)
  - A vocabulary for specifying how XML data should be **rendered**

  (This definition will be used in these slides)
- Different definitions: Some people say XSL and XSLT are separate, or rendering is a subset of XSLT (XML Boot camp)

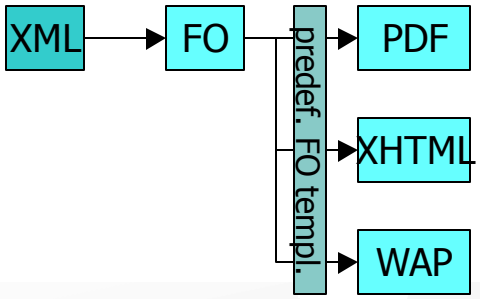
XSL - Surrounding technologies

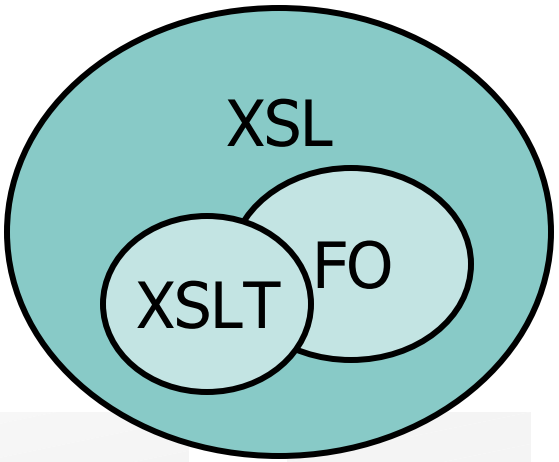
- XPath: used to select nodes from the XML document
- FO: formatting objects used to render into different formats

Without FO:



With FO:





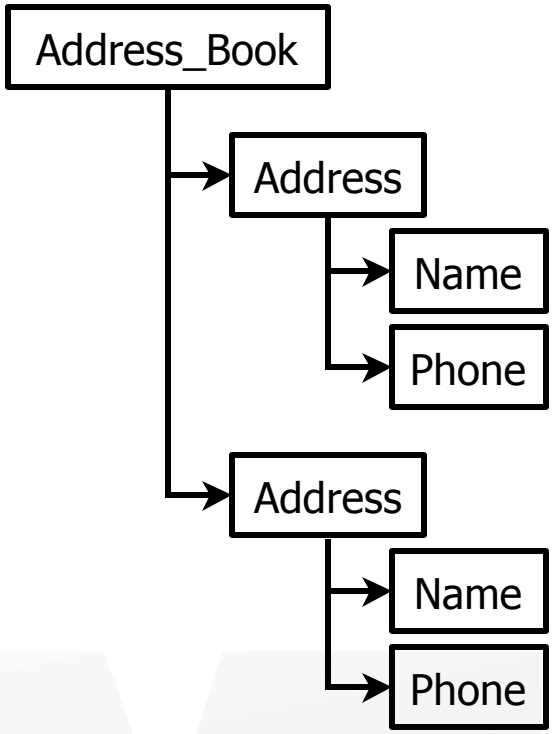
XSL - How does it work?

Address_Book_xsl.xml

```
<?xml version="1.0"?>
<?xml-stylesheet type="text/xsl"
href="Address_Book.xsl"?>
<Address_Book>
        <Address>
                <Name>Steve</Name>
                <Phone>123</Phone>
        </Address>

        <Address>
                <Name>Carol</Name>
                <Phone>321</Phone>
        </Address>
</Address_Book>
```

Trees!



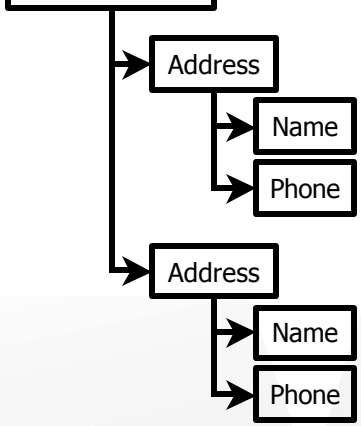
XSL - How does it work?

