XSL Variables

Martin Holmes

XSL Variables

- XSL variables allow you to store values (strings, integers etc.) so that you can re-use them easily.
- XSL variables are NOT VARIABLE. Once you set the value, you're stuck with it.
- This is different from other programming languages, where variables can have different values assigned to them.

Creating an XSL variable

```
<xsl:variable name="uvicName"
              select="'University of Victoria'" />
```

- The required `@name` attribute gives us a way to refer to the variable later, using a dollar sign: `$uvicName`.
- The optional `@select` attribute specifies a value that is assigned to the variable. More on this later.

Assigning a value to your variable

- A variable can hold lots of different value types, either hard-coded (as in the example above) or assigned through XPath expressions. Here are some examples:
- A hard-coded integer:

```
<xsl:variable name="myAge"
              select="53" />
```

- The result of a calculation:

```
<xsl:variable name="numCards"
              select="13 * 4 + 2" />
```

- A hard-coded string (note the single quotes inside the double quotes):

```
<xsl:variable name="myName"
              select="'Martin Holmes'" />
```

- An element from the input document, found through XPath:

```
<xsl:variable name="docTitle"
              select="/TEI/teiHeader/titleStmt/title[1]" />
<xsl:variable name="docAuthor"
              select="/TEI/teiHeader/titleStmt/author[1]" />
```

- A sequence of elements from the input document, found through XPath:

```
<xsl:variable name="docAuthors"
              select="/TEI/teiHeader/fileDesc/titleStmt/author" />
```

How to use an XSL variable (1)

- If your variable contains an atomic value such as a string or a number, you can output it with `<xsl:value-of>`:

```
My name is <xsl:value-of select="$myName"/>.
```

- You can do XPath calculations with the value of your variable:

```
In ten years I shall be <xsl:value-of select="$myAge + 10"/>.
```

- If your variable contains an element, you can treat it just like an element.

```
This book was written by
<xsl:value-of select="$docAuthor/persName/forename"/>
<xsl:value-of select="$docAuthor/persName/surname"/>.
```

How to use an XSL variable (2)

- If your variable contains a sequence of elements, you can treat it just like any sequence.

```
<p>This book was written by the following people:</p>
<ul>
  <xsl:apply-templates select="$docAuthors"/>
</ul>
<p>But <xsl:value-of select="$docAuthors[1]/persName/surname"/> is listed
 first.</p>
```

```
<xsl:template match="author">
  <li>
    <xsl:text>author #</xsl:text>
    <xsl:value-of select="position()"/>
    <xsl:text> is </xsl:text>
    <xsl:apply-templates select="./persName/forename"/>
    <xsl:text> </xsl:text>
    <xsl:apply-templates select="./persName/surname"/>
  </li>
</xsl:template>
```

Why use XSL variables?

XSL variables can be very convenient in a lot of different circumstances:

- For instance, you might need to output the title of a document many times in different places. If you put it in an XSL variable, and the title changes, you only need to change it in one location in your XSLT.
- For instance, you may want to use the value of the current date hundreds of times. Instead of calling `current-date()` every time, you can call it once, and store the value in a variable.
- For instance:

```
 <xsl:variable name="maxGeos" select="max(//place/
count(descendant::geo))"/>
 <xsl:value-of select="//place[count(descendant::geo) = $maxGeos]/
@xml:id"/>
```

```
<xsl:for-each select="ref">
  <xsl:variable name="targetBiblId" select="substring-after(@corresp,
 '#')" />
  <xsl:value-of select="//div[@xml:id='bibliography']//bibl[@xml:id=
$targetBiblId]/title"/>
</xsl:for-each>
```