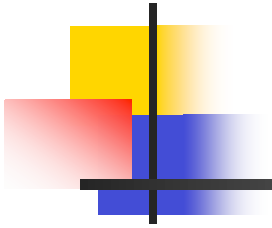
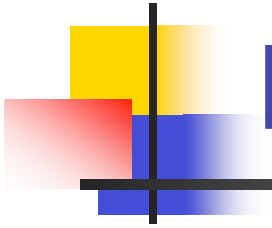


B336 Internet Systems Programming

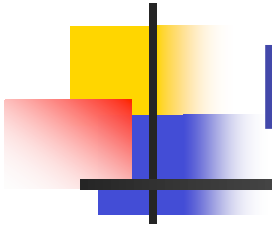
XSL:
The eXtensible Style Language

(Week 6 Lecture 2)



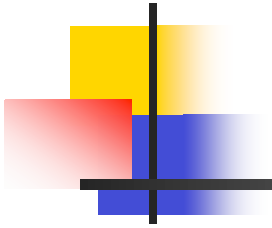
Learning Objectives

- Understand the role of XSL and XSLT in XML technologies.
- Know the basic syntax of XSLT to transform XML documents.



Learning Objectives

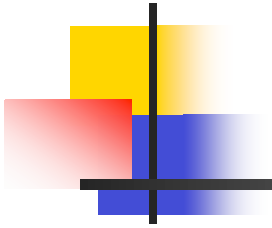
- In the scheme of what we are doing in this unit:
  - We are studying how to use XML as an important set of Internet technologies to use as solutions in different areas.
  - In most cases, information in XML documents will need to be processed in other formats. XSL is one of the standard ways of doing this.



Lecture Outline

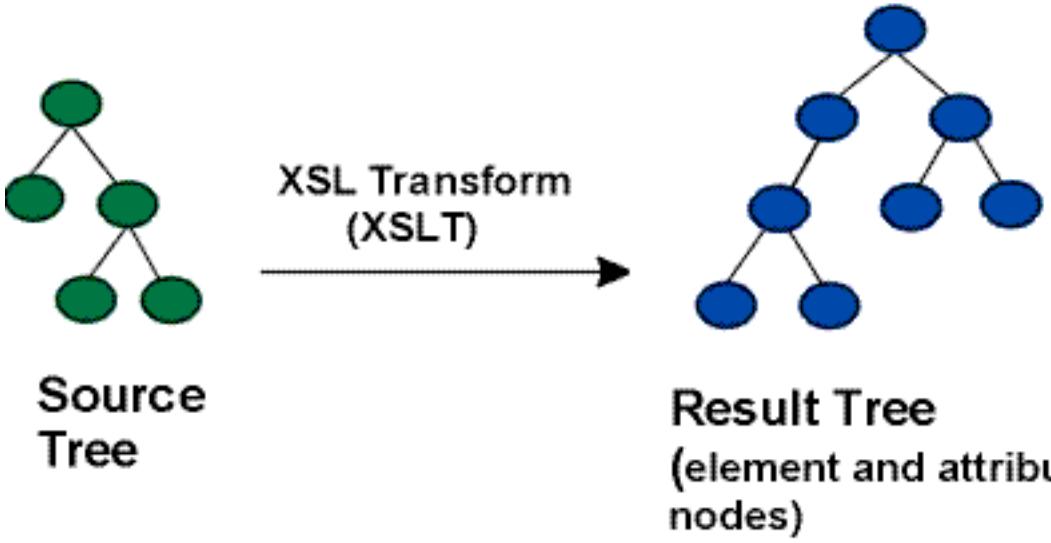
- What is XSL and XSLT?
- XPath expressions
- XSLT style sheet syntax.

The eXtensible Style-sheet Language (XSL)



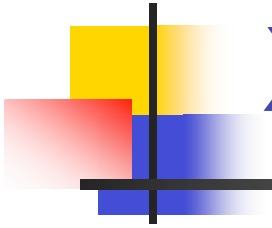
- XSL is an XML-based language used to create "style sheets".
- XML software can use an XSL style sheet to transform an XML document into another document
  - Eg. into another XML format, HTML, or any other text-based format (even RTF and PDF, although these are quite complex).
- The style sheet defines how to get from the input document (**source tree**) to the output document (**result tree**).

With tree transformation, the structure of the result tree can be quite different from the structure of the source tree



In constructing the result tree, the source tree can be filtered and reordered, and arbitrary structure and generated content can be added.