Adress_Book.xsl

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

<xsl:template match="Address_Book">
        <html>
                <body>
                        <xsl:apply-templates/>
                </body>
        </html>
</xsl:template>
...
```

XPath



Output:

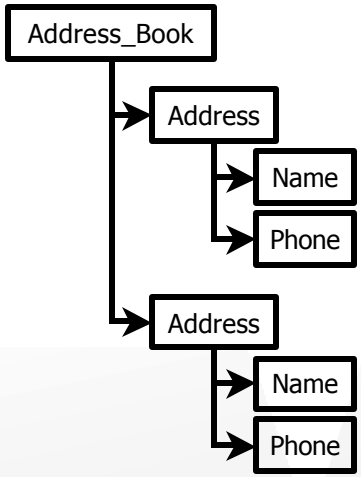
```
<html>
<body>
_
</body>
</html>
```

XSL - How does it work?

Adress_Book.xsl

```
…
<xsl:template match="Address">
        <xsl:apply-templates/>
</xsl:template>
…
```

XPath



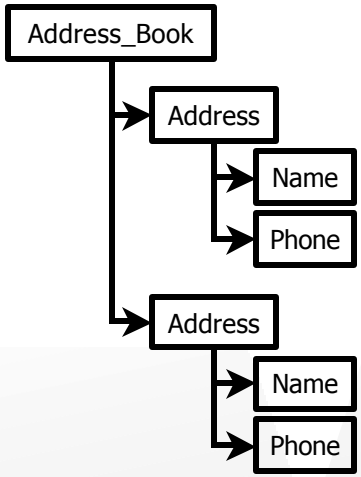
Output:

```
<html>
<body>
_
</body>
</html>
```

XSL - How does it work?

Adress_Book.xsl

```
…
<xsl:template match="Name">
        <font color="red">
                <xsl:value-of select="."/>
        </font><br/>
</xsl:template>
…
```



Output:

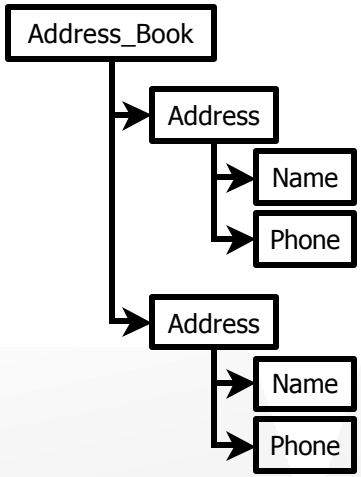
```
<html>
<body>
<font color="red">
Steve
</font><br/>
</body>
</html>
```

XSL - How does it work?

Adress_Book.xsl

```
…
<xsl:template match="Phone">
        <font color="blue">
                <xsl:value-of select="."/>
        </font><br/>
</xsl:template>

</xsl:stylesheet>
```



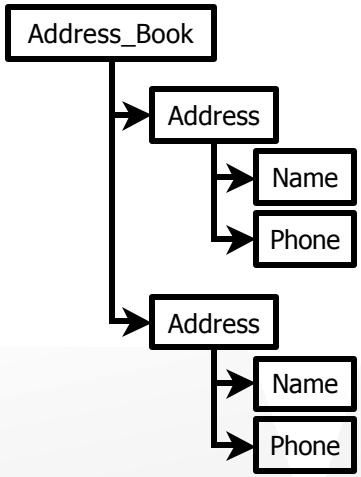
Output:

```
<html>
<body>
<font color="red">
Steve
</font><br/>
<font color="blue">
123
</font><br/>
_
</body>
</html>
```

XSL - How does it work?

Adress_Book.xsl

