Comp 336/436 - Markup Languages

Fall Semester 2017 - Week 6

Dr Nick Hayward

DEV Week Assessment

- demo and project report
  - *due on Wednesday 18th October 2017 @ 4.15pm*
- anonymous peer review
  - *similar to user comments and feedback*
  - *chance to respond to feedback before final project*

DEV Week assessment

- project outline and introduction
- developed using a chosen markup language
- current working examples - what does and does not work...
- outline research conducted
- describe data chosen for project, e.g.
  - *data used for DEV Week*
  - *data planned for final project*
- show any mockups, prototypes, patterns, and designs

DEV week presentation and demo...

brief presentation or demonstration of current project work

- 30% of final grade
- ~ 10 minutes per group
- analysis of work conducted so far
  - *eg: during semester & DEV week*
- presentation or demonstration
  - *outline current state of project*
  - *show mockups, designs, test &c.*
  - *explain what works & does not work*
  - *anything else considered relevant to your research or development...*

XML - XSLT working example - Agatha Christie

XSLT - `<xsl:if>`

- conditional test within the template
  - *for certain conditions and requirements in the XML*

```
<xsl:if test="year &gt; 1929">
  ...
</xsl:if>
```

- value of `test` attribute contains expression to be tested
  - *year elements with values greater than 1929*
  - *i.e. year > 1929*

XML - XSLT working example - Agatha Christie

XSLT - part 7

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<xsl:stylesheet version="1.0"
xmlns:xsl="http://www.w3.org/1999/XSL/Transform">

<xsl:template match="/">
  <html>
  <body>
  <h2>Collection</h2>
  <table>
    <tr>
      <th>Title</th>
      <th>Author</th>
      <th>Year</th>
    </tr>
    <xsl:for-each select="catalogue/book">
    <xsl:sort select="title"/>
    <xsl:if test="year &gt; 1937">
    <tr>
      <td><xsl:value-of select="title"/></td>
      <td><xsl:value-of select="author"/></td>
      <td><xsl:value-of select="year"/></td>
    </tr>
    </xsl:if>
    </xsl:for-each>
  </table>
  </body>
  </html>
</xsl:template>

</xsl:stylesheet>
```

- Agatha Christie - XSLT - part 7

XML - XSLT tests - initial XML

Exercise - part 6

- add a conditional option to `<xsl:for-each>`
- modify the output to render this conditional result
  - *add a note to rendered output for chosen conditional...*
- test XSL with XML file

10 minutes...

XML - XSLT working example - Agatha Christie

XSLT - `<xsl:choose>`

```
<xsl:choose>
  <xsl:when test="expression">
    ... some output ...
  </xsl:when>
  <xsl:otherwise>
    ... some output ....
  </xsl:otherwise>
</xsl:choose>
```

- use `<xsl:choose>` with `<xsl:when>` or `<xsl:otherwise>`
  - *test multiple conditions in XML*

XML - XSLT working example - Agatha Christie

XSLT - `<xsl:choose>`

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<xsl:stylesheet version="1.0"
xmlns:xsl="http://www.w3.org/1999/XSL/Transform">

<xsl:template match="/">
  <html>
  <body>
  <h2>Collection</h2>
  <table>
    <tr>
      <th>Title</th>
      <th>Author</th>
    </tr>
    <xsl:for-each select="catalogue/book">
    <tr>
      <td><xsl:value-of select="title"/></td>
      <xsl:choose>
      <xsl:when test="year &gt; 1941">
          <td class="post">After: <xsl:value-of select="author"/></td>
      </xsl:when>
      <xsl:otherwise>
          <td class="pre">Before: <xsl:value-of select="author"/></td>
      </xsl:otherwise>
      </xsl:choose>
    </tr>
    </xsl:for-each>
  </table>
  </body>
  </html>
</xsl:template>

</xsl:stylesheet>
```

- Agatha Christie - XSLT - part 8

XML - XSLT tests - initial XML

Exercise - part 7

- modify existing conditional option
  - *use `<xsl:choose>` with `<xsl:when>` and `<xsl:otherwise>`*
- modify the output to render this conditional result
  - *add a note to rendered output for chosen conditional...*
- test XSL with XML file

10 minutes...

