Week 2 Lecture : XML Applications, Creating XML Documents, and CSS!

Screencast of Recorded Lecture (2022):

- https://sjeccd-edu.zoom.us/rec/play/SpkbT2sxdszDBlyhABzp0SQicWHO1YMFMGmvqdlgdaF03JFUuTofUabquv4l83fX1jUaJAepNIYiHIvH.JRUEZKPf8VY_erCI (https://sjeccd-edu.zoom.us/rec/play/SpkbT2sxdszDBlyhABzp0SQicWHO1YMFMGmvqdlgdaF03JFUuTofUabquv4l83fX1jUaJAepNIYiHIvH.JRUEZKPf8VY_erCI)

Screencast of Recorded Lecture (2021):

- https://sjeccd-edu.zoom.us/rec/play/UZg1k1Qd8an-JmAyZa4ua34_czsC4UTjr2axW8j-L8rnGLJsJ0WgqaUfxTqzltCtpF-qeSCzfYjOh0e_.jMhyZ6pYXBlm2_4G (https://sjeccd-edu.zoom.us/rec/play/UZg1k1Qd8an-JmAyZa4ua34_czsC4UTjr2axW8j-L8rnGLJsJ0WgqaUfxTqzltCtpF-qeSCzfYjOh0e_.jMhyZ6pYXBlm2_4G)

First off, I'd like to talk for a few minutes about XML applications – how and where XML is used. XML is a meta-markup language for designing domain-specific markup languages. This means that the XML syntax (language, words, etc.) is written in the vocabulary of the field that it is used in, such as Music, Math, etc.

This is a tremendous advantage of XML and unique to XML and one reason it is becoming so popular for transferring documents and data on the internet.

Each XML application has its own syntax and vocabulary. This syntax and vocabulary adheres to the fundamental rules of XML. This is much like human languages, each which has its own vocabulary and grammar, while adhering to certain fundamental rules imposed by human anatomy and the structure of the brain.

XML is an extremely flexible format for text-based data. The reason XML was chosen as the foundation for the wildly different applications is that XML provides a sensible, well-documented format that's easy to read and write. By using this format for its data, a program can offload a great quantity of detailed processing to a few standard free tools and libraries. Furthermore it's easy for such a program to layer additional levels of syntax semantics on top of the basic structure XML provides.

Chemical Markup Language (CML)

Molecular documents often contain thousands of different, very detailed objects. For example, a single medium-sized organic molecule may contain hundreds of atoms, each with a least one bond, and many with several bonds, to other atoms in the molecule. CML seeks to organize these complex chemical objects in a straight forward manner that can be understood, displayed, and searched of a computer.

Mathematical Markup Language (MathML)

Until XML there hasn't been any good way to include equations in Web pages. MathML is an XML application for mathematical equations. MathML is sufficiently expressive to handle most math – from grammar-school arithmetic through calculus and differential equations.

Channel Definition Format (CDF)

Microsoft's Channel Definition Format (CDF) is an XML application for defining channels. Web sites use channels to upload information to readers who subscribe to the site rather than waiting for them to come and get it. This is alternately called webcasting or push. CDF was first introduced in Internet Explorer 4.0.

Synchronized Multimedia Integration Language (SMIL)

The Synchronized Multimedia Integration Language (SMIL, pronounced "smile") is a W3C-recommended XML application for writing "TV-like" multimedia presentations for the web. SMIL documents don't describe the actual multimedia content (that is the video and sound that are played). Instead SMIL documents describe when and where particular video and audio clips are played.

Open Software Description (OSD)

The Open Software Description (OSD) format is an XML application that was co-developed by Marimba and Microsoft to update software automatically. OSD defines XML tags that describe software components. The description of a component includes the version of the component, its underlying structure, and its relationships to and dependencies on other components. This provides enough information to decide whether a user needs a particular update.

Scalable Vector Graphics (SGV)

Scalable Vector Graphics (SGV) are better than bitmaps for many kinds of pictures including flow charts, cartoons, assembly diagrams, blueprints, and more.

Vector Markup Language (VML)

Microsoft has developed its own XML application for vector graphics called the Vector Markup Language (VML).