```
…
<xsl:template match="Address">
        <xsl:apply-templates/>
</xsl:template>
…
```



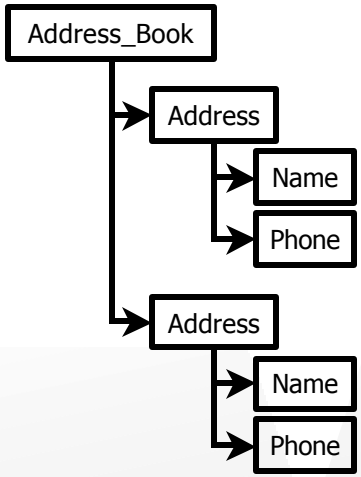
Output:

```
<html>
<body>
<font color="red">
Steve
</font><br/>
<font color="blue">
123
</font><br/>
</body>
</html>
```

XSL - How does it work?

Adress_Book.xsl

```
…
<xsl:template match="Name">
        <font color="red">
                <xsl:value-of select="."/>
        </font><br/>
</xsl:template>
…
```



Output:

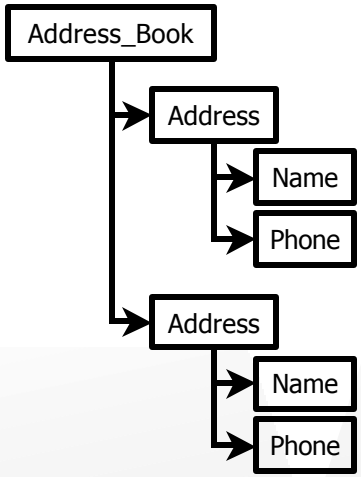
```
<html>
<body>
<font color="red">
Steve
</font><br/>
<font color="blue">
123
</font><br/>
<font color="red">
Carol
</font><br/>
</body>
</html>
```

XSL - How does it work?

Adress_Book.xsl

```
…
<xsl:template match="Phone">
        <font color="blue">
                <xsl:value-of select="."/>
        </font><br/>
</xsl:template>

</xsl:stylesheet>
```



Output:

```
<html>
<body>
<font color="red">
Steve
</font><br/>
<font color="blue">
123
</font><br/>
<font color="red">
Carol
</font><br/>
<font color="red">
Carol
</font><br/>
</body>
</html>
```

XSL - How does it work?

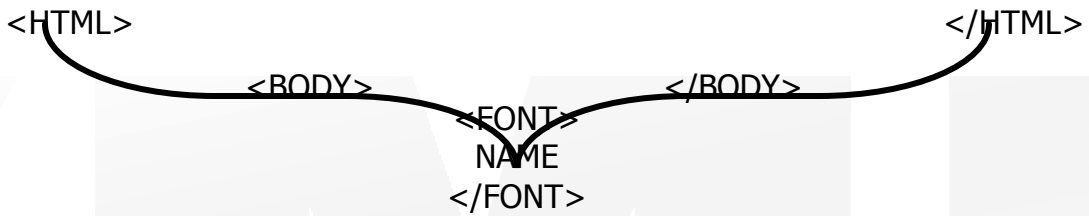
- We produced XHTML as output
- What do we do, if we want XML?
  Take a look at
  `Address_Book_xml.xsl`

Output:

```
<html>
<body>
<font color="red">
Steve
</font><br/>
<font color="blue">
123
</font><br/>
<font color="red">
Carol
</font><br/>
<font color="red">
Carol
</font><br/>
</body>
</html>
```

XSL - What does the engine?

- The XSL engine
  - Reads the tree it gets from the parser (built-in)
  - keeps track of where we stand in the document tree
  - Renders the output document
- The output is developed from the outside to the inside



XSL - Where do I get the engine?

- http://www.jclark.com
  - Provides XT, a Java XSL parser
  - \+ You can output .html files
- http://www.microsoft.com
  - Provides a browser with XSL parser
  - \- You cannot output .html files, only for viewing

The examples I will present work with both parsers

XSL - Elements and functions

- I will only explain some items
- For a complete list please see:
  http://www.w3.org/TR/xslt/

XSL - Elements

- Just a few examples

XSL - <xsl:value-of>

- You can select the values and attributes of your tags (as seen in previous example)
- `<xsl:value-of select="tag"/>`
  `<xsl:value-of select="@attribute"/>`
  `<xsl:value-of select="."/>`

XPath
For a complete list of XPath:
http://www.w3.org/TR/xpath

XSL - <xsl:for-each>

- Selects zero or more nodes for processing
- ```
  <xsl:template match="/">
  <xsl:for-each select="Address">
     …
  </xsl:for-each/>
  </xsl:template>
  ```
- So what is the difference to `<xsl:apply templates/>`?
  ```