XML - XSLT working example - Agatha Christie

XSLT - `<xsl:apply-templates>`

- add a template to a current element or its child nodes
- use a `select` attribute
  - *only process child element specified in the value*
- use a `select` attribute
  - *specify order of child node processing*

XML - XSLT working example - Agatha Christie

XSLT - template and match

- then use standard `<xsl:template>`
  - *add `match` attribute to specify required element*
  - *add further XSL options to modify elements &c.*

e.g.

```
<xsl:template match="title">
 <xsl:choose>
        <xsl:when test="../price&lt;10">
          <span style="color:#ff00ff">
          <xsl:value-of select="."/>
          </span>
        </xsl:when>
        <xsl:otherwise>
          <span>Price too high: <xsl:value-of select="."/></span>
        </xsl:otherwise>
      </xsl:choose>
</xsl:template>
```

- Agatha Christie - XSLT - part 9

XML - XSLT working example - Agatha Christie

XSLT - <xsl:apply-templates>

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<xsl:stylesheet version="1.0"
xmlns:xsl="http://www.w3.org/1999/XSL/Transform">

<xsl:template match="/">
  <html>
  <body>
  <h2>Collection</h2>
  <xsl:apply-templates/>
  </body>
  </html>
</xsl:template>

<xsl:template match="book">
  <p>
  <xsl:apply-templates select="title"/>
  <xsl:apply-templates select="author"/>
  </p>
</xsl:template>

<xsl:template match="title">
  Title: <span class="title"><xsl:value-of select="."/></span>
  <br />
</xsl:template>

<xsl:template match="author">
  Author: <span class="author"><xsl:value-of select="."/></span>
  <br />
</xsl:template>

