Built-in template rules and how to override them

David J. Birnbaum

What's built in?

- Where do we start?
  - # at the document node (itself the parent of the root *element*)
- What happens when there is no applicable template rule?
  - # The built-in defaults are applied. (Kay, 79)
- What happens where there is exactly one applicable template rule?
  - # Er ... it gets applied.
- What happens when more than one template rule would seem to apply?
  - # Built-in priority (Kay, 686)
  - # User-specified priority (Kay, 483)

Launching a transformation

- Processing starts at the document node, which sits above the root element (the top-level element that contains all other nodes).
- In a TEI document, the root element is normally `<TEI>`.
- Built-in template rules will walk the tree and process the entire document.
- Override the built-in rules by specifying your own to:
  - # Perform non-built-in processing
  - # Cause items not to be processed

Built-in template rules

- Element: **process children** (elements and text nodes), applying matching template rules (built-in or specified)
- Attribute: **do nothing** (no output)
- Text: **output string value**

(Kay 79)

Priority

Built-in priority: the rule with the most specific match is applied

```
<xsl:template match="div">
  <act>
    <xsl:apply-templates/>
  </act>
</xsl:template>

<xsl:template match="div/div">
  <scene>
    <xsl:apply-templates/>
  </scene>
</xsl:template>
```

User-specified priority

```
<xsl:template match="div" priority="10">
  <act>
    <xsl:apply-templates/>
  </act>
</xsl:template>
```

- Built-in priorities range from -0.5 to 0.5
- User-specified priorities may be any number (including negative)

Overriding a template rule

-

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

- The built-in rule for any element is to process its child elements and text nodes.

- The built-in rule for the document node would process its single root element. In the case of a TEI document, that's usually the single `<TEI>` node.
-

```
<xsl:template match="/">
  <xsl:apply-templates select="//head"/>
</xsl:template>
```

- In the example above, instead you grab all `<head>` elements everywhere. No other nodes are processed unless they're inside a `<head>`.

A demonstration of template priority

- Open the *Hamlet* XML: http://web.uvic.ca/~mholmes/dhoxss2013/examples/hamlet.xml
- Open the Template Priority XSLT file: http://web.uvic.ca/~mholmes/dhoxss2013/examples/template_priority.xsl
- Switch to the XSLT debugger in oXygen.
- Now we'll experiment with template priority.

- Demonstrate how the initial run outputs each head as an h2 element.
- Add a second template which matches div/div/head, and outputs h3.
- Explain how this works: this template is **more specific**, so it is chosen.
- Now add priority="1" to the first template, and see how this overrides the specificity.
- Explain that the built-in priority values run from -0.5 to 0.5, so if you use anything above or below those values, you'll override them.
- Now demonstrate what happens when you remove the select="//head". Use this to review the built-in processing model again.