XSL - <xsl:sort>

- Sorts by given argument
- Has to be nested inside <xsl:for-each>
-
```
<xsl:for-each select="node">
    <xsl:sort/>
    <xsl:value-of select="."/>
<xsl:for-each/>
```
- Check example `Address_Book_sort.xml`

XSL - <xsl:if>

- Conditional statement
- `<xsl:if test=".='Carol'">`

  `…`

  `</xsl:if>`
- Check example `Address_Book_if.xml`

XSL - <xsl:choose>

- Conditional statement
- <xsl:choose>
  <xsl:when test="…">…</xsl:when>
  <xsl:when test="…">…</xsl:when>
  …
  <xsl:otherwise>…</xsl:otherwise>
  </xsl:choose>

XSL - <xsl:variable>

- <xsl:template match="Name">
  <xsl:variable name="var"
  select="."/>
  <xsl:value-of select="**$var**"/>
  </xsl:template>
- A variable is write-once, read-many
- Check example Address_Book_var.xml

XSL - <xsl:number>

- Gives a list of elements consecutive numbers
-
```
<xsl:for-each select="Name">
    <xsl:number format="I"/>.
    <xsl:value-of select="."/><br/>
</xsl:for-each>
```
- Check example
  `Address_Book_number.xml`

Before I go to XSL Functions

- Let’s talk about XPath

XPath - Use

- <xsl:template match="xpath">
  <xsl:for-each select="xpath">
  <xsl:value of select="xpath">
  …