</xsl:stylesheet>
```

- Agatha Christie - XSLT - part 10

XML - XSLT tests - initial XML

Exercise - part 8

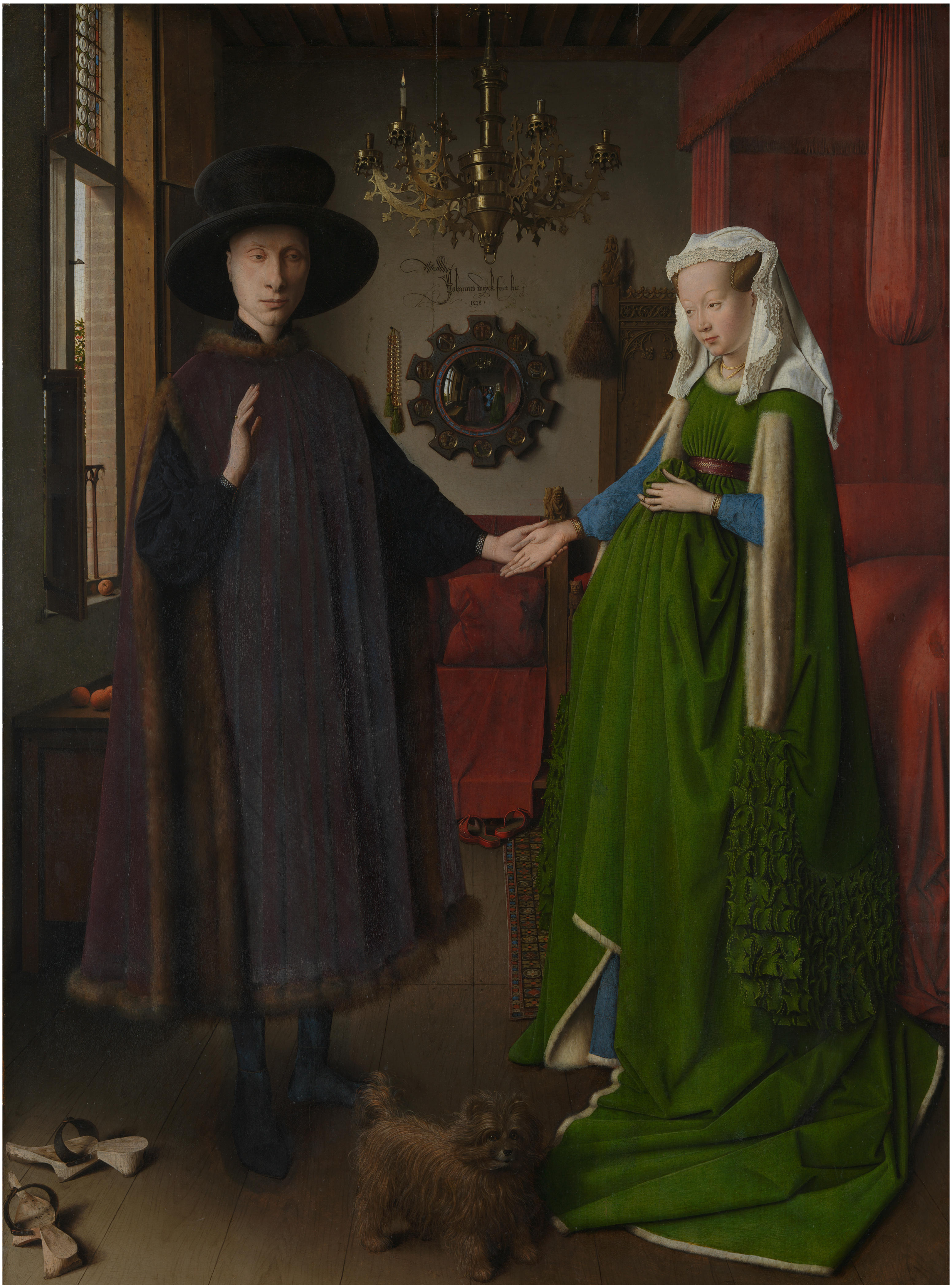
- modify existing XSL
- abstract XSL to use `<xsl:template>` and `<xsl:apply-templates>`
- test XSL with XML file

10 minutes...

XML - XSLT tests - initial XML

- DEMO

XML & XSLT - working with images - Arnolfini Portrait



Arnolfini Portrait

- YouTube - Arnolfini Portrait

XML & XSLT - image

Exercise

- create an XML file to markup this image
- consider the following
  - *what is it?*
  - *who created it?*
  - *where is it located now?*
  - *history?*
  - *brief description*
  - *other metadata*
  - *...*
- how does your XML help a user understand this image?
  - *e.g. if it was unknown in an archive, could a user associate the image with the metadata?*

~ 15 minutes

XML & XSLT - image test

- DEMO

XML - transforming & rendering - XSL-FO intro

- XSL-FO - XSL Formatting Objects (or Flow Objects)
- XML-based markup language
  - *describes formatting of XML data for output to screen, paper or other media...*
  - *e.g. output to PDF, XML, RTF, &c.*
- XSL-FO is used inside XSLT transformations
- XSLT transformation takes an XML document (source tree)
  - *directives to produce a result tree*
  - *work is done by the processor, which interprets the directives*
- once transformed
  - *formatting operation completed by the XSL processor*
  - *processor interprets the result tree*
  - *processor checks formatting objects contained in directives*
  - *checks directives specific to the XSL-FO language*
- XSL-FO language was designed for paged media
  - *concept of page is an important part of its structure*

XML - transforming & rendering - XLS-FO example

- XSL-FO documents are XML files with output information
- usually stored in files with .fo or .fob extension

e.g.

