COIN-078B : Internet Programming with XML

Assignment 3

Objectives:

- Learn how to handle XML documents with clien-side JavaScript using the Document Object Model (DOM).
- Load, parse and transform XML documents with XSLT stylesheets programmatically.

Alternative: The example implementation below is for MSIE6 or above browser. If you'd rather develop for the Firefox browser, please read "Processing XML Documents with JavaScript in Mozilla". Indicate that your apps is for FF in the index page.

Note: Please follow the specifications below. Points will be knocked off if not followed.

Read Assignment Guidelines & Requirements

The W3C DOM is a platform- and language-neutral interface that allows programs and scripts to access and update the content, structure and style of a document. The W3C DOM includes a model for how a standard set of objects representing HTML and XML documents are combined, and an interface for accessing and manipulating them. In this assignment you are going to write a Web application that provides a user-interface allowing user to dynamically load, view, update and transform XML documents with different XSLT stylesheets, and display its output. Using HTML create a frame-based UI containing a control panel (**a3_controls.htm**) on the left side and a larger content pane (**a3_results.htm** which is empty) on the right. Download the frameset file **assign3.htm** and delete the <pre> tags if present. The control panel shows a heading named "**XML\XSL Viewer**", followed by four sections namely **Load XML Data Files**, **Load XSL Stylesheets**, **Transform To** and **Sort By**. Use table to layout these sections. In each of the "Load..." cell are tags containing the appropriate filenames and onclick event handlers. The "Transform To" cell has two buttons labeled "**Browser**" and "**Source**"; while "**Sort By**" has a textbox, two dropdown select boxes - one labeled "**data-type:**" with two options (values equal `"text"` and `"number"` with the former `SELECTED`), and the other labeled "**order:**" with two options (values equal `"ascending"` and `"descending"`); and a "**Sort**" button. See the screenshots.

Clicking the following: (apply onclick event handlers in tags)

files in the "Load XML..." section
- will load the source file and display the text content in the results frame.

files in the "Load XSL..." section
- will load the stylesheet file and display the text content in the results frame.

Browser button in the "Transform To" section
- transformation process occurs and renders the output in browser display.

Source button in the "Transform To" section
- transformation process occurs and display the output in text.

Note: both XML and XSL files must be loaded by clicking on the files before transformation can occur - ie. do not proceed to transform if `documentElement` of either source xml or xslt document is null.

Sort By dropdown select box

- contains the field options to sort.

data-type: dropdown select box
- for user to choose "text" or "number". **Note:** place these as value="text", for example, in the option tag.

order: dropdown select box
- for user to choose "ascending" or "descending"

Sort button
- will modify the `<xsl:sort>` attribute values of `select`, `data-type` and `order` before transformation and then renders to browser.
Hint: When calling setSortAttr() in the onclick event handler of the "Sort" button, use `form.id_of_input.value` to get the value of each input/select box. `id_of_input` is whatever id you assigned to each `<input>` or `<select>` tag. Make sure the `<option>` elements of `<select>` have assigned `value` attributes. Pass the three input values as parameters when invoking the setSortAttr() function.

Using the Dynamic HTML (DHTML) Object Model together with W3C DOM supported in Internet Explorer, define JavaScript functions in the <script> element of the <head> region of **a3_controls.htm**.

loadXML(file)
- loads the **source** object with the passed-in xml file; then `writeResults("<xmp>" + source.xml + "</xmp>");` **Note:** the .xml is not a file extension, but rather a misnamed property that converts XML document object to string.

loadXSL(file)
- loads the **xslt** object with the passed-in xsl file; then `writeResults("<xmp>" + xslt.xml + "</xmp>");`

init()
- instantiates the XMLDOM parser for **source** and **xslt** objects (see Week 5 - Part 1) and invoke this in the `onload` handler of the <body> tag.
Note: Do not load file in this function!

output(as_source)
- `writeResults( source.transformNode( xslt ) )` if `as_source` is false; concatenate `source.transformNode( xslt )` in <xmp> tag as above if `as_source` is true

setSortAttr(select_val, datatype_val, order_val)
- use `xslt.getElementsByTagName( "element_name_including_prefix" )` and `setAttribute( attrName, value )` on select, data-type & order as shown in class to modify these attribute values of <xsl:sort>. Make sure to call output() at the end.

writeResults(text)
- writes the passed-in text into the results frame

```
function writeResults( text )
{
  parent.results.document.open();
  parent.results.document.writeln(text);
  parent.results.document.close();
}
```

Extra Credits (5 pts): Define and use methods that would dynamically move the colored background to the sorted column as shown in the screenshots. **Hints:** Apply CSS background-color using `class` in the sorted field <td> in the xslt document. Use DOM APIs to find the <td> containing the `class` attribute, remove it and set it to a newly sorted <td>.

You'll need to read Chapter 14 if you're not familiar with JavaScript & Chapter 12 for the DOM API. To quickly retrieve a node such as <xsl:sort> without having to traverse the document from the

root, simply use `getElementsByTagName()` method. See Week 5 - Part topic page for a list of useful DOM properties and methods you can use to do this assignment.

Build the HTML frames first; then the control panel UI. It's visually helpful to give it a background color. Add the table, div tags and form elements. View in IE constantly to check the layout and alignments. Apply temporary event handlers like onclick="alert('test1')" to check the interactivity. When the basic UI performs correctly, write the functions one at a time, invoke a function in the proper event handler and test it. An incremental approach can make it easier to swallow.

Use elements.xml and examples from the book or from other sources for the XML data files, and raw-xml.xsl, defaultss.xsl, assign1.xsl, assign2.xsl and other xslt samples. Save the controls page as **a3_controls.htm**. Remember to create an empty **a3_results.htm**. Test your apps thoroughly.

Upload all HTML files, XML documents and XSLT stylesheets into their appropriate directories of your CTIS website. Add the links to all these files in your instruct.htm file.

Submit a hard copy of your **a3_controls.htm** source code with the url where your pages can be accessed on the due date. (Classroom section ONLY!)

Don't forget to backup your local files.

Helpful Resources:

- Handling XML Documents with JavaScript
- MSIE Dynamic HTML (DHTML) Object Model
- MSIE W3C DOM
- Microsoft XML Developer Center

Due Week 6