

# **Comp 336/436 - Markup Languages**

---

Fall Semester 2018 - Week 8

Dr Nick Hayward

## 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...

## **XML - XPath details - expressions & patterns**

---

- in XSL - patterns and expressions written using XPath (XML Path language syntax)
- XPath used to select nodes and node sets
  - *specify location paths from XML document*
- XPath includes built-in functions
  - *e.g. mathematics, process strings, conditionals &c.*
  - *many more...*

## XML - XPath details - location - nodes

---

- nodes are the core of working with XPath
  - *use location paths to reference a node or node set*
- location path uses relationships to describe location of node or node set
  - *relative to a given node*
- XPath considers all XML documents as tree structures
- node trees
  - *hierarchical structure of nodes*
  - *XPath uses this hierarchy to navigate*



# XML - working example - ancient sites

---

```
<?xml version="1.0" encoding="UTF-8"?>
<ancient_sites>
  <site>
    <name language="english"></name>
    <location></location>
    <country></country>
    <description></description>
    <culture></culture>
    <history></history>
    <images>
      ...
    </images>
    <notes></notes>
  </site>
</ancient_sites>
```

## XML - XPath details - location - node tree - recap

---

- in the XML DOM, everything in an xml doc is a node, e.g.
  - *entire doc is a document node*
  - *text in xml elements are text nodes*
  - *each attribute is an attribute node*
- root node, or document node, can have any number of child nodes
- to child nodes, the root node is a parent node
- child nodes can have any number of child nodes themselves
- child nodes with the same parent are called sibling nodes
- descendant nodes are a node's child nodes, grandchild nodes &c.
- ancestor nodes are a node's parent node, grandparent nodes
- navigate and find nodes by knowing such node relationships

# XML - working example - ancient sites - node tree

---

```
| - ancient_sites
  | -- site
    | -- name [@language]
    | -- location
    | -- country
    | -- description
    | -- culture
    | -- history
    | -- images
    |   | -- ...
  | -- notes
```

# **XML - XPath & XSLT tests - initial XML**

---

## ***Exercise - part I***

- choose a favourite historical or tourist site
  - e.g. *Eiffel Tower in Paris*
- markup in XML relevant and useful details of this site
- add any required metadata, notes, comments, &c.
- add a reference to a representative image of this site

~ 10 minutes

## XML - XPath details - location - paths

---

- two kinds of location paths that we predominantly use with XPath
  - *relative and absolute location paths*
- *relative*
  - *consists of a sequence of location steps separated by /*
  - *each step selects a node or node set relative to the current node*
  - *e.g. parent/child/grandchild*
- *absolute*
  - */ optionally followed by a relative location path*
  - */ by itself selects the root node e.g. /root/child/grandchild*

# XML - XPath details - location - using paths

---

- two predominant uses
  - *wrapper to find a node's location and then process child nodes*
  - *get a node's value*
- in XPath, there are seven different node types
  - *root nodes (always one)*
  - *element nodes*
  - *text nodes*
  - *attribute nodes*
  - *comment nodes*
  - *processing instruction nodes*
  - *namespace nodes*
- a way to retrieve the value of each node type
- for some nodes
  - *value is part of node*
- for other nodes
  - *value is based on value of descendant nodes*

## **XML - XPath details - location - current node**

---

- as XSLT processor goes through your style sheet
  - *it works on one node at a time*
- XSLT processor knows the parts of a document to process using
  - *xsl:template, xsl:apply-templates, and xsl:for-each* elements
- node currently being processed is called the *current node*
  - *current node varies as processor traverses node tree*
- *context* node is the start position for an XPath location path address
- refer to current node with shortcut
  - *in a location path simply use . (a single period)*



# XML - working example - ancient sites - current node

---

## XML

```
<?xml version="1.0" encoding="UTF-8"?>
<?xml-stylesheet type="text/xsl" href="ancient-sites.xsl"?>
<ancient_sites>
  <site>
    <name language="english">Great Pyramid</name>
    <location>Giza</location>
    <country>Egypt</country>
    <description>Khufu's pyramid on the Giza plateau outside Cairo</description>
    <culture>Ancient Egypt</culture>
    <history>
      <period>Old Kingdom</period>
      <dynasty>4th</dynasty>
    </history>
    <images>
      ...
    </images>
    <notes>...</notes>
  </site>
</ancient_sites>
```

# XML - working example - ancient sites - current node

---

## XSL

```
<!-- first node match -->
<xsl:template match="/">
  <html>
    <body>
      <h3>Ancient Sites</h3>
      <!-- second match -->
      <xsl:apply-templates select="ancient_sites/site">
        <xsl:sort select="location" order="ascending" data-type="string" />
      </xsl:apply-templates>
    </body>
  </html>
</xsl:template>

<!-- third match -->
<xsl:template match="site">
  <xsl:apply-templates select="name[@lang='en']" />
</xsl:template>

<!-- fourth match -->
<xsl:template match="name[@lang='en']">
  <xsl:value-of select="." />
</xsl:template>
```

# XML - XPath & XSLT tests - current node

---

## Exercise - part 2

- create an XSL stylesheet for your XML document
  - *test the stylesheet work with the XML*
- add matches for root, first parent, &c.
- add sort order for your sites
- check current node throughout stylesheet
  - *i.e. reference current node in XSL*

~ 10 minutes

## References

---

- [XML.com - What is XSL-FO](#)
- [XMLNS - FOAF spec](#)
- [XPath Version 1.0](#)
- [W3Schools - XPath](#)