Problem: Putting dynamic content into the value of attributes

In the examples we have looked at so far, the XSLT stylesheets have grabbed content from attributes and elements, and then placed them as the content of elements. There haven’t been any attributes set so far.

Consider if you wanted to make a link to a book in Amazon.com’s website. You need to tack the ISBN number onto the URL of the book in the `href` attribute of the `a` element, such as:

http://www.amazon.com/exec/obidos/ASIN/0893470562

The straightforward approach won’t work:

```xml
<!-- won't work: -->
```

XML doesn’t allow the “<” character inside of an attribute value.

Two ways to handle this.

**xsl:attribute**

Adds attributes to elements in the output stream of an XSLT document.

```xml
<a>
  <xsl:attribute name="href">
    <xsl:value-of select="@isbn"/>
  </xsl:attribute>
</a>
```

Generates:

```xml
<a href="http://www.amazon.com/exec/obidos/ASIN/0893470562">
  … or whatever was in the value of the isbn attribute of the source.
</a>
```

The second way is to use:

**Attribute Value Templates**

A simpler syntax for setting attribute values:
The “{“ and “}” instruct the XSLT processor to dynamically fill-in the contents (essentially similar to a value-of instruction – according to the XSLT specification, the actual behavior is:

The attribute value template is instantiated by replacing the expression together with surrounding curly braces by the result of evaluating the expression and converting the resulting object to a string as if by a call to the `string` function.

**xsl:attribute-set**

Allows the definition of a set of attributes that can be reused.

```xml
<xsl:attribute-set name="body-style">
  <xsl:attribute name="bgcolor">white</xsl:attribute>
  <xsl:attribute name="text">green</xsl:attribute>
</xsl:attribute-set>
```

This is a “top level element” indicating it is a direct child of `xsl:stylesheet`.

**xsl:element**

Output elements are often directly represented in the input, and then passed through to the output, with the value being filled-in with “xsl:value-of” directives. `xsl:element` allows elements with computed names to be put into the output.

```xml
<xsl:element name="body" use-attribute-set="body-style">
  <h1>Title text</h1>
  <p>some paragraph text…</p>
</xsl:element>
```

**Outputs:**

```xml
<body bgcolor="white" text="green">
  <h1>Title text</h1>
  <p>some paragraph text…</p>
</body>
```

**Conditional Evaluation**

**xsl:if**
Consider if you wanted to output a name that is concatenated from the contents of first, middle, and last elements. The algorithm would be something like:

```xml
output value-of(first) {space}
output value-of(middle) {space}
output value-of(last)
```

If the person had no middle name, this would result in two spaces in the output. Want to check to see if the middle name element exists first.

Can do this with:

```xml
<xsl:if test="middle">
  ... conditional output
</xsl:if>
```

Can also check for existence of an attribute:

```xml
<xsl:if test="@arbitrary-attribute">
  ... conditional output
</xsl:if>
```

The value inside the test clause can be an arbitrary XPath expression. There are a number of XPath functions (not covered in class) that can be useful here.

No else clause for `xsl:if`.

**xsl:choose**

Similar to the Java switch statement. General format:

```xml
<xsl:choose>
  <xsl:when test="...">
    ... output elements
  </xsl:when>
  <xsl:when test="...">
    ... output elements
  </xsl:when>
  <xsl:otherwise>
    ... if none of the other conditions were hit
  </xsl:otherwise>
</xsl:choose>
```

**Variables**

Variables can be set as top-level elements, in which case they have global scope.

Otherwise, can set variables inside a template. Such variables are visible to elements that follow the declaration, and to their descendants.

Ways of setting variables:

```xml
<xsl:variable name="foo">value of variable foo</xsl:variable>
<xsl:variable name="bar" select="/arbitrary/XPath/expression"/>
```
Using variables:

```xml
<xsl:value-of select="$foo"/>
```

Can also use within Attribute Value Templates (AVTs):

```xml
<a href="{$link_location}">this is a link</a>
```

The braces inform the XSLT processor to use the value of the variable – otherwise, will include “$link_location” as a string value.

**Importing other stylesheets**

**xsl:include**

Destination stylesheet is inserted directly into the importing stylesheet (as if with a cut and paste).

```xml
<xsl:include href="…uri reference…"/>
```

Conflicts can occur.

**xsl:import**

Similar to xsl:include, but with conflict resolution rules.

```xml
<xsl:import href="…uri reference…"/>
```

In the event of conflicts, the importing stylesheet takes precedence.