


SENG1050
Web Technologies

Lecture 7: XSLT


UNIVERSITY OF
NEWCASTLE
AUSTRALIA

Lecture Plan

Weekly program (lectures)

➔

Week 7 – XSLT

Week 1 – The Internet, Protocols, TCP/IP, Email, HTTP

Week 2 – HTML basics

Week 3 – XML and DTD

Week 4 – CSS

Week 5 – More HTML with CSS

Week 6 – Revision and Midterm


Week 8 – JavaScript

Week 9 – More JavaScript and User Interface

Week 10 – Encoding, Compression and Information Retrieval

Week 11 – Security and Encryption

Week 12 – Ethics and Course review


UNIVERSITY OF
NEWCASTLE
AUSTRALIA


Lecture Overview

• Introduction

• XML + XSLT


• XSLT

- Templates
- XPath
- Control structures



UNIVERSITY OF
NEWCASTLE
AUSTRALIA

Example XML file: beatles2.xml

```
<?xml version="1.0"?>
<beatles>
  <beatle link="http://www.paulmccartney.com">
    <name>
      <firstname>Paul</firstname>
      <lastname>McCartney</lastname>
    </name>
    <plays>bass</plays>
    <hand>left</hand>
  </beatle>
  <beatle link="http://www.johnlennon.com">
    <name>
      <firstname>John</firstname>
      <lastname>Lennon</lastname>
    </name>
    <plays>guitar</plays>
    <hand>right</hand>
  </beatle>
</beatles>
```

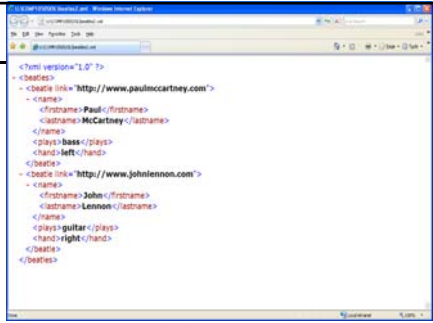

UNIVERSITY OF
NEWCASTLE
AUSTRALIA


Output obtained when using firefox on a Mac.




UNIVERSITY OF
NEWCASTLE
AUSTRALIA

Output obtained when using IE on a PC.




UNIVERSITY OF
NEWCASTLE
AUSTRALIA

XML looks boring!

- XML is a standard for the organisation of **well-defined data**
- XML does *not* attempt to present data in any visually appealing way
 - If you open an XML file in your web browser, all you will see is a tree-like structure of the tags.
- *XML is not about visual presentation – rather, it is about sharing of **well-defined data**.*

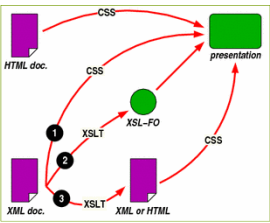


Motivation: Making XML Pretty

Question: How can we change a plain XML document into something more visually appealing such as HTML?

Answer: There are several languages that allow us to do this...

- CSS (only very briefly mentioned in this course)
- XSLT
- XSL-FO (only very briefly mentioned in this course)



XSL

- **XSL** stands for e**X**tensible **S**tylesheet **L**anguage.
 - CSS = Style Sheets for HTML (In HTML tags are predefined and styles can be added with CSS).
 - XSL = Style Sheets for XML (In XML we can make up our own tags. XSL specifies display of the XML document).
- XSL is more than a style sheet language. It consists of three parts:
 1. XSLT - a language for transforming XML documents into HTML documents or to other XML documents.
 2. XPath (XML Path Language) - a language for navigating in XML documents
 3. XSL-FO (XSL Formatting Objects) - a language for formatting XML documents (for print)



XSLT

- "T" stands for "Transformations"
- **XSLT** means **E**Xtensible **S**tylesheet **L**anguage **T**ransformations.
- For executing XSLT transformations (i.e. *reading the .xml and the .xsl file and producing an output document*), a specific parsing program or **XSLT processor** is required. These are integrated in most modern browsers.
 - Firefox, IE, Opera, and Chrome support XML, XSLT, and XPath.
 - Safari supports XML and XSLT



XSLT

- XSLT = **X**SL **T**ransformations
 - A way to transform an XML document into...
 - An HTML Webpage
 - A plain text document
 - A PDF document
 - A spreadsheet
 - Or anything else...
- <http://www.w3.org/TR/xslt>