```
<?xml version="1.0"?>
<fo:root xmlns:fo="http://www.w3.org/1999/XSL/Format">
<fo:layout-master-set>
  <fo:simple-page-master master-name="A4">
  <!-- Page template goes here -->
  </fo:simple-page-master>
</fo:layout-master-set>
<fo:page-sequence master-reference="A4">
<!-- Page content goes here -->
</fo:page-sequence>
</fo:root>
```

XML - transforming & rendering - XSL-FO structure

- XSL-FO document starts with a root element `<fo:root>`
  - *contains the appropriate namespace declaration*
  - *commonly* `“http://www.w3.org/1999/XSL/Format”`
- two main elements are declared
  - `fo:layout-master-set` *- contains collection of definitions*
    - page geometries and page selection patterns
  - `fo:page-sequence` *- contains definition of information*
    - for a sequence of pages with common static information
- xsl-fo structure works as follows,
  - *formatter reads the XSL-FO document*
  - *creates a page based on first template in* `fo:layout-master-set`
  - *fills it with content from the* `fo:page-sequence`
  - *after first page*
    - instantiates a second page based on a template
    - fills it with content
  - *process continues until the formatter runs out of content...*

XML - transforming & rendering - XSL-FO page formatting

- page templates are called page masters
- each defines a general layout for a page including
  - *its margins*
  - *sizes of the header, footer...*
  - *sizes of the body area of the page*
- e.g.

```
<fo:layout-master-set>
<fo:simple-page-master master-name="…"
page-height=".." page-width=".." […]>
<fo:region-body/>
</fo:simple-page-master>
</fo:layout-master-set>
```

- a `fo:layout-master-set` containing one `fo:simple-page-master`
- contains a single region, the body
  - *all content will be placed in this region*

XML - transforming & rendering - XSL-FO page sequence management

- in addition to a `fo:layout-master-set`
- each FO document contains one or more `fo:page-sequence elements`
- XSL-FO specifies the sequence of pages
  - *each page has an associated page master*
  - *page master defines how the page will look*
- each page sequence contains three child elements:
  - *optional `fo:title` element - inline content for title of document*
  - *zero or more `fo:static-content` elements - content for every page*
  - *one `fo:flow` element - data placed on each page, e.g. pagination*

XML - transforming & rendering - XSL-FO page sequence management - example

```
<fo:page-sequence master-reference=“chaps”>
<fo:static-content flow-name="…">
<fo:block text-align="outside" …>
Chapter
<fo:retrieve-marker
retrieve-class-name="chapNum"/>
<fo:leader leader-pattern="space" />
<fo:retrieve-marker retrieve-class-name="chap"/>
<fo:leader leader-pattern="space" />
Page
<fo:page-number font-style="normal" />
of
<fo:page-number-citation ref-id='end'/>
</fo:block>
</fo:static-content>
<fo:flow flow-name="…">
<fo:block>
<!-- Output goes here -->
</fo:block>
</fo:flow>
</fo:page-sequence>
```

- sequence of pages is defined for chapters in a book
- document gives directives for rendering
  - *chapter numbers, page number, and other information…*

XML - transforming & rendering - XSL-FO formatting objects

```
<fo:block font-family="Times" font-size="14pt">
This text will be Times of size 14pt!
</fo:block>
```

- similar in concept to CSS
  - *possible to enrich text with character-level formatting*
  - *several properties control font styles — `family`, `size`, `color`, `weight`, &c.*

XML - transforming & rendering - XSL-FO formatting objects

```
<fo:block line-height="1.0" text-align="justify">
Example of a justified formatted block.
The space between lines is 1.0.
</fo:block>
```

- other formatting objects are specialised
  - *e.g. describe output on different media - pagination, borders...*
- list-item formatting object
  - *a box containing two other formatting objects*
  - *`list-item-label` and `list-item-body`*
- tables, lists, side floats, and a variety of other features are available...
- many features comparable to CSS...

Demos

XML & XSLT - Part 2

- Agatha Christie - XSLT - part 7
- Agatha Christie - XSLT - part 8
- Agatha Christie - XSLT - part 9
- Agatha Christie - XSLT - part 10

References

- Oxygen XSLT Processors
- W3C - GRDDL
- W3C - OWL
- W3C - RDF
- W3C - SPARQL
- W3C - XML well formed
- Xalan Project