XSL (eXtensible Stylesheet Language) is a styling language for XML.
XSLT stands for XSL Transformations.

XSL(T) Languages

XSLT is a language for transforming XML documents.
XPath is a language for navigating in XML documents.
XQuery is a language for querying XML documents.

It Started with XSL

XSL stands for E**X**tensible **S**tylesheet **L**anguage.
The World Wide Web Consortium (W3C) started to develop XSL because there was a need for an XML-based Stylesheet Language.

CSS = Style Sheets for HTML

HTML uses predefined tags. The meaning of, and how to display each tag is well understood.
CSS is used to add styles to HTML elements.

XSL = Style Sheets for XML

XML does not use predefined tags, and therefore the meaning of each tag is not well understood.
A <table> element could indicate an HTML table, a piece of furniture, or something else - and browsers do not know how to display it!
So, XSL describes how the XML elements should be displayed.

XSL - More Than a Style Sheet Language

XSL consists of four parts:

- XSLT - a language for transforming XML documents
- XPath - a language for navigating in XML documents
- XSL-FO - a language for formatting XML documents (discontinued in 2013)
- XQuery - a language for querying 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
- XSLT is a W3C Recommendation

XSLT = 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.
If you want to study XPath first, please read our XPath Tutorial.

How Does it Work?

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.

XSLT Browser Support

All major browsers support XSLT and XPath.

XSLT is a W3C Recommendation

XSLT became a W3C Recommendation 16. November 1999.

Correct Style Sheet Declaration

The root element that declares the document to be an XSL style sheet is <xsl:stylesheet> or <xsl:transform>.
Note: <xsl:stylesheet> and <xsl:transform> are completely synonymous and either can be used!
The correct way to declare an XSL style sheet according to the W3C XSLT Recommendation is:

```
<xsl:stylesheet version="1.0"
xmlns:xsl="http://www.w3.org/1999/XSL/Transform">
```

or:

```
<xsl:transform version="1.0"
xmlns:xsl="http://www.w3.org/1999/XSL/Transform">
```

To get access to the XSLT elements, attributes and features we must declare the XSLT namespace at the top of the document.

The xmlns:xsl="http://www.w3.org/1999/XSL/Transform" points to the official W3C XSLT namespace. If you use this namespace, you must also include the attribute version="1.0".

Start with a Raw XML Document

We want to **transform** the following XML document ("cdcatalog.xml") into XHTML:

```
<?xml version="1.0" encoding="UTF-8"?>
<catalog>
  <cd>
    <title>Empire Burlesque</title>
    <artist>Bob Dylan</artist>
    <country>USA</country>
    <company>Columbia</company>
    <price>10.90</price>
    <year>1985</year>
  </cd>
.
.
</catalog>
```

Create an XSL Style Sheet

Then you create an XSL Style Sheet ("cdcatalog.xsl") with a transformation template:

```
<?xml version="1.0" encoding="UTF-8"?>
<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">
    <tr>
      <td><xsl:value-of select="title"/></td>
      <td><xsl:value-of select="artist"/></td>
    </tr>
    </xsl:for-each>
  </table>
  </body>
  </html>
</xsl:template>

</xsl:stylesheet>
```

Link the XSL Style Sheet to the XML Document

Add the XSL style sheet reference to your XML document ("cdcatalog.xml"):

```
<?xml version="1.0" encoding="UTF-8"?>
<?xml-stylesheet type="text/xsl" href="cdcatalog.xsl"?>
<catalog>
  <cd>
    <title>Empire Burlesque</title>
    <artist>Bob Dylan</artist>
    <country>USA</country>
    <company>Columbia</company>
    <price>10.90</price>
    <year>1985</year>
  </cd>
.
.
</catalog>
```

An XSL style sheet consists of one or more set of rules that are called templates.
A template contains rules to apply when a specified node is matched.

The <xsl:template> Element

The <xsl:template> element is used to build templates.
The **match** attribute is used to associate a template with an XML element. The match attribute can also be used to define a template for the entire XML document. The value of the match attribute is an XPath expression (i.e. match="/" defines the whole document).
Ok, let's look at a simplified version of the XSL file from the previous chapter:

```
<?xml version="1.0" encoding="UTF-8"?>
<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>
    <tr>
      <td>.</td>
      <td>.</td>
    </tr>
  </table>
  </body>
  </html>
</xsl:template>

</xsl:stylesheet>
```

Example Explained

Since an XSL style sheet is an XML document, it always begins with the XML declaration: **<?xml version="1.0" encoding="UTF-8"?>**.