Writing an XSLT Document

- In the XML file, add this line for referencing
 - `<?xml-stylesheet type="text/xsl" href="url"?>`
- A (minimal) XSLT document will look like this
 - `<?xml version="1.0" encoding="utf-8" ?>`
 - `<xsl:stylesheet version="1.0"`
 - `xmlns:xsl="http://www.w3.org/1999/XSL/Transform"`
 - `...`
 - `</xsl:stylesheet>`

prefix

HTML : XSL → tag name conflict
→ namespace



HTML Document in an XSL Template

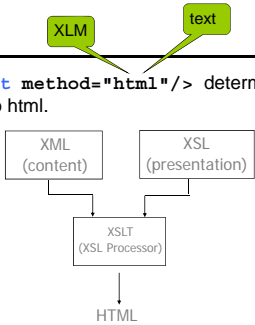
```
<?xml version="1.0" encoding="utf-8"?>
<xsl:stylesheet version="1.0"
  xmlns:xsl="http://www.w3.org/1999/XSL/Transform">
  <xsl:output method="html"/>

  <xsl:template match="/">
    <HTML>
      <HEAD>
        <TITLE>Welcome</TITLE>
      </HEAD>
      <BODY>
        Welcome!
      </BODY>
    </HTML>
  </xsl:template>

</xsl:stylesheet>
```

XSL → HTML

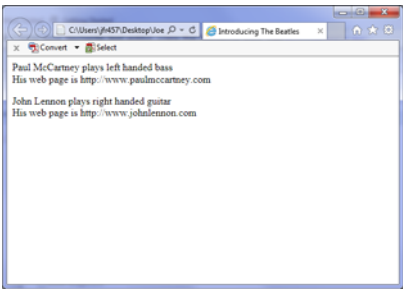
- `<xsl:output method="html"/>` determines that we transform into html.



try.xsl

```
<?xml version="1.0" encoding="utf-8"?>
<xsl:stylesheet version="1.0"
  xmlns:xsl="http://www.w3.org/1999/XSL/Transform">
  <xsl:output method="html"/>
  <xsl:template match="/beatles">
    <html>
      <head>
        <title>
          Introducing The Beatles
        </title>
      </head>
      <body>
        <p>
          <xsl:value-of select="name/firstname" />
          <xsl:value-of select="name/lastname" />
          <xsl:text> plays </xsl:text>
          <xsl:value-of select="hand" />
          <xsl:text> handed </xsl:text>
          <xsl:value-of select="plays" /> <br />
          <xsl:text>His web page is </xsl:text>
          <xsl:value-of select="link" />
        </p>
      </body>
    </html>
  </xsl:template>
</xsl:stylesheet>
```

```
<?xml version="1.0"?>
<xsl:stylesheet type="text/xsl" href="try.xsl">
  <beatles link="http://www.paulmccartney.com">
    <name>
      <firstname>Paul</firstname>
      <lastname>McCartney</lastname>
    </name>
    <plays>bass</plays>
    <hand>left</hand>
  </beatles>
  <beatles link="http://www.johnlennon.com">
    <name>
      <firstname>John</firstname>
      <lastname>Lennon</lastname>
    </name>
    <plays>guitar</plays>
    <hand>right</hand>
  </beatles>
</xsl:stylesheet>
```



```
<channels>
  <channel>
    :
    <logolmage> channel_logo.jpg </logolmage>
    <profileimage> mira_profile_photo.png </profileimage>
    <subscribeURL> http://www.youtube.com/channel/myChannel </subscribeURL>
  </channel>
  :
  <channel>
    :
    <channels>=====
    <html>
      :
      <img alt="Picture not found">
        <xsl:attribute name="src">
          <xsl:value-of select="channels/channel/logolmage"/>
        </xsl:attribute>
      </img>
      <img alt="Picture not found">
        <xsl:attribute name="src">
          <xsl:value-of select="channels/channel/profileimage"/>
        </xsl:attribute>
      </img>
      <a href="{channels/channel/subscribeURL}"><xsl:value-of select="channels/channel/subscribeURL"/></a>
    </html>
  </channel>
</channels>
```

```
<html>
:
<img alt="Picture not found">
  <xsl:attribute name="src">
    <xsl:value-of select="channels/channel/logolmage"/>
  </xsl:attribute>
</img>
<img alt="Picture not found">
  <xsl:attribute name="src">
    <xsl:value-of select="channels/channel/profileimage"/>
  </xsl:attribute>
</img>
<a href="{channels/channel/subscribeURL}"><xsl:value-of select="channels/channel/subscribeURL"/></a>
:
</html>

The code above is interpreted by browsers as follows:
<html>
:


<a href="http://www.youtube.com/channel/myChannel"> http://www.youtube.com/channel/myChannel </a>
:
</html>
```

