AJAX Workshop

Karen A. Coombs
University of Houston Libraries
Jason A. Clark
Montana State University Libraries

What is AJAX?

AJAX stands for Asynchronous Javascript and XML. However,not all AJAX apps involve XML. AJAX uses a combination of technologies including:
XHTML, CSS, DOM; XML, XSLT, XMLHttp, JavaScript; and a server scripting language such as PHP or Coldfusion. AJAX was created because web developers needed a method for building more responsive and interactive applications.

AJAX Components

XHTML and CSS

Ajax applies these familiar Web standards for styling the look and feel of a page and to markup those areas on a page that will be targeted for data updates.

DOM (document object model)

Ajax uses the DOM to manipulate dynamic page views for data and to walkthrough documents to “cherrypick” data. The DOM enables certain pieces of an Ajax page to be transformed and updated with data.

XML, JSON (Javascript Object Notation), HTML, or plain text

Ajax can use any of these standards to provide structure to the data it passes to and from a page.

XMLHttpRequest object

The heavy lifter for Ajax: It’s a javascript object embedded in most modern browsers that sets up data request/response pipelines between client and server.

Javascript

Lightweight programming language that Ajax uses for instructions to bind all of the components together.

Why Use AJAX?

- You want to make your applications more interactive
- You want to incorporate data from external Web Services
- You don’t want your users to have to download a plugin

Client vs. Server Scripting

- Client scripting
  - Web browser does all the work
- Server Scripting
  - Web server does all the work
- AJAX leverages both client and server side scripting

How AJAX Works

AJAX Web Interaction

- What you don't see
- Data reload happens in the background
- JavaScript queries the server to get the proper data without you knowing it
- Page updates without a screen "reload"

Potential Problems

- Javascript MUST be enabled
- Back button doesn't always work
- Pages can be difficult to bookmark
- Search engines may not be able to index all portions of an AJAX site
- Cross browser differences in how XML is dealt with

Some AJAX examples

- Gmail
- Flickr
- Rojo
- Google Suggest
- Tada Lists

Basic AJAX Components

- Server-side Component
  - Communicates with the database, or web service
  - Can be written in any server-side language (PHP, ASP, Coldfusion, etc)
- Client-side Component
  - Written in Javascript, often uses XMLHttp
  - Accesses the server side page in the background

Hidden Frame Method

- Communication with server takes place in a frame that user can't see
- Back and Forward buttons still work
- If something goes wrong user receives no notification

XMLHttp Method

- Code is cleaner and easier to read
- Able to determine if there is a failure
- No browser history, Back and Forward buttons break

Potential Uses for AJAX

- Error checking in forms
- AutoSuggest
- Drag and Drop objects functionality
- Move around on image or map so you can see different parts
- Preload content you want to show later
- Apply limits to search results and get new results quickly

AJAX for Libraries

- Browsing subject headings
- "Pre-displaying" indexes and databases categories
- Complex ILL or contact forms
- Federated Search
- OPAC and digital library interfaces

AJAX - Library Use Cases

- Phoenix Live OPAC - OCLC Research (http://phoenix.orhost.org)
- Federated Search - Curtin University of Technology Library (Perth, Western Australia) (http://library.curtin.edu.au/cgi-bin/search/search.cgi?query=&submit=Search)
- Guesstimate - Virginia Tech Libraries (http://addison.vt.edu)

AJAX – Sample Applications

- PageInsert - WorldCat Form (http://localhost/ajax/)

- BrowseSearch - LOC Subject Headings (http://localhost/ajax/browseloc.php)

Code Sample #1: WorldCat Form
WorldCat XML file to provide content

```
<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
<content>
<header>What is Open WorldCat?</header>
<description>The Open WorldCat program makes records of library-owned
  materials in OCLC's WorldCat database available to Web users on
  popular Internet search, bibliographic and bookselling sites. Links
  to content in library collections—books, serials, digital images and
  many other formats—appear alongside links to traditional Web
  content.</description>
<sourceDomain>worldcatlibraries.org</sourceDomain>
<sourceUrl>http://worldcatlibraries.org/wcpa/isbn/0471777781</sourceUrl
  >
</content>
```

Explanation

- Our source file
- Various and sundry factoids about WorldCat, some associated urls
- header and description element to populate the heading and description of the content
- sourceDomain will give an action value to our WorldCat search form
- sourceUrl element will provide a link to an Open Worldcat record

Code Sample #2: WorldCat Form
Web page for user interface and display

```
…
<div id="container">
<div id="main"><a name="mainContent"></a>
<h1>Find it in a Library.  Use Open WorldCat.</h1>
<p><a onclick="createRequest('xml/worldcat.xml');" href="#show">+ Learn
  more about Open Worldcat</a></p>
<div id="content"></div>
</div>
<!-- end main div -->
</div>
<!-- end container div -->
…
```

Explanation

- XHTML form that gives action to our script
- Notice the javascript “onclick” event handler on <p> tag
- will be populated with script messages OR new html tags received via our Ajax requests