XPath - Examples

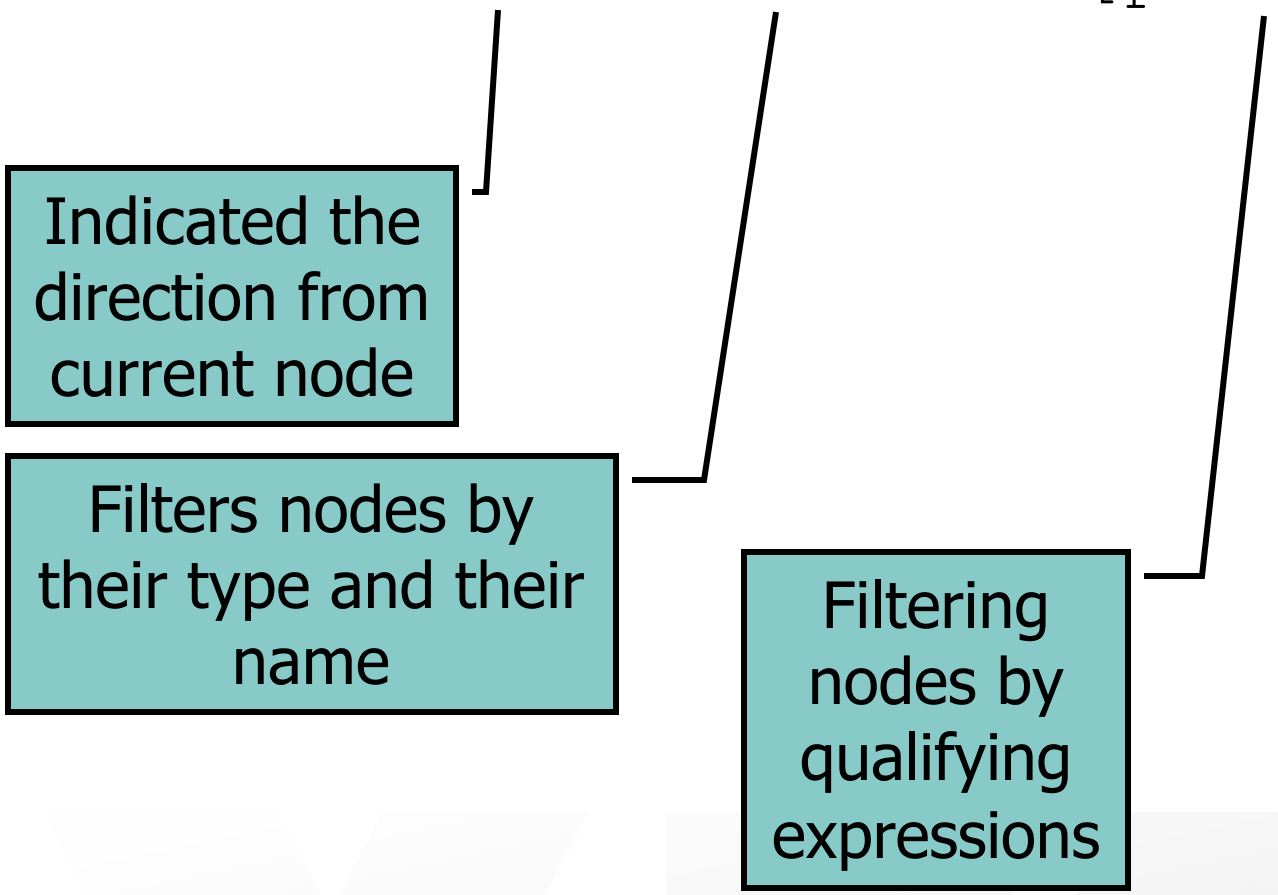
- `/Address/Name` - Name node selected
- `/Address[2]/Name` - Name in second Address node selected
- ...
- Also called XSL Patterns

XPath - Axis identifier

- With XPath you can go from the current location to:
  - Parent
  - Ancestor
  - Preceding-sibling
  - Following-sibling
  - Child
  - descendant