XSL strcuture

```
• XSL is a collection of templates

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

  <xsl:template match="/">
    [content]
  </xsl:template>
  <xsl:template match="name">
    [content]
  </xsl:template>
  <xsl:template match="play">
    [content]
  </xsl:template>
  ...
</xsl:stylesheet>
```

XSL Elements

- Matching and Selecting Templates
 - xsl:template
 - xsl:apply-template
 - xsl:value-of
 - xsl:sort
- Text Elements
 - xsl:text
- Branching Elements (Flow Control)
 - xsl:for-each
 - xsl:if
 - xsl:choose

XSLT → xsl:template

XSL contains a set of “templates”

```
<xsl:template match="pattern">
  ... Template content ...
</xsl:template>
```

- The **templates** are the **most important components** of a XSLT document.
- The attribute **match** specifies which elements of the XML document will be affected by this template.
- The **Template content** inside the **<xsl:template>** element defines some **HTML** to write to the output.

XSLT → xsl:template

XSL contains a set of “templates”

```
<xsl:template match="pattern">
  ... Template content ...
</xsl:template>
```

- XSLT engine **traverses** XML document looking for tags that match **pattern**
- The **pattern** syntax is a subset of **XPath** expression

XSLT and XPath

- XPath provides a way to **navigate** through elements and attributes in XML documents.
- XSLT uses XPath to find **information** in an XML document.
- XPath contains over 100 built-in functions

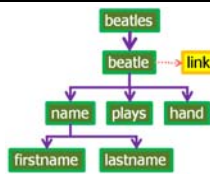
XSLT

```
<?xml version="1.0"?>
<?xml-stylesheet type="text/xsl" href="try.xsl">
<beatles>
  <beatle link="http://www.paulmccartney.com">
    <name>
      <firstname>Paul</firstname>
      <lastname>McCartney</lastname>
    </name>
    <plays>bass</plays>
    <hand>left</hand>
  </beatle>
  <beatle link="http://www.johnmccartney.com">
    <name>
      <firstname>John</firstname>
      <lastname>McCartney</lastname>
    </name>
    <plays>guitar</plays>
    <hand>right</hand>
  </beatle>
</beatles>
```

```
graph TD
    beatles[beatles] --> beatle1[beatle link="http://www.paulmccartney.com"]
    beatles --> beatle2[beatle link="http://www.johnmccartney.com"]
    beatle1 --> name1[name]
    beatle1 --> plays1[plays]
    beatle1 --> hand1[hand]
    beatle2 --> name2[name]
    beatle2 --> plays2[plays]
    beatle2 --> hand2[hand]
    name1 --> firstname1[firstname]
    name1 --> lastname1[lastname]
    name2 --> firstname2[firstname]
    name2 --> lastname2[lastname]
```

beatle is a child element of the beatles element. name, plays and hand are children elements of the beatle element.

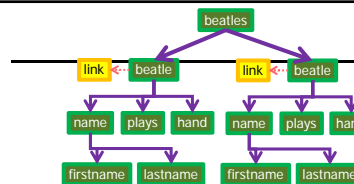
XPath



<beatles> :root element node
<beatle>, <name> .. <hand> :element node
link = * :attribute node

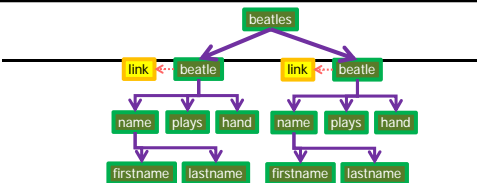
Parent, children, siblings, ancestors, descendants

expression	Description
nodename	Select all nodes with the name "nodename"
/	Select from the root node
//	Select nodes in the XML from the current node that match the selection no matter where they are
.	Select the current node
..	Select the parent of the current node
@	Select attributes

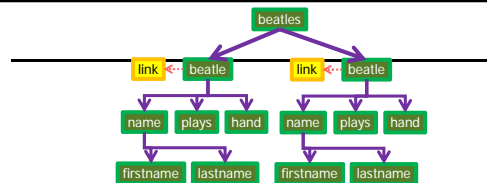


expression	Description
nodename	Select all nodes with the name "nodename"
/	Select from the root node
//	Select nodes in the XML from the current node that match the selection no matter where they are
.	Select the current node
..	Select the parent of the current node
@	Select attributes

