COIN-078B
Internet Programming with XML

Handling XML Documents with JavaScript using MSXML3.0

Module Objectives

- Know the W3C Document Object Model (DOM) and the XML DOM objects
- Learn to load and parse XML documents with JavaScript using MSXML3
- Handle events while loading XML documents
- Transform XML documents with XSLT stylsheets dynamically
- Update XML components using XML DOM API

Reading Assignments:

- **Textbook:** Chapters 12 & 14

Microsoft XML Core Services (MSXML)

Microsoft XML Core Services (MSXML) is a COM-based library that allows developers who use JScript, VBScript, and Microsoft Visual Studio 2005 (or later) to build Windows-native XML-based applications. It supports XML 1.0, DOM, SAX, an XSLT 1.0 processor, XML schema support including XSD and XDR, as well as other XML-related technologies.

Loading an XML Document

When you load and parse an XML document, you instantiate an ActiveX object library, `Microsoft.XMLDOM` (or `Msxml2.DOMDocument.3.0` to specify the MSXML version), that knows how to load and parse the document. After the object has been instantiated, you can command it to start working. Here are the steps to follow:

1. Instantiate the XML parser using the **ActiveXObject()** function.

```
xmldoc = new ActiveXObject("Microsoft.XMLDOM");
```

2. Set the **async** property on the object to false to ensure that the XML document is fully loaded (barring errors) before proceeding.

```
xmldoc.async = false;
```

3. Load a specific XML document.

```
xmldoc.load("path/some.xml");
```

Transforming an XML Document with XSLT Stylesheet

Process the source node and its children with `transformNode()` using the supplied XSLT stylesheet to return the resulting transformation in string.

```
output = xmldoc.transformNode( xsltstyle );
```

Writing Output to Page

- Using **document.write()**

  Place `document.write(output)` in `<SCRIPT>` block at the exact spot on the page where you want the output to appear.

- Using **outerHTML** or **innerHTML**

  Create a "token" element, <DIV ID="someToken">, whose `outerHTML` you'll set to the output generated by the XSL transformation.
  Note: `outerHTML` replaces the <DIV> tag while `innerHTML` inserts the ouput inside it.

  ```
  someToken.innerHTML = output;
  ```

Document Object Model (DOM)

Interfaces	Uses
Document Object Module (DOM)	• XML to XML • non-XML to XML • XML to non-XML

W3C definition:

- The Document Object Model is a platform- and language-neutral interface that will allow programs and scripts to dynamically access and update the content, structure and style of documents. The document can be further processed and the results of that processing can be incorporated back into the presented page.

W3C DOM

- DOM Level 1
- DOM Level 2
- DOM Level 3

Some Usefull DOM Properties

- **nodeName**

  Contains the qualified name of the element, attribute, or entity reference, or a fixed string for other node types.
  (read-only)

- **nodeValue**

  Contains the text content associated with the node.

(read/write)

- **firstChild**

  Contains the first child of the node.
  (read-only)

- **lastChild**

  Contains the last child of the node.
  (read-only)

- **previousSibling**

  Contains the previous sibling of the node in the parent's child list.
  (read-only)

- **nextSibling**

  Contains the next sibling of the node in the parent's child list.
  (read-only)

- **length**

  Indicates the number of items in the collection.
  (read-only)

Some Usefull DOM Methods

- **getElementsByTagName("tagName")**

  Returns a collection of elements that have the specified name in the order in which they would be encountered in a preorder traversal of the document tree. In a preorder traversal, the parent root node is visited first and each child node from left to right is then traversed. For example, to retrieve the 49th <book> in a <catalog>,

  ```
  book_node = xmldoc.getElementsByTagName("book")[48];
  ```

- **setAttribute(name, value)**

  Sets the value of the named attribute.

  ```
  book_node.setAttribute( 'isbn', '0735712867' );
  ```

- **getNamedItem(name)**

  Retrieves the attribute with the specified name.

  ```
  attribute = book_node.attributes.getNamedItem("isbn");
  ```

- getNodeType
- getNodeName
- getNodeValue and setNodeValue (value)
- getParentNode and setParentNode (parentNode)
- getChildNodes
- getFirstChild
- getLastChild

- getPreviousSibling
- getNextSibling
- getAttributes
- getOwnerDocument
- insertBefore (newChild, refChild)
- replaceChild (newChild, oldChild)
- removeChild (oldChild)
- appendChild (newChild)
- hasChildNodes
- cloneNode (deep)
- normalize
- removeAttribute (name)

W3C DOM in Microsoft Internet Explorer

Programming DOM in JScript using MSXML Tutorial (click on "source code" to view examples)

For processing XML in .NET see From MSXML3.0 to .NET XML Classes: A Quick Guide

Resources:

- W3C: Document Object Model (DOM)
- W3C: Document Object Model (DOM) Level 1 Specification
- W3C: Document Object Model (DOM) Level 2 Core Specification
- W3C: Document Object Model (DOM) Level 3 Core Specification
- ZVON: DOM1 Reference
- ZVON: DOM2 Reference
- W3C DOM Compatibility in Browsers
- W3C DOM in Microsoft Internet Explorer
- Microsoft XML Core Services (MSXML)
- DOM in Mozilla
- JavaScript Tutorial
- W3Schools: JavaScript Tutorial
- W3Schools: XML DOM Tutorial

Hands-on Assignment:

- **Assignment 3** due **Week 6**