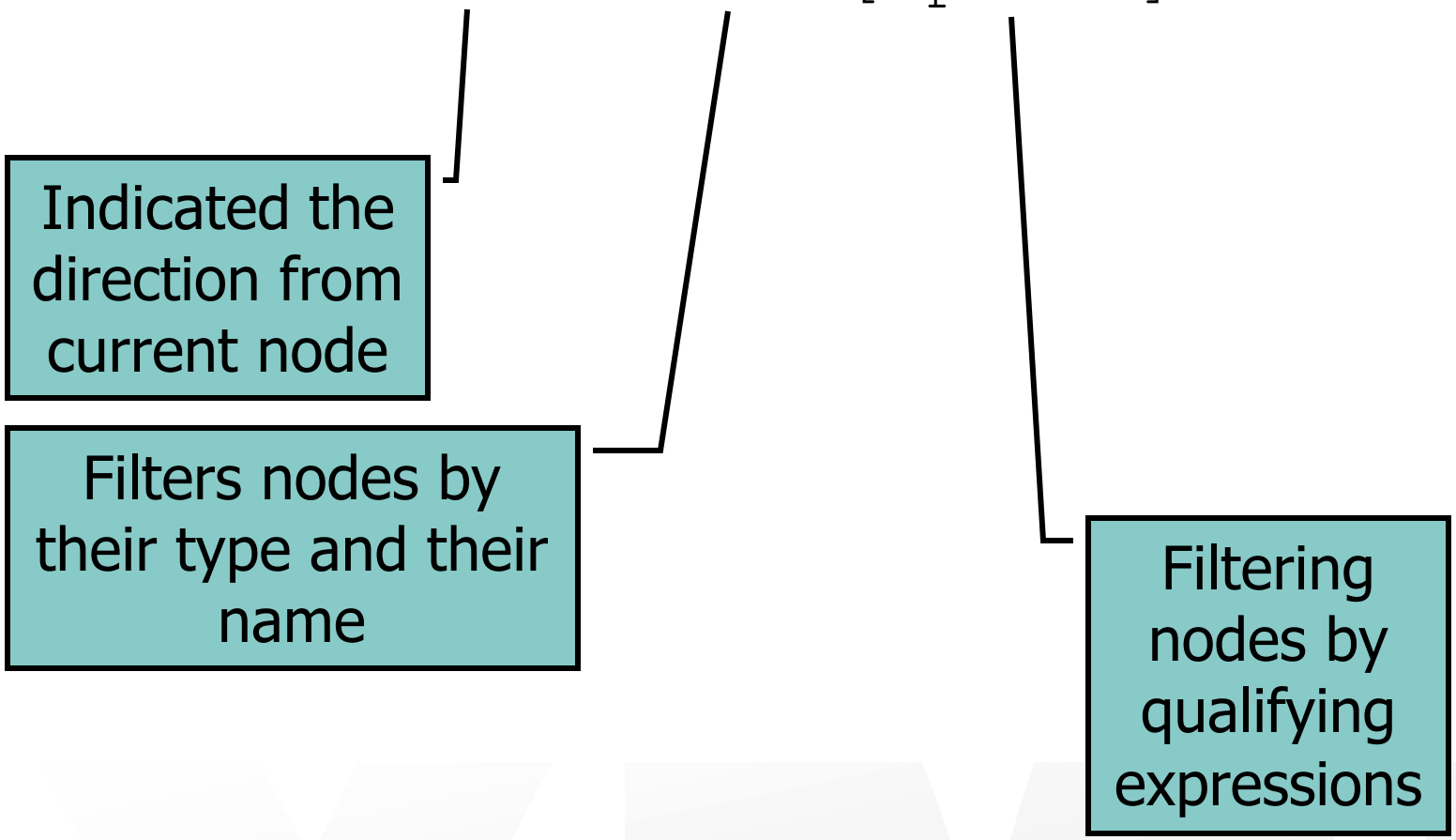
XPath - General Schema

- `select="axis::node test[predicate]"`



XPath - Advanced Example

- `select="child::name[@phone]"`



XPath - Continued

- Template Rule Conflict Resolution
- User assigned priorities

XSL - Functions

- Just a few examples

XSL - Date Type ConversionFunctions

- `<xsl:value-of select="format-number(25,'#0.00')"/>`
  Output: `25.00`
- `<xsl:value-of select="string(5.00)"/>`
  Output: `5` (string)

XSL - String functions

- <xsl:if test="**starts-with(node, 'Image')/>**
- <xsl:if test="**contains(node, 'Image')**/>
- <xsl:if test="**string-length(node) < 10**"/>
- <xsl:value-of select=
  "**translate(., "ABCDEFG", "abcdefg")**"/>

XSL - Additional functions

- Pull in nodes from another document with `document()`
- Enforcing a unique id (besides XSD) `id()`
- Mathematical functions / operators like `sum()`, `round()`, `floor()`, `+`, `div`, …and so on
- Boolean and rational operators `not`, `and`, `or`, `<`, `>`, `=`, `!=`, …and so on

Interesting sites

- http://www.ibm.com/developer/xml/
  Education link
  i.e. Transforming XML documents to PDF
  or XML programming in Java
- http://www.alphaworks.ibm.com
  XML dropdown

Freehand exercises

- Play aroung with XSL and the provided `juicers.xml`

Last session will cover

- XML APIs (SAX and DOM)
  - All done in Java
- XML and Databases
  - How to pull out and plug data into databases
- Outlook (XSQL, SVG, SOAP, …)