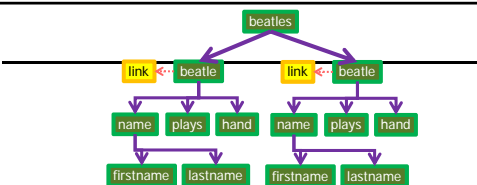
expression	Description
hand	Select all nodes with the name "hand"
/beatles	Select from the root element beatles
//beatle	Select all beatle elements no matter where they are
beatles/beatle	Select all beatle elements that are children of beatles
//@link	Select all attributes that are named link



Path expression	Result
/beatles/beatle[1]	Select the first beatle element
/beatles/beatle[last()]	Select the last beatle element
/beatles/beatle[position() < 3]	Select the first two beatle elements
/beatle[@link]	Select all the beatle elements that have an attribute named link
//beatle[@link='newcastle.edu.au']	Select all beatle elements that have an attribute named link with a value of 'newcastle.edu.au'



wildcard	Description
*	Matches any element node
@*	Matches any attribute node
/beatles/*	Select all the child nodes of the beatles
//beatle[@*]	Select all beatle elements which have any attribute



Built-in functions	Description
name()	The name of the matched tag
text()	The text of the match
position()	Return the index position of the node
last()	return the number of items



XSLT and Xpath: How do they work together?

- XSLT uses
 - XPath to define parts of the source document
 - that **should match** one or more **predefined templates**.
- When a match is found,
 - XSLT **will transform** the matching part of the source document into the **result document**.



```
<?xml version="1.0" encoding="utf-8"?>
<xsl:stylesheet version="1.0"
  xmlns:xsl="http://www.w3.org/1999/XSL/Transform">
  <xsl:output method="html"/>
  <xsl:template match="//beatles">
    <html>
      <head>
        <title>
          Introducing The Beatles
        </title>
      </head>
      <body>
        <p>
          <xsl:value-of select="name/firstname" />
          <xsl:value-of select="name/lastname" />
          <xsl:text> plays </xsl:text>
          <xsl:value-of select="hand" />
          <xsl:text> handed </xsl:text>
          <xsl:value-of select="plays" /> <br />
          <xsl:text>His web page is </xsl:text>
          <xsl:value-of select="@link" />
        </p>
      </body>
    </html>
  </xsl:template>
</xsl:stylesheet>
```

31

XSL Stylesheet Elements

- Matching and Selecting Templates
 - xsl:template
 - **xsl:apply-template**
 - xsl:value-of
 - xsl:sort
- Text Elements
 - xsl:text
- Branching Elements
 - xsl:for-each
 - xsl:if
 - xsl:choose

32

XSLT apply- templates

XSLT offers an option to call only selected templates:

Use

```
<xsl:apply-templates select="subpath" />
```

and restrict which template to call using the Xpath expression **"subpath"**

33

```
<?xml version="1.0" encoding="UTF8"?>
<xsl:stylesheet version="1.0"
  xmlns:xsl="http://www.w3.org/1999/XSL/Transform">
  <xsl:output method="html"/>
  <xsl:template match="beatles">
    <html>
      <body>
        <xsl:apply-templates select="beatle/name"/>
      </body>
    </html>
  </xsl:template>

  <xsl:template match="name">
    <h2>Beatles !</h2>
  </xsl:template>
</xsl:stylesheet>
```

```
<html>
<body>
<h2>Beatles !</h2>
<h2>Beatles !</h2>
</body>
</html>
```

34

```
<?xml version="1.0" encoding="UTF8"?>
<xsl:stylesheet version="1.0"
  xmlns:xsl="http://www.w3.org/1999/XSL/Transform">
  <xsl:output method="html"/>
  <xsl:template match="beatles">
    <html>
      <body>
        <xsl:apply-templates select="beatle/name"/>
      </body>
    </html>
  </xsl:template>

  <xsl:template match="name">
    <h2>Beatles !</h2>
  </xsl:template>
</xsl:stylesheet>
```

```
<html>
<body>
<h2>Beatles !</h2>
<h2>Beatles !</h2>
</body>
</html>
```

35

XSLT

```
<xsl:stylesheet xmlns:xsl="http://www.w3.org/1999/XSL/Transform" version="1.0">
  <xsl:template match="//document/*">
    <body>
      <xsl:apply-templates select="name" />
      <xsl:apply-templates select="dob" />
      <xsl:apply-templates select="age" />
    </body>
  </xsl:template>

  <xsl:template match="name">
    <p><xsl:value-of select="name" /></p>
  </xsl:template>

  <xsl:template match="dob">
    <p>DOB: <xsl:value-of select="dob" /></p>
  </xsl:template>

  <xsl:template match="age">
    <p>AGE: <xsl:value-of select="age" /></p>
  </xsl:template>
</xsl:stylesheet>
```

xsl:apply-template applied like this and they are used by select option

36

XSL Stylesheet Elements

- Matching and Selecting Templates
 - xsl:template
 - xsl:apply-template
 - **xsl:value-of**
 - xsl:sort
- Text Elements
 - xsl:text
- Branching Elements
 - xsl:for-each
 - xsl:if
 - xsl:choose