The next element, **<xsl:stylesheet>**, defines that this document is an XSLT style sheet document (along with the version number and XSLT namespace attributes).

The **<xsl:template>** element defines a template. The **match="/"** attribute associates the template with the root of the XML source document.

The content inside the <xsl:template> element defines some HTML to write to the output.

The last two lines define the end of the template and the end of the style sheet.

The result from this example was a little disappointing, because no data was copied from the XML document to the output. In the next chapter you will learn how to use the **<xsl:value-of>** element to select values from the XML elements.

The <xsl:value-of> element is used to extract the value of a selected node.

The <xsl:value-of> Element

The <xsl:value-of> element can be used to extract the value of an XML element and add it to the output stream of the transformation:

```
<table border="1">
  <tr bgcolor="#9acd32">
    <th>Title</th>
    <th>Artist</th>
  </tr>
  <tr>
    <td><xsl:value-of select="catalog/cd/title"/></td>
    <td><xsl:value-of select="catalog/cd/artist"/></td>
  </tr>
</table>
```

Example Explained

Note: The **select** attribute, in the example above, contains an XPath expression. An XPath expression works like navigating a file system; a forward slash (/) selects subdirectories.
The result from the example above was a little disappointing; only one line of data was copied from the XML document to the output. In the next chapter you will learn how to use the **<xsl:for-each>** element to loop through the XML elements, and display all of the records.

The <xsl:for-each> element allows you to do looping in XSLT.

The <xsl:for-each> Element

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

```
<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">
    <tr>
      <td><xsl:value-of select="title"/></td>
      <td><xsl:value-of select="artist"/></td>
    </tr>
    </xsl:for-each>
  </table>
  </body>
  </html>
</xsl:template>
</xsl:stylesheet>
```

The value of the **select** attribute is an XPath expression. An XPath expression works like navigating a file system; where a forward slash (/) selects subdirectories.

Filtering the Output

We can also filter the output from the XML file by adding a criterion to the select attribute in the <xsl:for-each> element.
<xsl:for-each select="catalog/cd[artist='Bob Dylan']">
Legal filter operators are:

- = (equal)
- != (not equal)
- < less than
- > greater than

Take a look at the adjusted XSL style sheet:

```
<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[artist='Bob Dylan']">
    <tr>
      <td><xsl:value-of select="title"/></td>
      <td><xsl:value-of select="artist"/></td>
    </tr>
    </xsl:for-each>
  </table>
  </body>
  </html>
</xsl:template>
```

The <xsl:sort> element is used to sort the output.

Where to put the Sort Information

To sort the output, simply add an <xsl:sort> element inside the <xsl:for-each> element in the XSL file:

```
<xsl:for-each select="catalog/cd">
  <xsl:sort select="artist"/>
  <tr>
    <td><xsl:value-of select="title"/></td>
    <td><xsl:value-of select="artist"/></td>
  </tr>
</xsl:for-each>
```

The <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>
```

Where to Put the <xsl:if> Element

To add a conditional test, add the <xsl:if> element inside the <xsl:for-each> element in the XSL file:

```
<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"/></td>
      <td><xsl:value-of select="price"/></td>
    </tr>
  </xsl:if>
</xsl:for-each>
```

The <xsl:choose> element is used in conjunction with <xsl:when> and <xsl:otherwise> to express multiple conditional tests.

The <xsl:choose> Element

Syntax

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

Where to put the Choose Condition

To insert a multiple conditional test against the XML file, add the <xsl:choose>, <xsl:when>, and <xsl:otherwise> elements to the XSL file:

```
<xsl:for-each select="catalog/cd">
<tr>
  <td><xsl:value-of select="title"/></td>
  <xsl:choose>
    <xsl:when test="price &gt; 10">
      <td bgcolor="#ff00ff">
      <xsl:value-of select="artist"/></td>
    </xsl:when>
    <xsl:otherwise>
      <td><xsl:value-of select="artist"/></td>
    </xsl:otherwise>
  </xsl:choose>
</tr>
</xsl:for-each>
```

The <xsl:apply-templates> element applies a template to the current element or to the current element's child nodes.

The <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 version="1.0" encoding="UTF-8"?>
<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:template match="title">
  Title: <span style="color:#ff0000">
  <xsl:value-of select="."/></span>
  <br />
</xsl:template>

<xsl:template match="artist">
  Artist: <span style="color:#00ff00">
  <xsl:value-of select="."/></span>
  <br />
</xsl:template>

</xsl:stylesheet>
```