UFCEKG-20-2 : Data, Schemas and Applications

The Document Object Model (DOM) & Asynchronous Javascript And XML (AJAX) : an introduction
DOM : What is it?

An **object-based, language-neutral, application programming interface (API)** for XML and HTML documents

- allows programs and scripts to build documents, navigate their structure, add, modify or delete elements and content
- provides a foundation for developing *querying, filtering, transformation, rendering* etc.

applications on top of DOM implementations

In contrast to “**Serial Access XML**” (sax) a good way to think of the DOM is as “**Directly Obtainable in Memory**” (dom) objects representing the nodes, attributes and content of documents
Based on O-O concepts:

- **methods** (to access or change object’s state)
- **interfaces** (declaration of a set of methods)
- **objects** (encapsulation of data and methods)

Roughly similar to the XSLT/XPath data model

≈ a parse tree

Tree-like structure implied by the abstract relationships defined by the programming interfaces;

Does not necessarily reflect data structures used by an implementation (but probably does)
DOM : What is it? (3)

- Language-independence:
  DOM interfaces are defined using OMG Interface Definition Language (IDL; Defined in Corba Specification)

- Language bindings (implementations of DOM interfaces) defined in the Recommendation for Java and ECMAScript (standardised JavaScript)
“Dynamic HTML" is a term used by some vendors to describe the combination of HTML, style sheets and scripts that allows documents to be animated. The W3C has received several submissions from member companies on the way in which the object model of HTML documents should be exposed to scripts. These submissions do not propose any new HTML tags or style sheet technology. The W3C DOM Activity is working hard to make sure interoperable and scripting-language neutral solutions are agreed upon.
The DOM specification is separated into 3 different parts / levels:

**Core** DOM - standard model for any structured document

**XML** DOM - standard model for XML documents

**HTML** DOM - standard model for HTML documents

The DOM defines the *objects and properties* of all document elements, and the *methods* (interface) to access them.
A simple **DOM (html)** tree:
Preparing pages for DOM

• All pages *must* be well-formed XHTML/HTML5 documents.
• All pages *must* include a valid DOCTYPE.
• All pages *must* include all text inside valid XHTML/HTML5 elements.
Referencing Objects

• Objects can be referenced
  – by their id or name (this is the easiest way, but you need to make sure a name is unique in the hierarchy)
  – by their numerical position in the hierarchy, by walking the array that contains them
  – by their relation to parent, child, or sibling (parentNode, previousSibling, nextSibling, firstChild, lastChild or the childNodes array)
What is AJAX

○ Asynchronous Javascript And XML
  – allows the updating of a web page without doing a page reload
    • creates much nicer user experience

○ AJAX is not really a technology by itself
  – combination of Javascript, XML and some server-side scripting to create the XML
    • server-side scripting could be done in PHP, .NET, Java Servlet or Java Server Page (JSP)
Client/server interaction (synchronous)
Ajax web application model (asynchronous)

- Client
  - Browser UI
  - User activity
    - Input
    - Display
  - Ajax engine
    - Client-side processing
    - Data transmission
    - Server-side processing

- Time

- Server
general technique (simplified model)

Web Page

- info parsed from XML / text / json and used to update DOM by Javascript

Server-side Script

- script run, XML created
- XML document returned
- requests server-side script to be run
general technique
(showing XMLHttpRequest)
sending a request for a URL

- xmlHttpRequest Object
  - mozilla
    - tmpXmlHttpRequest = new XMLHttpRequest();
  - IE
    - tmpXmlHttpRequest = new ActiveXObject("Microsoft.XMLHTTP");

- create the URL

- tell the browser the name of the function to handle the response

- send the url to the server
html + javascript (browser)

3 functions (javascript):
function createRequestObject() {
    // initialize request object
}
function makeGetRequest(wordId) {
    // initiate the request
}
function processResponse() {
    // process server response and update dom
}

html <div> section updated with returned text
processResponse

• when the document is received by the browser control is transferred to where ever we told it to
  – xmlHttp.onreadystatechange=processResponse
  – in this case the function named processResponse
function processResponse() {
    // check if the server responded
    if(http.readyState == 4) {

        // read and assign the response from the server
        var response = http.responseText;

        // do additional parsing of the response, if needed

        // assign the response to the contents of the <div> tag.
        document.getElementById('description').innerHTML = response;

        // If the server returned an error,
        // message would be shown within the div tag!!.
        // So it may be worth doing some basic error checking
        // before setting the contents of the <div>
    }
}
XMLHttpRequest Object

• Methods:
  – abort()  - stop the current request
  – getAllResponseHeaders  - Returns complete set of headers (labels and values) as a string
  – getResponseHeader(:headerLabel”)  – returns the string value of the requested header field
  – open(“method”,”URL”)  sets a pending request
  – send(content)  – transmits the request
  – setRequestHeader(“label”,”value”)  – sets label/value in the header
(continued)

• Properties
  – onreadystatechange - event handler to use
  – readyState (0-uninitialized, 1-loading, 2-loaded, 3-
    interactive, 4- complete)
  – responseText – string version of the data returned
  – responseXML – DOM compatible document object
    returned by server
  – status – http response header code (200 – good)
  – statusText – string message of status code
server-side script

- creates a “well formed XML document”
- sets the content type to text/xml
- can be written in any language
  - PHP
  - ASP
  - .NET
  - Java
  - JSP
sample PHP script

$id = $_GET['id'];

switch ($id) {
    case 1:
        echo 'Astraphobia, also known as .....';
        break;
    case 2:
        echo 'Arithmophobia is the fear of numbers. .... ';
        break;
    case 3:
        echo 'Ophidiophobia or ophiophobia is a ..... ';
        break;
}
Pros & Cons

• Pros
  - Interactivity
  - Portability

• Cons
  - Usability criticisms:
    - Back button: users generally expect that the back button will undo the last state change.
    - Bookmarking: users might also expect to be able to bookmark pages retrieve them as they would a static html page.
  - Response time concerns
    - Network latency may lead to interface delays that user may accept from a web page but not an ‘application’
  - JavaScript?
    - AJAX requires users to have JavaScript enabled in their browsers. This applies to all browsers that support AJAX except for Microsoft
Popular Javascript Frameworks supporting AJAX

• Jquery
  - [http://jquery.com/](http://jquery.com/)
  - open source
• Prototype
  – open source
• mootools
  – open source
• extJS
  - open source / commercial