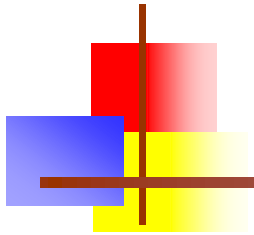
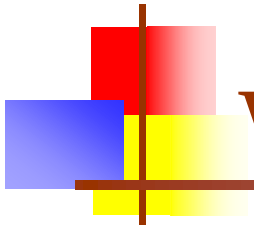


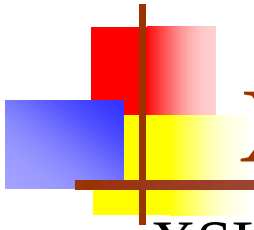
XSL

XSLT and XPath



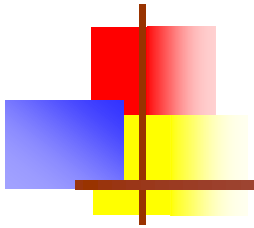
What is XSL?

- XSL stands for Extensible Stylesheet Language
- CSS was designed for styling HTML pages, and can be used to style XML pages
- XSL was designed specifically to style XML pages, and is much more sophisticated than CSS
- XSL consists of *three* languages:
  - XSLT (XSL Transformations) is a language used to transform XML documents into other kinds of documents (most commonly HTML, so they can be displayed)
  - XPath is a language to select parts of an XML document to transform with XSLT
  - XSL-FO (XSL Formatting Objects) is a replacement for CSS
    - There are no current implementations of XSL-FO, and we won't cover it



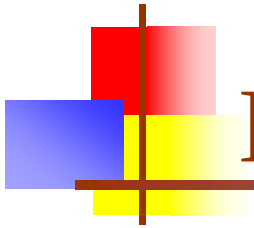
XSL Vs CSS

- XSL and Cascading Style Sheets (CSS) have similar goals
- XSL is more powerful than CSS in many ways, but it's also more complex
- XSL and CSS have two things in common:
- Each provides a mechanism for selecting elements and for specifying how the selected elements are to be presented. CSS uses selectors and properties in this way:
- *selector { properties; }*
- XSL uses patterns and formatting objects:
- <xsl:template pattern="pattern">
  <formatting objects/>
  </xsl:template>



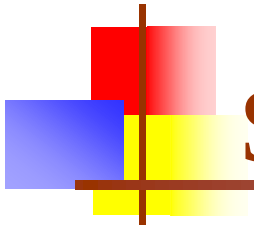
There are two key differences between XSL and CSS:

- CSS can be used to style HTML documents, while XSL cannot.
- XSL can be used to transform XML documents, while CSS cannot.



How does it work?

- The XML source document is parsed into an XML source tree
- You use XPath to define templates that *match* parts of the source tree
- You use XSLT to *transform* the matched part and put the transformed information into the result tree
- The result tree is output as a result document
- Parts of the source document that are not matched by a template are typically copied unchanged

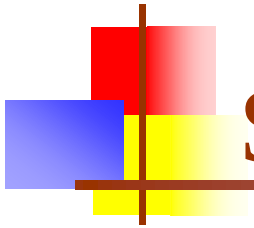


Simple XPath

- Here's a simple XML document:

```
<?xml version="1.0"?>
<library>
  <book>
    <title>XML</title>
    <author>Gregory Brill</author>
  </book>
  <book>
    <title>Java and XML</title>
    <author>Brett McLaughlin</author>
  </book>
</library >
```

- XPath expressions look a lot like paths in a computer file system
  - / means the document itself (but no specific elements)
  - /library selects the root element
  - /library/book selects *every* book element
  - //author selects *every* author element, wherever it occurs



Simple XSLT

- `<xsl:for-each select="//book">` loops through every `book` element, everywhere in the document