XSLT Processing <xsl:value-of>

- <xsl:value-of **select**="xpath-expression"/>
- The XSL element <xsl:value-of> can be used to extract **the value** of an element that is selected from the source XML document
- The selected element is located by an XPath expression that appears as the value of the **select** attribute



XSLT Processing <xsl:value-of>

- The command **<xsl:value-of>** allows to select values using the attribute **select**
 - In our XML example

```
<name>
  <firstname>Paul</firstname>
  <lastname>McCartney</lastname>
</name>
```
 - In the .xsl file we use the appropriate **XPath** expression to select the attribute **firstname**

```
<xsl:value-of select="name/firstname" />
```
 - As result "Paul" and "John" will be copied from the .xml source file to the **output (?)**.



XSLT Processing <xsl:value-of>

The <xsl:value-of> element is used to output the text value of a node.

```
<?xml version="1.0" encoding="UTF-8"?>
<xsl:stylesheet version="1.0"
  xmlns:xsl="http://www.w3.org/1999/XSL/Transform">
<xsl:output method="text"/>
<xsl:template match="name">
  We found a first name and it's <xsl:value-of select="firstname"/>
</xsl:template>
</xsl:stylesheet>
```



Example XML file

```
<bibliography>
  <book>
    <author>
      <givenname>Homer</givenname>
      <surname>Simpson</surname>
    </author>
    <title>"Homer's Odyssey"</title>
    <year>2006</year>
    <pub>Springer Verlag</pub>
  </book>
</bibliography>
```



XSLT Template Example 1

```
<xsl:template match="/bibliography">
  <html>
    <head>
      <title>Bookstore Example 1</title>
    </head>
    <body>
      <h2>Bookstore Example 1</h2>
      <p>This is simply matching the title of the first book.</p>
      <table border="1">
        <tr><th>Title</th></tr>
        <tr><td><xsl:value-of select="book/title"/></td></tr>
      </table>
    </body>
  </html>
</xsl:template>
```

```
<book>
  <author>
    <givenname>Homer</givenname>
    <surname>Simpson</surname>
  </author>
  <title>"Homer's Odyssey"</title>
  <year>2006</year>
  <pub>Springer Verlag</pub>
</book>
```






XSLT Template Example 2

```
<xsl:template match="book">
<p>
  <xsl:value-of select="author/givenname" />
  <xsl:text> </xsl:text>
  <span class="surname">
    <xsl:value-of select="author/surname" />
  </span>
  <span class="title">
    <xsl:value-of select="title" />
  </span>
  <span class="pub">
    <xsl:value-of select="pub" />
  </span>
</p>
</xsl:template>
```


```
<book>
  <author>
    <givenname>John</givenname>
    <surname>Muir</surname>
  </author>
  <title>Homer's Odyssey</title>
  <pub>The University of Newcastle</pub>
</book>
```



44

XSLT Processing: Inserting text

- XSLT ignores whitespace before and after its tags
- `<xsl:value-of select="author/givenname"/> <xsl:value-of select="author/surname" />` would place the names without a space in between
- `<xsl:text>` lets you insert text, including spaces
- `<xsl:value-of select="author/givenname"/> <xsl:text> </xsl:text> <xsl:value-of select="author/surname" />`
- Within HTML tags, normal spacing rules apply
 - (Week4 Lecture: slide 63: white-space)



45

XSLT Processing

- ThXSLT


```
<xsl:stylesheet xmlns:xsl="http://www.w3.org/1999/XSL/Transform" version="1.0">
  <xsl:output method="text" />
  <xsl:template match="/">
    <xsl:apply-templates select="//BBB" />
  </xsl:template>
  <xsl:template match="BBB">
    <xsl:text>
      <xsl:value-of select="position()" />
      <xsl:text>]: </xsl:text>
      <xsl:value-of select="*" />
    </xsl:template>
</xsl:stylesheet>
```

XML (open client-side version)

```
<AAA>
  <BBB>10 </BBB>
  <BBB>5 </BBB>
  <BBB>7 </BBB>
</AAA>
```