Code Sample #3: WorldCat Form
Using the XMLHttpRequest Object

```
//creates browser specific request using XmlHttpRequest Object
function createRequest(url)
{
  if(window.XMLHttpRequest)
  {
      request = new XMLHttpRequest();
  }
  else if(window.ActiveXObject)
  {
      request = new ActiveXObject("MSXML2.XMLHTTP");
  }
  else
  {
      alert("Please upgrade to a newer browser to use the full
  functionality of our site.");
  }

  makeRequest(url);
}

//sends request using GET HTTP method to grab external data
function makeRequest(url)
{
  request.onreadystatechange = parseData;
  request.open("GET", url, true);
  request.send(null);
}
```

Explanation

- First part of our javascript
- Creates the XMLHttpRequest
- Using the if and else statements to check for Web browsers' different implementations of XMLHttpRequest
- Ends with makeRequest function

Code Sample #4: WorldCat Form

Communicating the status of our request

```
//checks state of HTTP request and gives brief status note to user
function communicateStatus(obj)
{
  if(obj.readyState == 0)
  { document.getElementById('content').innerHTML = "Sending
  Request..."; }
  if(obj.readyState == 1)
  { document.getElementById('content').innerHTML = "Loading
  Response..."; }
  if(obj.readyState == 2)
  { document.getElementById('content').innerHTML = "Response
  Loaded..."; }
  if(obj.readyState == 3)
  { document.getElementById('content').innerHTML = "Response
  Ready..."; }
  if(obj.readyState == 4)
  {
      if(obj.status == 200)
      {
         return true;
      }
      else if(obj.status == 404)
      {
         // Add a custom message or redirect the user to another page
         document.getElementById('content').innerHTML = "File not
  found";
      }
      else
      {
         document.getElementById('content').innerHTML = "There was a
  problem retrieving the XML.";
      }
  }
}
```

Explanation

- Next part of our javascript
- Displays different messages to the user based on the status of the request on the server
- uses the “obj” variable which was created earlier when we called the XMLHttpRequest
- First peek at Document Object Model (DOM) in action

Code Sample #5: WorldCat Form

Using the DOM (Document Object Model)

```
//loads data from external file into page, breaks out variables from
  sections of file, and populates html with specific variable values
function parseData()
{
  if(communicateStatus(request))
  {
      //declare format of the data to be parsed and retrieved
      var response = request.responseXML.documentElement;
      var header =
  response.getElementsByTagName('header')[0].firstChild.data;
      var description =
  response.getElementsByTagName('description')[0].firstChild.data;
      var sourceDomain =
  response.getElementsByTagName('sourceDomain')[0].firstChild.data;
      var sourceUrl =
  response.getElementsByTagName('sourceUrl')[0].firstChild.data;
      document.getElementById('content').innerHTML = "<h2>" + header +
  "</h2>\n"
                                          + "<p>" + description +
  "</p>\n"
                                          + "<form method=\"get\"
  action=\"http://www.google.com/search\">\n"
                                          + "<fieldset>\n"
                                          + "<label>Search Open
  WorldCat:</label>\n"
                                          + "<input type=\"hidden\"
  name=\"as_sitesearch\" value='" + sourceDomain + "'>\n"
                                          + "<input type=\"text\"
  name=\"q\" size=\"40\" maxlength=\"255\" value=\"\">\n"
                                          + "<input class=\"submit\"
  type=\"submit\" name=\"sa\" value=\"Find Books\">\n"
                                          + "</fieldset>\n"
                                          + "</form>\n"
                                          + "<p><a href='" + sourceUrl +
  "'>View a sample Open WorldCat record</a></p>\n";
  }
}
```

Explanation

- Last part of our javascript
- Applies DOM to give us a standard means of modeling the structure of XHTML or XML documents
- DOM functions like “getElementsByTagName”
- Grab data and push it into prescribed sections of our XHTML page

Code Sample #6: WorldCat Form

CSS (Cascading Style Sheets)

```
…
/* =container
---------------------------------------------- */
div#container {width:65em;margin:0 auto;background:#fff;}

/* =main
---------------------------------------------- */
div#main {width:63em;margin:0 auto;padding:1em .5em 2em .5em;}

/* =content
---------------------------------------------- */
div#content {width:95%;margin:0 auto;}
#content p.warn {color:red;}

/* =forms
---------------------------------------------- */
form {padding:10px;border-top:1px solid #ccc;border-right:2px solid
  #ccc;border-bottom:2px solid #ccc;border-left:1px solid
  #ccc;background-color:#F2F2F2;}
fieldset {border:none;}
label {font-size:1.2em;color:#2b4268;vertical-
  align:middle;cursor:pointer;}
input, select, textarea {width:25em;font:1.0em verdana,arial,sans-
  serif;padding:3px;margin:3px;border:1px solid gray;border-color:#AAA
  #DDD #DDD #AAA;vertical-align:middle;}
input:focus {border:1px #000 solid;}
input.submit {width:10em;font-size:.90em;color:#2b4268;}
…
```

Explanation

- Part of our CSS file
- Means of passing style rules for different pieces of the Web page
- tags are given specific, relative widths, <form> tags are styled with attractive borders