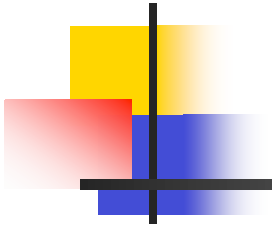
Source:
http://www.w3.org/TR/xsl/slice1.html#section-N629-Introduction-and-Overview



XSLT and XSL:FO

- There are two complete languages under XSL:
  - XSL for Transformation (XSLT) - to transform the XML document to another document format.
  - XSL Formatting Objects (XSL:FO) - to format the result tree for display (on different devices). Eg. define font sizes.
- The official recommendations for both can be found at:

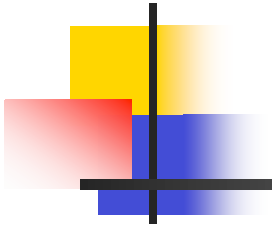
  http://www.w3.org/TR/xsl



Why is XSL important?

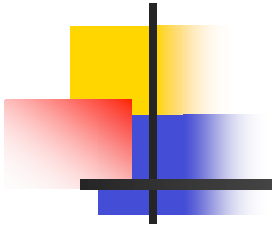
- Extracting appropriate information.
  - Software and users would probably only require a small part of the information in one XML document.
  - Software and users would probably require information from more than one XML document.

- Use across different applications
  - Must be able to get the information in an XML document into and out of legacy systems which do not deal with XML.
  - Even systems which deals in XML may not use the same XML language.



Pattern Matching

- The XSLT language basically defines a set of **templates**.
- The templates specifies
  - what to look for in the source tree, and
  - what to put in the result tree.
- The process of a style-sheet involves matching templates against the contents XML documents
  - If match, then do what the templates says.



Pattern Matching

- Many students in the past have encountered many problems writing XSLT because they failed to think in terms of this template concept.
  - The natural tendencies is for students to think of XSLT statements as "instructions".
  - Unfortunately if you do this, your style-sheets will not work as you expect them to.
  - Please take time to understand the this lecture, and the examples in the textbook to grasp this concept of defining "patterns" rather than defining "instructions".

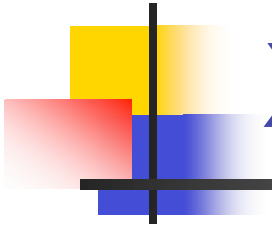
Associating XML document to XSLT style sheets

- An XML document can be associated with a given XSLT style sheet
  - By putting a `<?xml-stylesheet…?>` tag in the *Processing Instructions* parts of the XML document.
  - Some of you would have done something similar like this before with CSS in HTML.

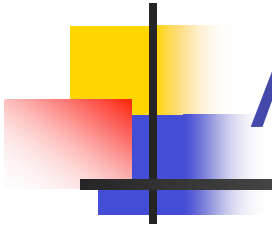
- Eg.

```
<?xml version="1.0"?>
<?xml-stylesheet type="text/xsl" href="myxsl.xsl"?>

...
```

Associating XML document to XSLT style sheets



- This gives the software processing the XML document (eg. browsers like IE5) the option of using that style sheet to transform the document if it deems appropriate.
  - The software will first have to know how to deal with “text/xsl” documents, though.



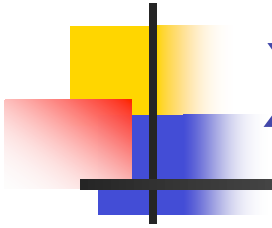
An XSLT Style Sheet

- Since and XSLT style sheet is also an XML document, it conforms to all the well-formed rules we discussed before.
  - It usually has the `<?xml …?>` declaration.
  - Its root element of the style sheet is `<xsl:stylesheet>`.
  - `<xsl:stylesheet>` contains the zero or more of the element `<xsl:template>` which defines the templates for the tranformations.

An XSLT Style Sheet

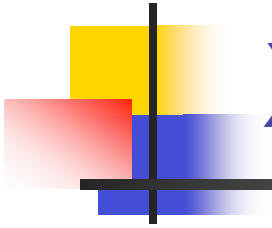
- An example:

```
<?xml version="1.0"?>
<xsl:stylesheet version="1.0" xmlns:xsl="http://www.w3.org/1999/XSL/Transform">
  <xsl:template match="/">
    <HTML>
      <BODY>
        <xsl:for-each select="/poetry/anthology/poem/stanza/line">
          <P>
            <xsl:value-of select="."/>
          </P>
        </xsl:foreach>
      </BODY>
    </HTML>
  </xsl:template>
</xsl:stylesheet>
```



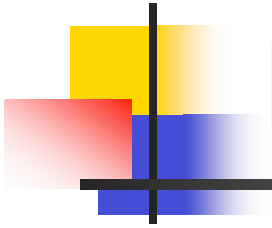
Matching the Source Tree using XPath Expressions

- As we mentioned, a major part of XSLT's job is to get data from a source tree (the original XML document).
- For it to do this, there must be some convenient way of locating components of the source tree. We do this using XPath expressions.