Output

```
BBB[1]: 10
BBB[2]: 5
BBB[3]: 7
```



46

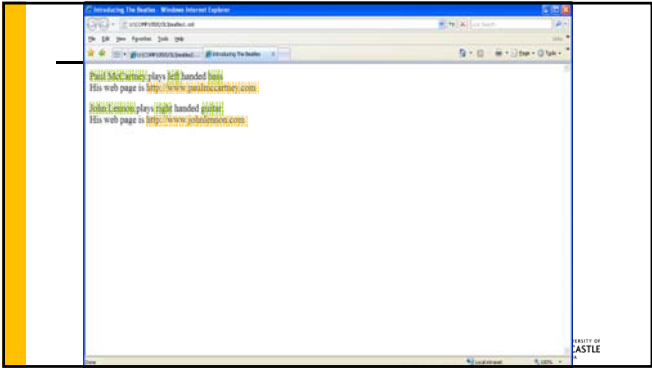
try.xsl

```
<?xml version="1.0"?>
<xsl:stylesheet version="1.0"
  xmlns:xsl="http://www.w3.org/1999/XSL/Transform">
  <xsl:output method="html"/>
  <xsl:template match="//beatle">
    <html>
      <head>
        <title>Introducing The Beatles</title>
      </head>
      <body>
        <p>
          <xsl:value-of select="name/firstname" />
          <xsl:text> </xsl:text>
          <xsl:value-of select="name/lastname" />
          <xsl:text> plays </xsl:text>
          <xsl:value-of select="hand" />
          <xsl:text> handed </xsl:text>
          <xsl:value-of select="plays" /> <br />
          <xsl:text>His web page is </xsl:text>
          <xsl:value-of select="@link" />
        </p>
      </body>
    </html>
  </xsl:template>
</xsl:stylesheet>
```

```
<beatles>
  <beatle link="http://www.paulmccartney.com">
    <name>
      <firstname>Paul</firstname>
      <lastname>McCartney</lastname>
    </name>
    <plays>bass</plays>
    <hand>left</hand>
  </beatle>
</beatles>
```



47



XSL Stylesheet Elements

- Matching and Selecting Templates
 - xsl:template
 - xsl:apply-template
 - xsl:value-of
 - xsl:sort
- Text Elements
 - xsl:text
- Branching Elements
 - xsl:if
 - xsl:for-each
 - xsl:choose



XSLT Processing: <xsl:if>

A conditional test against the content of the XML file.

Choosing: yes or no

```
<xsl:if test="expr">
    ... some output if expr is true ...
</xsl:if>
```

- The condition appears as the value of the **test** attribute.
- If the condition is true the contents inside the <xsl:if> are processed, for example:

```
<xsl:if test="price > 10">
    some output ...
</xsl:if>
```



XSLT Processing: <xsl:if>

- if the value of the test attribute is just an Xpath, then the test is satisfied if the nodelist of this Xpath expression is not empty.

XSLT	XML
<pre><xsl:stylesheet xmlns:xsl="http://www.w3.org/1999/XSL" transform="L1" > <xsl:output method="text" /> <xsl:template match="/" > <xsl:text> <xsl:if test="//BOOK[2] &lt; /> //BOOK[3]" > <xsl:text> 10 &lt; /> 5 </xsl:text> </xsl:if> <xsl:if test="//BOOK[1] = //BOOK[2]" > <xsl:text> AAAAAAAA </xsl:text> </xsl:if> </xsl:template> </xsl:stylesheet></pre>	<pre><!-- (open client-side version) --> <AAA> <BBB>10 </BBB> <BBB>5 </BBB> <BBB>7 </BBB> </AAA></pre> <p>Output</p> <p>10 < 5</p>



XSL Stylesheet Elements

- Matching and Selecting Templates
 - xsl:template
 - xsl:apply-template
 - xsl:value-of
 - xsl:sort
- Text Elements
 - xsl:text
- Branching Elements
 - xsl:if
 - xsl:choose
 - xsl:for-each



XSLT Processing: <xsl:choose>

XSLT	XML
<p>The <xsl:choose> element is used to perform conditional tests.</p> <p>Choosing</p> <pre><xsl:choose> <xsl:when test="..."> ... </xsl:when> <xsl:when test="..."> ... </xsl:when> <xsl:otherwise> ... </xsl:otherwise> </xsl:choose></pre>	<pre><!-- (open client-side version) --> <AAA> <BBB>10 </BBB> <BBB>5 </BBB> <BBB>7 </BBB> </AAA></pre> <p>Output</p> <p>otherwise test=5 test=7</p>