Music Markup Language (MusicML)

The Connection Factory has created an XML application for sheet music called MusicML. MusicML includes notes, beats, clefs, staffs, rows, rhythms, rests, beams, rows, chords, and more.

Voice Markup Language (VoiceXML)

VoiceXML is an XML application for the spoken work. In particular, it's intended for those annoying voice mail and automated phone response systems. VoiceXML enables the same data that's used on a Web site to served up via telephone. It's particularly useful for information that's created by combining small nuggets of data, such as stock prices, sports scores, weather reports, airline schedules, and test results.

This just scratches the surface of the use of XML for internal data. Many other projects that use XML are just getting started and many more will be started over the next several years. Most of these won't receive any publicity or write-ups in the trade press, but they nonetheless have the potential to save their companies millions of dollars in development costs over the life of the project. The self-documenting nature of XML can be as useful for a company's internal data as for its external data. For instance, recently many companies were scrambling to try to figure out whether programmers who retired 20 years ago used two-digit or four-digit dates. If that were you job, would you rather be pouring over data that looked like this?

```
    3c  79  65  61  72  3e  39  39  3c  2f  79  65  61  72  3e
```

or that looked like this?

```
    <YEAR>99</YEAR>
```

Binary like formats meant that programmers were stuck trying to clean up data in the first format. XML even makes the mistakes easier to find and fix.

The Life of an XML Document

It may be helpful to think about the life of an XML document as flowing through three stages: create, parse, and consume.

Stage 1: Creating an XML document. The first stage is to create an XML document either by hand using an editor, such as Notepad, or to export data from another software application into XML format. Many databases now allow you to export data as XML documents, along with such everyday software applications as Microsoft Excel or Word.

Stage 2: Parsing an XML document. After you've created the XML document, the second stage is to parse (or process) the document to check to see whether the document structure and content is acceptable for using the document.

Stage 3: Consuming the XML document. The final stage is actually doing something with the XML document that has been "approved" by the parser. Whereas an HTML document has one primary purpose – display is a Web browser – an XML document actually has several possible ways of being utilized by a variety of applications.

I would like to say a few more words about the W3C since we will be talking about it a lot more in

later sessions.

Just exactly what is the W3C? XML was developed by a group known as the World Wide Web Consortium, or the W3C. The W3C is a group of Web stakeholders, including universities, companies, and other technology geeks that come together under the W3C umbrella to define Web-related standards (known technically as “W3C recommendations”). In addition to XML, the W3C defines recommendations for HTML and many XML-related technologies that we will be looking at during this course, such as XSLT (Extensible Stylesheet Language Transformations), XPath, and XLink.

Our First XML Document

I am going to follow an old programmer’s tradition of introducing a new language with a program that prints “Hello World” on the screen. XML is a markup language, not a programming language, but the basic principle still applies. It’s easiest to get started if we begin with a complete working example that we can build on, rather than starting with more fundamental pieces that by themselves don’t do anything.

Let’s create the following simple XML document and save it in a file. This is about as simple as you can get.

```
<?xml version="1.0"?>
<GREETING>
  Hello XML World!
</GREETING>
```

This is not very complicated, but it is a good XML document. To be more precise, it is a **well-formed** XML document..

Type the above XML statements in your text editor (Notepad/Brackets/Atom/Notepad++/etc), and save it in a file called **hello.xml**. The three letter extension .xml is the standard for XML files. However, do make sure that you save it in plain-text format, and not in the native format of a word processor such as Microsoft Word. If you are using Notepad to edit your files, be sure to enclose the filename in double quotes when saving the document, for example “hello.xml”, not merely hello.xml. Without the quotes, Notepad will append the .txt extension to your file name, naming it hello.xml.txt, which is not what you want.

Loading the XML file into your Web browser

Once we have created the XML document, we'll want to look at it. The file can be opened directly in a browser that supports XML, such as Firefox, Google Chrome, or IE. What you see will vary from browser to browser. It should be a nicely formatted view of the document's source code. To see the document in its intended format you will have to help out the browser. The browser doesn't really know what to do with the GREETING tag. You have to tell the browser how to handle each element by adding a style sheet, which we'll talk about shortly.

Now, let's take a closer look at our first XML document.

The first line is called the XML declaration, sometimes even referred to as the prolog.