XPath

- XPath is an official W3C recommendation.
  - http://www.w3.org/TR/xpath



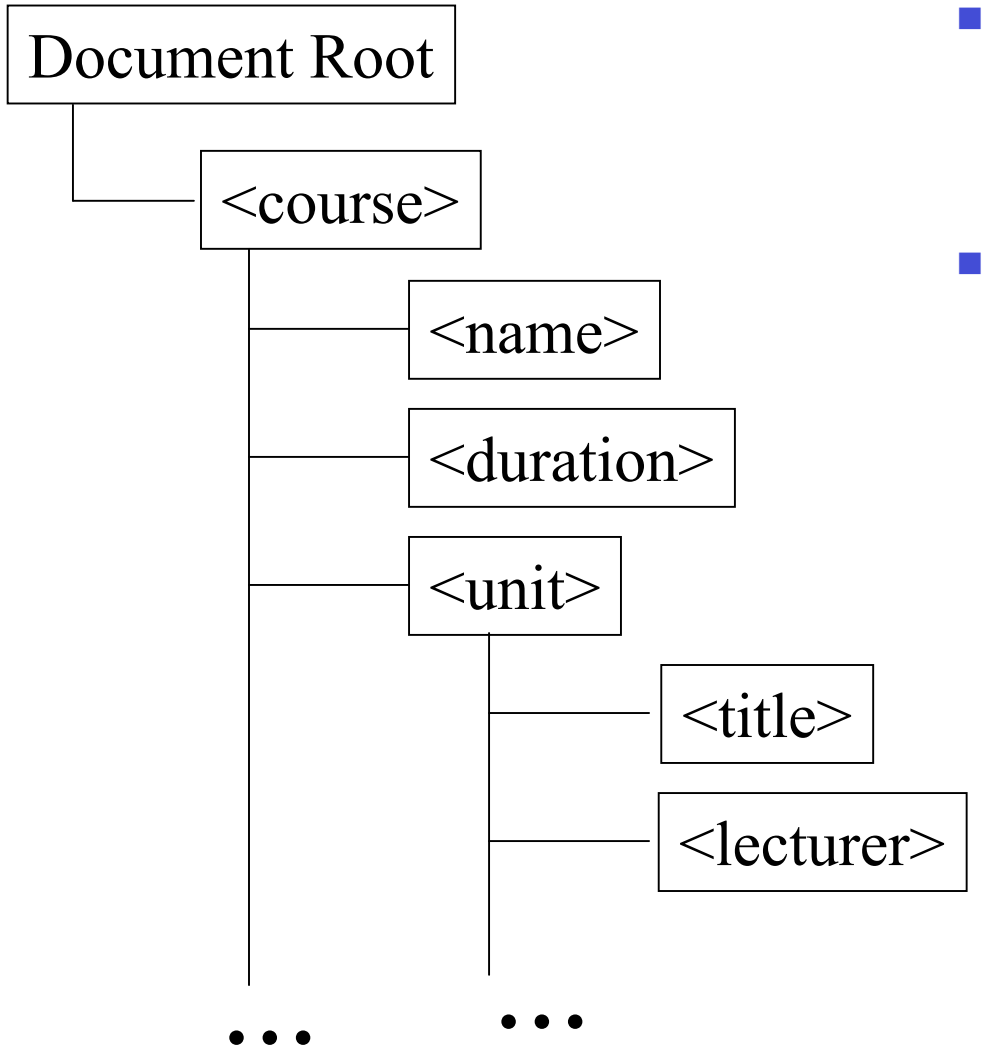
Nodes in XPath

- XPath to refer to anything in the source tree as a *node*
  - Elements
  - Attributes
  - Processing Instructions
  - etc