XSL Stylesheet Elements

- Matching and Selecting Templates
 - xsl:template
 - xsl:apply-template
 - xsl:value-of
 - xsl:sort
- Text Elements
 - xsl:text
- Branching Elements
 - xsl:if
 - xsl:choose
 - xsl:for-each



XSLT Processing <xsl:for-each>

<xsl:for-each> is a kind of loop statement
<xsl:for-each> element can be used to select every XML element of a specified node-set

Looping

```
<xsl:for-each select="subpattern">
  content ...
</xsl:for-each>
```

- Process the content of **xsl:for-each** for each such selected element under the current match



XSLT Processing <xsl:for-each>

The XSL of a spec

Looping

```
<xsl:
...
</xs
- Proc
unc
```

(open client-side version)

```
<AAA>
<BBB>CC </BBB>
<BBB>FF </BBB>
<BBB>AA </BBB>
<BBB>FFF </BBB>
<BBB>AA </BBB>
<BBB>CCCC </BBB>
</AAA>
```

Example 31-2: An XSL + XSLT >>>

See [XSLT](#) for up-to-date

Output

```
CC
FF
AA
FFF
AA
CCCC
```

ed element



XSLT Loops: for-each

Sample XML

```
<zoo>
<birds>
<albatross pop="4" />
<buzzard pop="2" />
<chickadee pop="12" />
</birds>
<mammals>
<aardvark pop="5" />
<bat pop="200" />
<cheetah pop="2" />
</mammals>
</zoo>
```

Sample XSLT → HTML

```
<xsl:stylesheet version="1.0" xmlns:xsl="
http://www.w3.org/1999/XSL/Transform">
<xsl:output method="html" />
<xsl:template match="/zoo">
<html><head><title>Zoo</title></head>
<body>
<xsl:for-each select="*">
<h1>
<xsl:value-of select="name()" />
</h1>
</xsl:for-each>
</body></html>
</xsl:template>
</xsl:stylesheet>
```



XSLT Loops: for-each

Sample XML

```
<zoo>
<birds>
<albatross pop="4" />
<buzzard pop="2" />
<chickadee pop="12" />
</birds>
<mammals>
<aardvark pop="5" />
<bat pop="200" />
<cheetah pop="2" />
</mammals>
</zoo>
```

Result HTML

birds

mammals



XSLT Loops: for-each

Sample XML

```
<zoo>
<birds>
<albatross pop="4" />
<buzzard pop="2" />
<chickadee pop="12" />
</birds>
<mammals>
<aardvark pop="5" />
<bat pop="200" />
<cheetah pop="2" />
</mammals>
</zoo>
```

Sample XSLT → HTML

```
<xsl:stylesheet version="1.0" xmlns:xsl="
http://www.w3.org/1999/XSL/Transform">
<xsl:output method="html" />
<xsl:template match="/zoo">
<html><head><title>Zoo</title></head>
<body>
<xsl:for-each select="*">
<h1><xsl:value-of select="name()" /></h1>
<ul>
<xsl:for-each select="*">
<li><xsl:value-of
select="name()" /> (<xsl:value-
of select="@pop"/>)</li>
</xsl:for-each>
</ul>
</xsl:for-each>
</body></html>
</xsl:template>
</xsl:stylesheet>
```



XSLT Loops: for-each

Sample XML

```
<zoo>
<birds>
<albatross pop="4" />
<buzzard pop="2" />
<chickadee pop="12" />
</birds>
<mammals>
<aardvark pop="5" />
<bat pop="200" />
<cheetah pop="2" />
</mammals>
</zoo>
```

birds

- albatross (4)
- buzzard (2)
- chickadee (12)

mammals

- aardvark (5)
- bat (200)
- cheetah (2)



XSLT Example

```
<xsl:for-each select="author">
  <xsl:choose>
    <xsl:when test='givenname!=""'>
      <xsl:value-of select="givenname" />
      <xsl:text> </xsl:text>
    </xsl:when>
    <xsl:when test='nickname!=""'>
      <xsl:value-of select="nickname" />
      <xsl:text> </xsl:text>
    </xsl:when>
    <xsl:choose>
      <span class="surname">
        <xsl:value-of select="surname" />
      </span>
      <xsl:text>, </xsl:text>
    </xsl:choose>
  </xsl:for-each>
```



XSL Stylesheet Elements

- Matching and Selecting Templates
 - xsl:template
 - xsl:apply-template
 - xsl:value-of
 - xsl:sort
- Text Elements
 - xsl:text
- Branching Elements
 - xsl:for-each
 - xsl:if
 - xsl:choose



