XSL

What is XSL?
- XSL stands for EXtensible Stylesheet Language.
- XSL = Style Sheets for XML
- XSL describes how the XML document should be displayed!
- XSL - More Than a Style Sheet Language
- XSL consists of three parts:
  - XSLT - a language for transforming XML documents
  - XPath - a language for navigating in XML documents
  - XSL-FO - a language for formatting XML documents

What is XSLT?
- XSLT stands for XSL Transformations.
- XSLT is the most important part of XSL.
- XSLT transforms an XML document into another XML document.
- XSLT uses XPath to navigate in XML documents.

Explain XSL Transformation and XSL Elements
- The style sheet provides the template that transforms the document from one structure to another.
- In this case <xsl:template> starts the definition of the actual template, as the root of the source XML document.
- The match = "/" attribute makes sure this begins applying the template to the root of the source XML document.

Linking
- The style sheet is linked into the XML by adding the connecting statement to the XML document: 
  <?xml-stylesheet type="text/xsl" href="libstyle.xsl" ?>

XSL Transformations
- XSLT is the most important part of XSL.
- XSLT is used to transform an XML document into another XML document, or another type of document that is recognized by a browser, like HTML and XHTML. Normally XSLT does this by transforming each XML element into an (X) HTML element.
- With XSLT you can add/remove elements and attributes to or from the output file. You can also rearrange and sort elements, perform tests and make decisions about which elements to hide and display, and a lot more.
- A common way to describe the transformation process is to say that XSLT transforms an XML source-tree into an XML result-tree.
- XSLT Uses XPath:
  - XSLT uses XPath to find information in an XML document.
  - XPath is used to navigate through elements and attributes in XML documents.
- XSLT Works as:
  - In the transformation process, XSLT uses XPath to define parts of the source document that should match one or more predefined templates.
  - When a match is found, XSLT will transform the matching part of the source document into the result document.
XSL Elements
• XSL contains many elements that can be used to manipulate, iterate and select XML, for output.
  • value-of
  • for-each
  • sort
  • if
  • choose

<xsl:value-of> Element
• The <xsl:value-of> element extracts the value of a selected node.
• The <xsl:value-of> element can be used to select the value of an XML element and add it to the output.
• Syntax
  <xsl:value-of select="expression" />

  • expression: This is Required. An XPath expression that specifies which node/attribute to extract the value from. It works like navigating a file system where a forward slash (/) selects subdirectories.

<xsl:for-each> Element
• The XSL <xsl:for-each> element can be used to select every XML element of a specified node-set.

<xsl:if> Element
• To put a conditional if test against the content of the XML file, add an <xsl:if> element to the XSL document.
• Syntax
  <xsl:if test="expression">
    ...some output if the expression is true...
  </xsl:if>

<xsl:sort> Element
• The <xsl:sort> element is used to sort the output.
• <xsl:sort select="artist"/>
• The select attribute indicates what XML element to sort on.
Example using value-of, for-each and if

```xml
<?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>My CD Collection</h2>
<table border="1">
<tr bgcolor="#9acd32">
<th>Title</th>
<th>Artist</th>
</tr>
<xsl:for-each select="catalog/cd">
<xsl:if test="price &gt; 10">
<tr>
<td>
<xsl:value-of select="title"/>
</td>
<td xsl:value-of select="artist"/>
</tr>
</xsl:if>
</xsl:for-each>
</table>
</body>
</html>
</xsl:stylesheet>
```
<xsl:choose> Element
• The <xsl:choose> element is used in conjunction with <xsl:when> and <xsl:otherwise> to express multiple conditional tests.
• Syntax

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

<xsl:apply-templates> Element
• The <xsl:apply-templates> element applies a template to the current element or to the current element's child nodes.
• If we add a select attribute to the <xsl:apply-templates> element it will process only the child element that matches the value of the attribute. We can use the select attribute to specify the order in which the child nodes are processed.
• Look at the following XSL style sheet:

```xml
<?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>My CD Collection</h2>
<xsl:apply-templates/>
</body>
</html>
</xsl:template>

<xsl:template match="cd">
<p>
<xsl:apply-templates select="title"/>
<xsl:apply-templates select="artist"/>
</p>
</xsl:template>
</xsl:stylesheet>
```
Explain transforming with XSLT.

- It is possible to convert an XML document to XHTML using the browser’s own parser. However, this is not always possible:
  - The browser at the client end may not be suitable or equipped to do the transformation.
  - It may not be a good idea to include the reference to the style sheet or even have the style sheet available!
- The answer to this process the document and style sheet outside of the browser’s own mechanism for doing this task.
- This task can be done either on the client side or the server side.

Using JavaScript

- One way to process and transform XML on the client side is using JavaScript, which has several features for doing the task very well.

```html
<html>
<body>
<script type="text/javascript">
//Load the XML document
var xml= new ActiveXObject("Microsoft.XMLDOM")
xml.async=false
xml.load("lib.xml")
//Load the XSL document
var xsl= new ActiveXObject("Microsoft.XMLDOM")
xsl.async= false
xsl.load("libstyle.xsl")
//Do the actual transform
document.write(xml.transformNode(xsl))
</script>
</body>
</html>
```

- Above example shows one way to transform with JavaScript using Microsoft’s proprietary Application Programming Interface (API) for the Internet Explorer browser.
- It is also possible to process XML using the DOM.
- Using both the of these mechanisms it is possible to also traverse an XML document and process either according to a style sheet or simply using the JavaScript to make the stylistic decisions.
- Apart from JavaScript, it is also possible to use other programming languages (such as Java and .Net) to process and then output a transformed document.