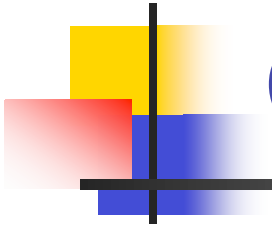
Example XML Document

```
<?xml version="1.0"?>

<course>
     <name>Bachelor of Science - Internet Computing</name>
     <duration>3 years</duration>
     <unit>
          <title>B336 Internet Systems Programming</title>
          <lecturer>
               <surname language="English">Hiew</surname>
               <othernames language="English">Hong Liang</othernames>
               <email>h.hiew@murdoch.edu.au</email>
          </lecturer>
     </unit>
     <unit>
          <title>B108 Introduction to Multimedia and the Internet</title>
          <lecturer>
               <surname>Rai</surname>
               <othernames>Shri</othernames>
               <email>s.rai@murdoch.edu.au</email>
          </lecturer>
     </unit>
</course>
```

Nodes in our example document



- We refer to the document root as “/” (note: document root is **not** the root element)
- We refer to the rest of the tree using the same notation as a directory paths in a UNIX file system. Eg
  - “/course”
  - “/course/unit/lecturer”
  - These are called **location paths** in XSLT

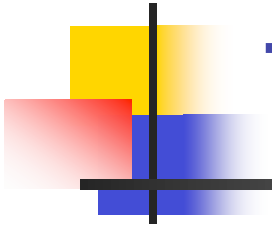


Context Node

- At any single time, XPath will consider one of the nodes to be the ***context node*** (ie. “the node the processing is currently at”).

- XPath expressions will be evaluated based on the current context node.

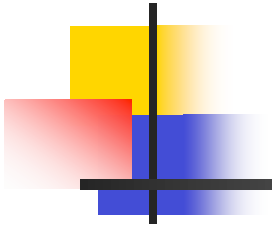
  - Let’s look more closely at an XSLT style sheet to see what this means.



The XSLT style sheet again

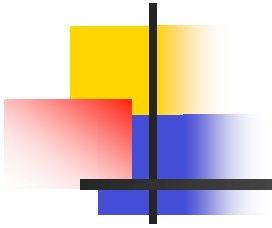
- So the basic structure of the XSLT style sheet is normally:

```
<?xml version="1.0"?>
<xsl:stylesheet ...>

  <xsl:template match=XPath expression >
    ... Something to do ...
  </xsl:template>

  <xsl:template match=XPath expression >
    ... Something to do ...
  </xsl:template>

  ...

</xsl:stylesheet>
```



Evaluating the Templates

- What XSLT will do is set the current context node to the document root, and start finding the template that matches the current node.
  - It will then produce the result tree according to what is specified in the body of the template.
- If more than one template matches the current node, the last one (counting from top to bottom) will be evaluated.



Evaluating the Templates

- If you want the templates to be applied to the nodes below the document root, you must specify the element

  `<xsl:apply-templates …>`

  within the body of the template which matches the document root.

Evaluating the Templates

- Eg. The following will not work:

```
<xsl:template match="/">
  <!-- Do nothing -->
</xsl:template>

<xsl:template match="line">
  The line is: <xsl:value-of select="."/>
</xsl:template>
```

This template is never matched!

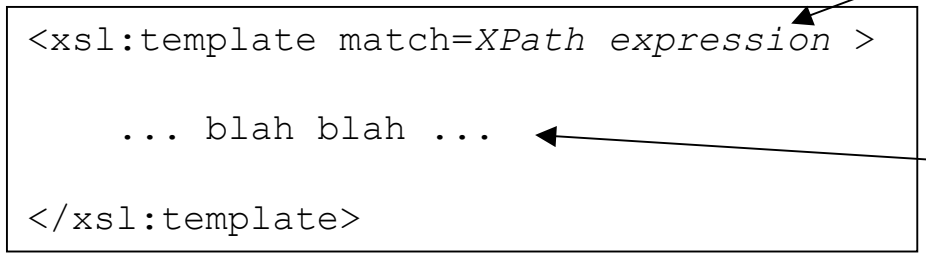
- But this will:

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

<xsl:template match="line">
  The line is: <xsl:value-of select="."/>
</xsl:template>
```

Match the document root "/" or any other node "*" - so this recursively applies all templates to EVERY node.

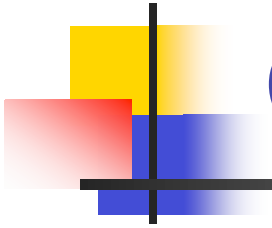
Evaluating the Templates

- In summary: at any time while going through the XML source tree…



…if the current context node matches this XPath expression…

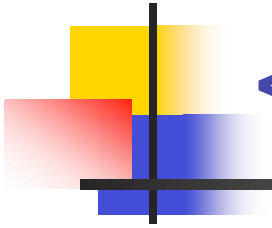
… then put the content specified here into the result tree. If the part of the content is a valid xsl tag (eg <xsl:apply-templates/>), then the tag is evaluated and results also put in.



Creating the Result Tree

- We have seen the basis of traversing through the source tree. Now lets look at getting interesting information out of the source tree to put in the result tree.

- The most commonly used element for doing that is `<xsl:value-of>`

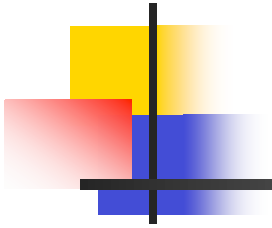


<xsl:value-of>

- The common format for the element is:

```
<xsl:value-of select="XPath Expression">
```

  - This puts into the result tree the value as specified by the XPath expression.
  - The expression can contain a lot more than just a location path.



Functions in XPath

- To get interesting information out of the source tree to use in elements like <xsl:value-of>, we use the set of functions available in XPath.
- Eg.

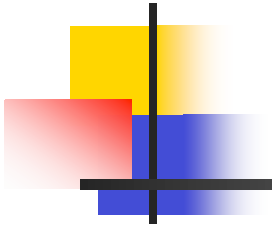
name()	The name of the node.
text()	The #PCDATA of the current node.
sum()	The sum of the numbers given in the #PCDATA of specified nodes.
concat()	Concatenate two strings.

Functions in XPath

- Egs.

```
<xsl:template match="*|/">
  The node is <xsl:value-of select="name()"/>
  <xsl:apply-templates/>
</xsl:template>
```

```
<xsl:template match="/">
  The sum of all numbers in this document is
  <xsl:value-of select="sum(/spreadsheet/numbers)"/>
</xsl:template>
```



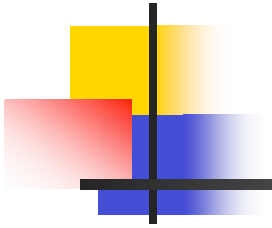
Functions in XPath

- XPath functions are also useful for specifying location paths
- Eg. Finding different lines

```
<xsl:template match="/course/unit(position()=1)">
```

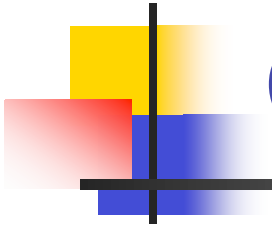
```
<xsl:template match="/course/unit[1]">
```

```
<xsl:template match="/course/unit(position()=last)”>
```



Conditional Processing

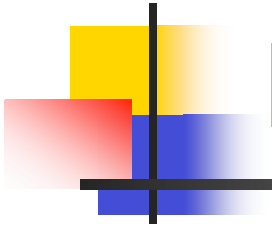
- When we are traversing the source tree, we can also do processing based on conditions. We do so using the elements such as `<xsl:if>` and `<xsl:choose>`.



Conditional Processing

- The basic syntax:

```
<xsl:if test="Boolean expression"> ... </xsl:if>


<xsl:choose>
  <xsl:when test="Boolean expression"> ... </xsl:when>
  <xsl:when test="Boolean expression"> ... </xsl:when>
  ...
</xsl:choose>
```

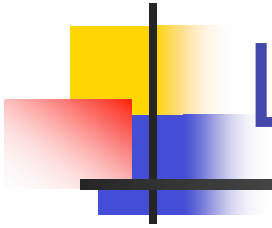


Loop

- You can also do loops (iterations) using the element `<xsl:for-each>`.

- The basic syntax:

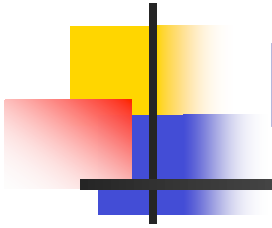
```
<xsl:for-each select="XPath expression">
```



Loop

- Eg.

```
<xsl:template match="unit">
    <xsl:for-each select="lecturer">
       Another lecturer
    </xsl:for-each>
</xsl:template>
```

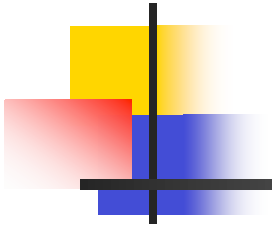


Built-in Templates

- By default, the following templates will already exist:

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

<xsl:template match="text()|@*">
  <xsl:value-of select="."/>
</xsl:template>
```

  - So even if you have no templates in your style sheet, the #PCDATA in every node your document will be put in the result tree. If a node doesn't have #PCDATA, a new-line character will be put.
  - If you don't define any templates to override the default behaviours, they will be invoked. This can sometimes cause confusion among students because their templates are producing results they never defined.



Extra Reading

- Required reading
  - Sybex textbook chapter 12.