XSLT Processing <xsl:sort>

- The <xsl:sort> element is used to sort the output.
- Sorting is achieved by adding xsl:sort elements as children of <xsl:apply-templates> or <xsl:for-each>

Sorting

```
<xsl:sort select="subpattern"
data-type="text | number"
order="ascending | descending"
case-order="upper-first | lower-first" />
```



XSLT Example

```
<xsl:apply-templates select="author">
  <xsl:sort select="surname" />
  or
  <xsl:sort select="givenname" />
</xsl:apply-templates>
```

http://www.w3schools.com/xsl/tryxslt.asp?xmlfile=cdcatalog&xsltfile=tryxslt_sort



XSLT Processing <xsl:sort>

XML

```
<zoo>
  <birds>
    <albatross pop="4" />
    <buzzard pop="2" />
    <chickadee pop="12" />
  </birds>
  <mammals>
    <aardvark pop="5" />
    <bat pop="200" />
    <cheetah pop="2" />
  </mammals>
</zoo>
```

XSLT fragment

```
<xsl:template match="birds | mammals">
  <ul>
    <xsl:for-each select="*">
      <xsl:sort select="@pop" order="descending"
data-type="number" />
      <li><xsl:value-of select="name(.)" />
        (<xsl:value-of select="@pop"/>)
      </li>
    </xsl:for-each>
  </ul>
</xsl:template>
```



XSLT Processing <xsl:sort>

XML

```
<zoo>
  <birds>
    <albatross pop="4" />
    <buzzard pop="2" />
    <chickadee pop="12" />
  </birds>
  <mammals>
    <aardvark pop="5" />
    <bat pop="200" />
    <cheetah pop="2" />
  </mammals>
</zoo>
```

birds

- chickadee (12)
- albatross (4)
- buzzard (2)

mammals

- bat (200)
- aardvark (5)
- cheetah (2)



XSL Stylesheet Elements

- Matching and Selecting Templates
 - xsl:template
 - xsl:apply-template
 - xsl:value-of
 - xsl:sort
- Text Elements
 - xsl:text
- Branching Elements
 - xsl:for-each
 - xsl:if
 - xsl:choose



XSLT Processing: A few more things

- Context functions

Name	Description
Position()	Returns the index position of the node that is currently being processed. Eg. //book[position() <= 3] Res: selects the first three book elements
Last()	Returns the last element of the items in the processed node list Eg. //book[last()] Res: selects the last book element
Count(node-set)	Returns the number of nodes in the node-set argument



XSLT Processing: A few more things

- Functions on Strings

Name	Description
string(arg)	Returns the string value of the argument. Eg. string(314) Res: "314"
compare(cmp1, cmp2)	Returns -1 if cmp1 less than (<) cmp2, 0 if cmp1 is equal to (=) cmp2, or 1 if cmp1 is greater than (>) cmp2.
concat(string, string, ...)	Returns the concatenation of the strings Eg. Concat('Xpath ', 'is ', 'FUN!') Res: 'Xpath is FUN!'
upper-case(string)	Converts the string argument to upper-case
lower-case(string)	Converts the string argument to lower-case



XSLT Processing: A few more things

- Functions on sequences

Name	Description
Index-of((item, item, ...), searchitem)	Returns the positions within the sequence of items that are equal to the searchitem argument. eg. Index-of((15, 40, 25, 40, 10), 40) Result: (2,4) Eg. Index-of(("a","dog","and","a"), "a") Result: (1,4)

- Functions on Nodes

Name	Description
name()	Returns the name of the current node or the first node in the specified node set
name(nodeset)	



Credits

- New Perspectives: HTML, XHTML and XML by Patrick Carey, Course Technology, 2010
<http://www.cengage.com/c/newperspectives/>
- Web Developer Foundations: Using XHTML by Terry Felke-Morris, Addison Wesley, 2005 –Chapter 7
- Web Technologies. A Computer Science Perspective by Jeffrey C. Jackson, Pearson Education, 2007.
 - <http://www.w3.org/XML/>
 - <http://www.w3.org/TR/xmlschema-0/>
 - <http://www.w3.org/TR/xslt>
 - <http://www.w3.org/TR/xpath>
 - <http://www.w3schools.com/xpath/>
 - <http://www.w3schools.com/xsl/>
 - <http://www.zvon.org/xsl/XPathTutorial/Output/example1.html>
 - <http://www.zvon.org/xsl/NamespacesTutorial/Output/example1.html>

