Lecture 10

This week our main focus is going to be on XML. You may have heard of *Extensible Markup Language (XML)* and the related alphabet soup related to XML: *XSL*, *XSLT*, *XSD*, *XSL-FO*, *XHTML*, *SOAP*, *AJAX*, and so on. Although the focus of this course is JavaScript and DHTML, I would feel negligent as an instructor if you were to leave this course without some familiarity with these related technologies.

XML

In order to really grasp the usefulness of XML we need to understand the shortcomings of HTML. HTML is a markup language for representing display and formatting rules for data. HTML does not *describe* or *represent* the data itself, though.

XML is just the opposite. It is intended to *describe* data, but not to dictate or suggest anything about how the data is to be *displayed* or *formatted*.

Like HTML, XML uses a series of *tags* for markup. Here is a sample XML document:

Example

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<CATALOG>
<CD>
<TITLE>Germfree Adolescents</TITLE>
<ARTIST>X-Ray Spex</ARTIST>
<COUNTRY>UK</COUNTRY>
<COMPANY>EMI</COMPANY>
<PRICE>14.00</PRICE>
<YEAR>1978</YEAR>
</CD>
<CD>
<TITLE>Entertainment!</TITLE>
<ARTIST>Gang of Four</ARTIST>
<COUNTRY>UK</COUNTRY>
<COMPANY>EMI</COMPANY>
<PRICE>6.99</PRICE>
<YEAR>1979</YEAR>
</CD>
<CD>
<TITLE>Escape From Noise</TITLE>
<ARTIST>Negativland</ARTIST>
<COUNTRY>SST</COUNTRY>
<COMPANY>Slash</COMPANY>
<PRICE>8.99</PRICE>
<YEAR>1987</YEAR>
</CD>
</CATALOG>
```

Notice that I don't have to tell you *anything* about this data. By looking at the XML itself, you can quickly glean the meaning of the data itself. This is the nature and purpose of XML. Since the markup describes the data, the data itself can be interpreted quite easily. Take a look at this document for more information about

self-describing data:

- The Importance of Self-Describing Documents

Another wonderful side effect of XML is the fact that, since it is just *text*, we can very easily transport it via *HTTP (Hypertext Transfer Protocol, the format that allows resources to be shared over a client/server connection)*. What this means is that we have a transport mechanism for sending data from machine to machine regardless of distance, platform, or security restrictions.

XML is *platform independent* by nature. Any computer can read text. This means that *UNIX*, *WinTel*, *Mac/OS X* and other machines can all communicate with one another via predefined XML data streams.

Since XML is text, and since we can transport text via HTTP, this means we can now send messages through firewalls. If you have ever attempted to create enterprise applications that communicate across machines you know this can be difficult. Some mechanisms that exist to do this include:

- Windows Sockets
- DCOM
- CORBA
- And others

All of these have their difficulties. None of them allow cross-platform communications or communications through firewalls (without jumping through hoops, at least). XML rids us of these shortcomings.

Another important thing to discuss is the fact that HTML has a well-defined set of expected *tags*. The tags used for XML, on the other hand, are completely open ended and are determined by the *user/author* of the XML document - hence the *extensible* part of its name.

This means you can create any tags you want to describe any type of data you want. The only caveat is that in order to consume any document you have written, the "consumer" must know in advance what tags you will be using. In other words, you must have some means of communicating what tags you will be using and what the meanings of those tags will be. As long as you have communicated this information in advance to any and all consumers of your XML document, those consumers should be able to write application software that consumes your XML document and processes it accordingly.

Note: There are *some* conventions in place for XML documents, regardless of the freedom we have in creating them. Search "Well-Formed XML" for more details.

You can view an XML in any *XML-equipped* browser (essentially any modern browser).

CD catalog example

Save this file from your browser - you will use it later in your assignment:

- XML CD Catalog

Although the XML specification does not publish standard tags (as discussed above), there are hundreds (if not *thousands*) of industry standard *XML Schemas* that have been published so that companies and others can engage in business-to-business transactions with agreed-upon "standard" XML documents. There are published XML specification standards for data from any industry you can think of, such as:

- healthcare
- credit reporting
- shipping and receiving
- retail
- finance
- etc.

Check out this Wikipedia page for a taste of the variety of standards out there:

- Category: XML-based standards

XSLT

So now we know what XML is, and we understand that XML can be used to send data between two machines. Clearly, this provides a perfect mechanism for *business-to-business (B2B)* computing and application development.

Extensible Stylesheet Language Transformations (XSLT) is a related technology that is very impressive and quite useful. It allows us to take an XML document and *transform* it into something else. The end result could be an HTML page, an EDI file, a PDF, an Excel spreadsheet, or one of a great number of other file types.

An example use case for this can be found in the B2B market. Suppose you manufacture widgets, and you deal with ten different retailers that sell your widgets. The individual retailers each submit purchase orders to you via XML and HTTP. You then take the XML documents received and process them; here that means creating records in your "Orders" database for the orders of widgets.

Now here is where a problem emerges: your order entry software expects XML to be in a very specific format if it is to parse this XML and add the necessary data to your database. This is complicated by the fact that each of your retail customers sends XML purchase order data to you in some kind of proprietary format. In fact, as your bring on new retailers, you can likewise expect their XML formats to be proprietary. The open ended nature of XML makes it impossible in practice to expect each retailer to use the exact same XML format.

This is where XSLT comes to the rescue.

Instead of changing the code in your order entry software to accomodate each new retailer, you can simply create an *XSLT file* for each retailer's specific XML document format; this file will take their XML input and convert it to your own standard XML format for purchase orders. Your oder entry application can subsequently process the *converted* version of the data consumed from these external standardized documents.

The topic of XSLT is large enough that it prohibits detailed explanation herein. Refer to the *XSLT Tutorial* on **w3schools.com** to read more about XSLT. As you read through this tutorial, pay close attention to the `<xsl:for-each>` and `<xsl:value-of>` elements.

Remember that, unlike HTML, XSL is *case-sensitive*. Consequently, you need to be careful when building your XSL templates, and when specifying element names in your searches. The element name **cd** is *not* the same as the element name **CD**, for example.

Here is the link to the example XSLT Stylesheet. Download and save just as you did the XML document above.

Cd catalog XSL sheet

- cdcatalog.xsl
- Since an XSL sheet is, technically, an XML document in its own right, you can view this document in an XML-equipped browser (Firefox, Chrome, Safari, IE, etc)
- Save this document for your lab exercises.

To instruct an XML document to use a specific XSL stylesheet, simply add a reference to the *XSL template* in the XML document itself: `<?xml-stylesheet type="text/xsl" href="cdcatalog.xsl"?>`

The following link demonstrates the **CD Catalog** XML Document after styling with a simple XSLT stylesheet:

- cdcatalog_with_xsl.xml

XSD

XML Schema Definition (XSD) is a way of formally describing the expected elements in a given XML document. As an example, if you are building an application that processes student records and you need to send some of this student data to a third party vendor that handles financial aid processing, you would likely want to send and receive XML documents back and forth between your company and the vendor.

If you and the vendord id not have formalized, specific definitions of what those XML documents are to look like, you would not be able to develop applications to integrate with them.

XSD is the XML-based schema definition language that allows you to communicate your standard with your vendors.

Like XSLT, XSD merits a course all to itself. To learn more about XSD, go to **w3schools.com** and look for the "XSD Schema Tutorial".