- `<xsl:value-of select="title"/>` chooses the *content* of the `title` element at the current location

- ```
  <xsl:for-each select="//book">
      <xsl:value-of select="title"/>
  </xsl:for-each>
  ```
  chooses the content of the `title` element for each `book` in the XML document
  ```

Using XSL to create HTML

- Our goal is to turn *this:*

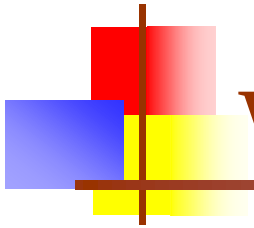
```
<?xml version="1.0"?>
<library>
  <book>
    <title>XML</title>
    <author>Gregory Brill</author>
  </book>
  <book>
    <title>Java and XML</title>
    <author>Brett McLaughlin</author>
  </book>
</library >
```

- Into HTML that displays something like *this:*

Book Titles:
- XML
- Java and XML

Book Authors:
- Gregory Brill
- Brett McLaughlin

- Note that we've grouped titles and authors separately



What we need to do

- We need to save our XML into a file (let's call it books.xml)
- We need to create a file (say, books.xsl) that describes how to select elements from books.xml and embed them into an HTML page
  - We do this by intermixing the HTML and the XSL in the books.xsl file
- We need to add a line to our books.xml file to tell it to refer to books.xsl for formatting information

books.xml, revised

```
<?xml version="1.0"?>
<?xml-stylesheet  type="text/xsl"  href="books.xsl"?>
<library>
  <book>
    <title>XML</title>
    <author>Gregory Brill</author>
  </book>
  <book>
    <title>Java and XML</title>
    <author>Brett McLaughlin</author>
  </book>
</library >
```

This tells you where to find the XSL file

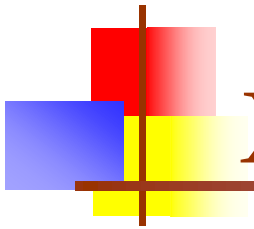
Desired HTML

```
<html>
  <head>
    <title>Book Titles and Authors</title>
  </head>
  <body>
    <h2>Book titles:</h2>
    <ul>
      <li>XML</li>
      <li>Java and XML</li>
    </ul>
    <h2>Book authors:</h2>
    <ul>
      <li>Gregory Brill</li>
      <li>Brett McLaughlin</li>
    </ul>
  </body>
</html>
```

Blue text is data extracted from the XML document

Brown text is our HTML template

We don't necessarily know how much data we will have



XSL outline

```
<?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> ... </html>

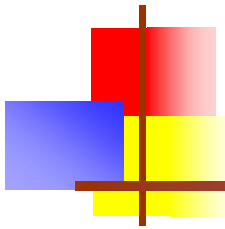
    </xsl:template>
</xsl:stylesheet>
```

Selecting titles and authors

```
<h2>Book titles:</h2>
  <ul>
    <xsl:for-each  select="//book">
      <li>
        <xsl:value-of  select="title"/>
      </li>
    </xsl:for-each>
  </ul>
<h2>Book authors:</h2>
  ...same thing, replacing title with author
```

Notice the xsl:for-each loop

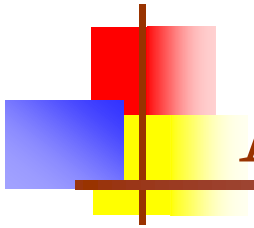
- Notice that XSL can rearrange the data; the HTML result can present information in a different order than the XML



All of books.xml

```
<?xml version="1.0"?>
<?xml-stylesheet  type="text/xsl"  href="books.xsl"?>
<library>
  <book>
    <title>XML</title>
    <author>Gregory Brill</author>
  </book>
  <book>
    <title>Java and XML</title>
    <author>Brett McLaughlin</author>
  </book>
</library >
```

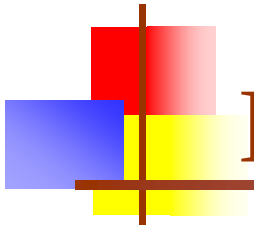
Note: if you do View Source, *this* is what you will see, not the resultant HTML



All of books.xsl

```
<?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>
 <head>
  <title>Book Titles and Authors</title>
 </head>
 <body>
  <h2>Book titles:</h2>
  <ul>
   <xsl:for-each select="//book">
    <li>
     <xsl:value-of select="title"/>
    </li>
   </xsl:for-each>
  </ul>
```

```
  <h2>Book authors:</h2>
   <ul>
    <xsl:for-each
          select="//book">
     <li>
      <xsl:value-of
            select="author"/>
     </li>
    </xsl:for-each>
   </ul>
 </body>
</html>
</xsl:template>
</xsl:stylesheet>
```



How to use it

- In a *modern* browser you can just open the XML file
  - Older browsers will ignore the XSL and just show you the XML contents as continuous text
- You can use a program such as Xalan, MSXML, or Saxon to create the HTML as a file
  - This can be done on the server side, so that all the client side browser sees is plain HTML
  - The server can create the HTML *dynamically* from the information currently in XML

The result (in IE)