**Introduction to Ajax**

**(Asynchronous JavaScript and XML)**

**ITC 250/CPET 499 Web Systems**

Oct. 16, 2014

References

* XMLHttpRequest Level 1, W3C Working Draft 30 January 2014, <http://www.w3.org/TR/XMLHttpRequest/>
* XMLHttpRequest.js object implementation, <https://github.com/ilinsky/xmlhttprequest>
* Xmlhttprequest, <http://code.google.com/p/xmlhttprequest/>
* Using XMLHttpRequest, Mozilla Developer Network, <https://developer.mozilla.org/en-US/docs/Web/API/XMLHttpRequest/Using_XMLHttpRequest>
* XMLHttpRequest object, [http://msdn.microsoft.com/en-us/library/ie/ms535874(v=vs.85).aspx](http://msdn.microsoft.com/en-us/library/ie/ms535874%28v%3Dvs.85%29.aspx)
* Chapter 16 AJAX-Enabled Rich Internet Applications with XML and JSON, Internet & World Wide Web: How to Program, 5th edition, by P. Deitel, H. Deitel, and A. Deitel, published by Pearson, <http://www.etcs.ipfw.edu/~lin/CPET499-ITC250/Fall2013-ITC-250-CPET-499/cpet499-ITC250-WebSystems-Lectures-F2013.html>
* Chapter 10 Intro to Ajax, Programming the World Wide Web, 8th Ed, by Robert W. Sebesta, published by Pearson.
* jQuery Ajax, <http://api.jquery.com/jquery.ajax/>
* Ajax with Dojo, <http://dojotoolkit.org/documentation/tutorials/1.6/ajax/>
* Prototype JavaScript Framework, <http://prototypejs.org/>
* OWAASP Ajax Security Guidelines, <https://www.owasp.org/index.php/OWASP_AJAX_Security_Guidelines>
* Testing for AJAX Vulnerabilities, [https://www.owasp.org/index.php/Testing\_for\_AJAX\_Vulnerabilities\_(OWASP-AJ-001)](https://www.owasp.org/index.php/Testing_for_AJAX_Vulnerabilities_%28OWASP-AJ-001%29)

AJAX References

* W3C Web Accessibility Initiative, WAI-ARIA (Accessible Rich Internet Application) overview, <http://www.w3.org/WAI/intro/aria.php>
* W3C XMLHttpRequest Level 2, Draft 17, January 2012, <http://www.w3.org/TR/XMLHttpRequest/>
* W3C JavaScript Web APIs (Dynamic HTML => AJAX), <http://www.w3.org/standards/webdesign/script.html>
* OpenAjax Alliance, <http://www.openajax.org/index.php>
* W3C Mobile Web Initiative, OpenAjax Alliance, <http://www.w3.org/2007/06/mobile-ajax/>
* Intro to Mobile Ajax for Developers, <http://www.openajax.org/whitepapers/Introduction%20to%20Mobile%20Ajax%20for%20Developers.php>

AJAX browser support References

* Bing Maps AJAX Control 7.0 Supported Browsers, <http://msdn.microsoft.com/en-us/library/gg427618.aspx>
* IE XMLHttpRequest object (Internet Explorer), [http://msdn.microsoft.com/en-us/library/ie/ms535874(v=vs.85).aspx](http://msdn.microsoft.com/en-us/library/ie/ms535874%28v%3Dvs.85%29.aspx)
* Mozilla XMLHTTPRequest, <https://developer.mozilla.org/en-US/docs/DOM/XMLHttpRequest>
* Opera, Intro to XMLHttpRequest Level 2, <http://dev.opera.com/articles/view/xhr2/>
* Chrome, Cross-Origin XMLHttpRequest, <http://developer.chrome.com/extensions/xhr.html>
* Google, Asynchronous XMLHttpRequests with Xhrlo, <https://developers.google.com/closure/library/docs/xhrio>

Ajax (Asynchronous JavaScript and XML)

* Use JavaScript and XML fro creating dynamic web applications
* Use client-side scripting to make web applications more responsive
* The Ajax component that manages interaction with the server is usually implemented with JavaScript’s XMLHttpRequest object.
* The server side processing can be implemented using any server-side technology, such as PHP, ASP.NET and JavaSever Faces

Traditional Web Applications

* Request 1 (page 1 form) => Process request (server) => Generate response (server; page 2 form) => Page loading on client side
* Request 2 (page 2 form) => Process request (server) => Generate response (server; page 3 form) => Page reloading on client side

Ajax Web Applications (Rich Internet Applications)

* Request object (call back functions) => Process request 1 => Generate response (data, partial page update on the page; async response processing at client side) =>
* Request object (call back functions) => Process request 2 => Generate response (data, partial page update)

XMLHttpRequest Level 1, W3C Working Draft 30 January 2014, <http://www.w3.org/TR/XMLHttpRequest/>

The XMLHttpRequest object is an API for fetching resources.

* It supports any text based format, including XML.
* It can be used to make request over both HTTP and HTTPs.

Some simple code to do something with data from an XML document fetched over the network:

**function processData(data) {**

 **// taking care of data**

**}**

**function handler() {**

 **if(this.readyState == this.DONE) {**

 **if(this.status == 200 &&**

 **this.responseXML != null &&**

 **this.responseXML.getElementById('test').textContent) {**

 **// success!**

 **processData(this.responseXML.getElementById('test').textContent);**

 **return;**

 **}**

 **// something went wrong**

 **processData(null);**

 **}**

**}**

**var client = new XMLHttpRequest();**

**client.onreadystatechange = handler;**

**client.open("GET", "unicorn.xml");**

 **client.send();**

If you just want to log a message to the server:

**function log(message) {**

 **var client = new XMLHttpRequest();**

 **client.open("POST", "/log");**

 **client.setRequestHeader("Content-Type", "text/plain;charset=UTF-8");**

 **client.send(message);**

**}**

Or if you want to check the status of a document on the server:

**function fetchStatus(address) {**

 **var client = new XMLHttpRequest();**

 **client.onreadystatechange = function() {**

 **// in case of network errors this might not give reliable results**

 **if(this.readyState == this.DONE)**

 **returnStatus(this.status);**

 **}**

 **client.open("HEAD", address);**

 client.send();

}

Interface XMLHttpRequest

* Constructor
* Garbage Collection
* Event Handlers
	+ onloadstart
	+ onprogress
	+ onabort
	+ onerror
	+ onload
	+ ontimeout
	+ onloadend
	+ onreadystatechange
* States
	+ UNSENT (numeric value 0)
	+ OPENED (numeric value 1)
	+ HEADERS\_RECEIVED (numeric value 2)
	+ LOADING (numeric value 3)
	+ DONE (numeric value 4)
* Request – methods and attributes
	+ open()
	+ setRequestHeader()
	+ send()
	+ Infrastructure for the send() method
	+ abort()
	+ timeout
	+ withCredentilas
	+ upload
* Response – methods and attributes
	+ getResponseHeader()
	+ getAllResponseHeaders()
	+ overrideMimeType()
	+ status
	+ statusText
	+ responseType
	+ response
	+ responseText
	+ responseXML
* Event Summary
	+ readystatechange
	+ readyState
	+ responseText
	+ responseXML
	+ status
	+ statusText

Interface FormData