Template Modes

Martin Holmes

Template Modes

- You know how to write templates to process nodes. Typically, you have one template to process `<head>` nodes, another to process `<p>` nodes, and so on.
- However, sometimes you want to process the same node in different ways.
- One effective way to do this is using the `@mode` attribute.

Why process the same node in two different ways?

```
<body>
  <div type="chapter" n="1">
    <head>Chapter 1: How it all started</head>
    <p>[...]</p>[...]<p>[...]</p>
  </div>
  <div type="chapter" n="2">
    <head>Chapter 2: What happened next</head>
    <p>[...]</p>[...]<p>[...]</p>
  </div>
  <div type="chapter" n="3">
    <head>Chapter 3: Things that subsequently transpired</head>
    <p>[...]</p>[...]<p>[...]</p>
  </div>
  [...]
</body>
```

- Each `<head>` element needs to be processed in a particular way when it shows up in its chapter.
- However, you also want to create a table of contents, based on a list of the `<head>` elements.
- The `<head>`s will need to be processed differently for the TOC.

Two templates for the same node

Using modes, we can write two different templates for the same node:

```
<xsl:template match="head">
  <h2 id="chapter_{parent::div/@n}"><xsl:apply-templates /></h2>
</xsl:template>
```

```
<xsl:template match="head" mode="toc">
  <li>
    <a href="#chapter_{parent::div/@n}"><xsl:apply-templates /></a>
  </li>
</xsl:template>
```

- The first template, which processes the `<head>` at the beginning of the chapter, creates an `<h2>` element with a unique id.
- The second template, which processes the `<head>` element in the context of a table of contents, creates a list item (`<li>`) element containing a link which points to the chapter heading.

Creating the table of contents

```
<div id="tableOfContents">
  <ul class="tocList">
            <xsl:apply-templates select="//div[@type='chapter']/head"
 mode="toc"/>
  </ul>
</div>
```

Results

This is what the output might look like:

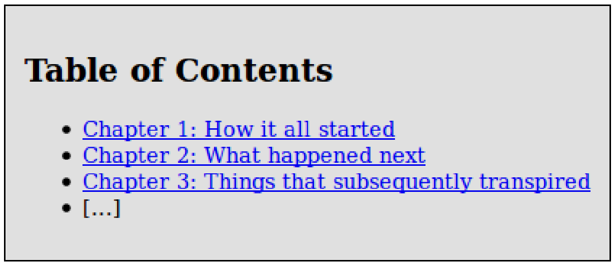
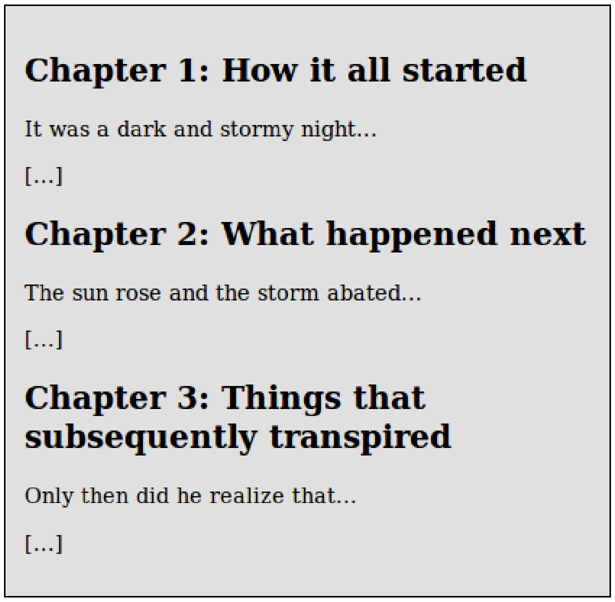


Table of Contents

- Chapter 1: How it all started
- Chapter 2: What happened next
- Chapter 3: Things that subsequently transpired
- [...]



Chapter 1: How it all started

It was a dark and stormy night...

[...]

Chapter 2: What happened next

The sun rose and the storm abated...

[...]

Chapter 3: Things that subsequently transpired

Only then did he realize that...

[...]

Modes Task 1: Creating a TOC

- Download the example short David Copperfield XML file: http://web.uvic.ca/~mholmes/dhoxss2013/examples/copperfield.xml
- Download the corresponding simple XSL file: http://web.uvic.ca/~mholmes/dhoxss2013/examples/copperfield.xsl
- Set up a transform in the XSLT debugger, or through a Transformation Scenario, whichever suits you best. We'll work together through the process of creating a table of contents.

Modes Task 2: Handling footnotes

In the same David Copperfield file, you'll see that at the end of each chapter, there's a `<note>` element.

Your task is to turn the notes into footnotes, and put numbers in the text which link to the footnotes.

- First create a normal template for `<note>` which turns the note into a number.
- Turn the number into a link.

- Create another template for note, with a different mode, which outputs the contents of the note inside a `<li>` element.
- Add a `<ul>` element to the root template before the end of the body.
- Inside the `<ul>` element, apply templates to `//note` with the new mode.

More on modes

- One template can serve multiple modes:

```
<xsl:template match="head" mode="toc index abstract">
```

- There are two special mode values you can apply to an `<xsl:template>`:
  # **#all** (applies to all modes)
  # **#default** (applies when no `@mode` attribute is specified)
- There is one special mode value you can apply to an `<xsl:apply-templates>`:
  # **#current** (apply templates using whatever mode is current in the processing cascade)
- These only really become useful when you are using lots of modes throughout a transformation.