Code Sample #1: LOC Subject Headings
Web page for user interface and display

```
<div id="main"><a name="mainContent"></a>
<h2 class="mainHeading">CIL 2006 :: Example: Library of Congress
BrowseSearch</h2>
<form id="searchbox" action="browseSearch.php" method="post">
  <p><label
for="query"><strong>BrowseSearch:</strong></label> <input
type="text" name="query" autocomplete="off" id="query"
onKeyUp="preSearch()" />
   <img id="searchStatus" alt="searching..."
src="./meta/img/spinner.gif" /></p>
</form>
<div id="result"> </div>
</div>
```

Explanation

- XHTML form that gives action to our script
- Note the javascript “onKeyUp” event handler on <input> tag
- <input> also given “name” and “id”
- will be populated with script messages OR new html tags received via our Ajax requests

Code Sample #2: LOC Subject Headings
Using javascript to “presearch” database

```
function preSearch() {
    //Put the form data into a variable
   var theQuery = document.getElementById('query').value;

    //If the form data is *not* blank, query the DB and return the
results
   if(theQuery !== "") {
        //If search pauses when fetching, change the content of the
"result" DIV to "Searching..."
         document.getElementById('result').innerHTML =
"Searching...";

        //This sets a variable with the URL (and query strings) to
our PHP script
         var url = 'browseSearch.php?q=' + theQuery;
        //Open the URL above "asynchronously" (that's what the
"true" is for) using the GET method
         xmlhttp.open('GET', url, true);
        //Check that the PHP script has finished sending us the
result
         xmlhttp.onreadystatechange = function() {
               if(xmlhttp.readyState == 4 && xmlhttp.status == 200)
{
                //Replace the content of the "result" DIV with the
result returned by the PHP script
                     document.getElementById('result').innerHTML =
xmlhttp.responseText + ' ';
```

Explanation

- Piece of javascript that creates instant search
- Talks to server-side PHP script - browseSearch.php
- Uses DOM to populate with search results

Code Sample #3: LOC Subject Headings
PHP search loc database script

```
<?php
//declare variables to be used in query and display
$keywords = $_GET['query'];
$link = "<p><a href=\"browseSearch.php\">Library of Congress
LiveSearch</a></p>";
...
// bring database parameters and functions onto page
...
//form sql statement
$query = "SELECT subject_id, label, callno FROM subject WHERE label
LIKE '%$keywords%' ORDER BY callno ASC";
//store sql result as an array
$result = mysql_query($query) or die('<p class=\"warn\">Error
retrieving subjects from loc database!<br />'.
                'Error: ' . mysql_error() . '</p>');
//create message if no rows match search terms
…
//format sql result for display
   while($record = mysql_fetch_object($result))
         {
         echo '<dl><dt><strong>'.stripslashes($record-
>label).'</strong></dt>';
         echo '<dd>Call Number Range: '.stripslashes($record-
>callno).'</dd>';
         echo '<dd><a
href="http://www.lib.montana.edu/help/locationguide.html">Find Call
Number on Library Floor Map</a></dd></dl>';
         echo '<hr size="1" />';
         }
         echo $link;
?>
```

Explanation

- Piece of PHP script that searches loc database
- Basic SQL SELECT statement
- Uses <dl> to format search results

AJAX - Further References

Articles

- Ajax: A New Approach to Web Applications by Jesse James Garrett
  http://www.adaptivepath.com/publications/essays/archives/000385.php
- Ajax gives software a fresh look (from CNET News)
  http://beta.news.com.com/Ajax+gives+software+a+fresh+look/2100-1007_3-5886709.html?
- Weighing the Alternatives (from ajax info)
  http://www.ajaxinfo.com/default~viewart~8.htm

Resources

- XMLHttpRequest & Ajax Based Applications (from Fiftyfoureleven.com)
  http://www.fiftyfoureleven.com/resources/programming/xmlhttprequest/
- *Foundations of Ajax* by Ryan Asleson, Nathaniel T. Schutta
  ISBN: 1590595823 http://www.worldcatlibraries.org/wcpa/isbn/1590595823

Tutorials

- Getting Started with AJAX (from A List Apart)
  http://www.alistapart.com/articles/gettingstartedwithajax
- AJAX:Getting Started (from Mozilla Developer Center)
  http://developer.mozilla.org/en/docs/AJAX:Getting_Started
- Dynamic HTML and XML: The XMLHTTPRequest Object (from Apple Developer Connection)
  http://developer.apple.com/internet/webcontent/xmlhttpreq.html
- Mastering Ajax, Part 1: Introduction to Ajax (from IBM developerWorks)
  http://www-128.ibm.com/developerworks/web/library/wa-ajaxintro1.html?ca=dgr-wikiAJAXinto1

Contact Information

Karen Coombs
University of Houston Libraries
Web Services Librarian
kacoombs@uh.edu
http://librarywebchic.net/
713-743-3713

Jason A. Clark
Montana State University Libraries
Digital Initiatives Librarian
jaclark@montana.edu
www.jasonclark.info
406-994-6801