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

The XML declaration has a version attribute. An attribute is a name-value pair separated by an equals sign. The name is on the left side of the equals sign, and the value is on the right side between the double quotes. Every XML document should begin with an XML declaration that specifies the version of XML in use. In our first XML document the version attribute says that this document conforms to the XML 1.0 specification.

Looking at the next three lines:

```
<GREETING>
Hello XML World!
</GREETING>
```

Collectively these three lines form a **GREETING** element. Separately, <GREETING> is a start tag, </GREETING> is an end tag, and the Hello XML World! is the content of the GREETING element. Divided another way, the start tag, end tag, and XML declaration are all markup. The text **Hello XML World!** is character data.

Creating a Style Sheet for an XML Document

XML allows you to create any tags that you need. Since you have almost complete freedom in

creating tags, there's no way for a generic browser to anticipate your tags and provide rules for displaying them. Therefore, you also need to write a style sheet for the XML document that tells browsers how to display particular tags. Like tag sets, style sheets can be shared between different documents and different people, and the style sheets you create can be integrated with style sheets others have written. There is more than one style sheet language to choose from. We will start with by using Cascading Style Sheets (CSS). CSS has the advantage of being an established W3C standard.

The following is a very simple style sheet that specifies that the contents of the GREETING element should be rendered as a block-level element in 24-point type.

```
GREETING {display: block; font-size: 24pt; font-weight: bold; color: pink; background: purple;}
```

The above should be typed in a text editor and saved in a new file called **greeting.css** in the same directory as the XML document we just created. The .css extension stands for Cascading Style Sheets. The .css extension is important.

Using CSS, you can greatly modify the look and feel of your XML document. We'll cover this in greater detail throughout the course, but for the moment, let's stick to the basic font choices.

- To make an element appear on its own line, use the *display: block;* rule.
- To change an element's font size, use the *font-size* property with a value of the size you want (e.g., *font-size: 12pt;* or *font-size: 200%;* or *font-size: 2em;*)
- To make an element bold, use the *font-weight: bold;* rule
- To make an element italic, use the *font-style: italic;* rule
- To change an element's font color, use the *color: colorname* rule (e.g., *color: pink;* or *color: purple;*)
- To change the background of an element, use the *background: colorname* rule (e.g., *background: pink;* or *background: purple;*)

Once you have written an XML document and a Cascading Style Sheet for that document, you need to tell the browser to apply the style sheet to the document. There are a number of ways to do this, but we will be including a **<?xml-stylesheet?>** processing instruction in the XML document to specify the style sheet to be used.

The <?xml-stylesheet?> processing instruction has two required attributes: **type** and **href**. The type attribute specifies the style sheet language used, and the href attribute specified a URL, possibly relative, where the style sheet can be found.

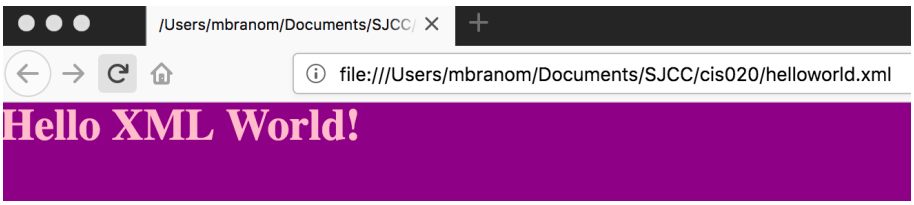
Our final XML document (*helloworld.xml*) should now look like the following:

```
<?xml  version="1.0"?>
<?xml-stylesheet type="text/css" href="greeting.css"?>
<GREETING>
Hello XML World!
</GREETING>
```

And our *greeting.css* file should look like this:

```
GREETING  {display:  block;  font-size;  24pt;  font-weight:  bold; color: pink; background: purple;}
```

And when rendered in an XML-compliant browser (like Firefox), it'd look like this:



If you have never played around with HTML and CSS before, here is a little mini-lecture on Cascading Style Sheets you should read through. It covers the basics of CSS (and some of the newer CSS3 rules):

CSS (Cascading Style Sheets) (https://sjeccd.instructure.com/courses/39453/pages/css-cascading-style-sheets)

Well, I think that's enough for now.