AJAX for Libraries: Code Samples, Explanations, and Downloads

Jason A. Clark
Head of Digital Access and Web Services
Montana State University Libraries
jaclark@montana.edu

Karen A. Coombs
Head of Web Services
University of Houston Libraries
kacoombs@uh.edu

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://news.cnet.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

- Google AJAX Libraries API
  http://code.google.com/apis/ajaxlibs/

Tutorials

- W3 Schools - AJAX Tutorial
  http://www.w3schools.com/Ajax/

- 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

AJAX – Sample Applications

People @ the Library (HTML and Feedback)
http://www.lib.montana.edu/~jason/files/ajax/show/

BrowseSearch Library of Congress Subject Headings
http://www.lib.montana.edu/~jason/files/ajax/browsesearch/

Javascript AJAX Code Libraries Examples

- Hello User Example (Prototype JS Library)
- WorldCat Search API (Prototype JS Library)
- WorldCat Search API (MooTools JS Library)

Code Sample #1: People @ the Library - xHTML file to provide content

```
<h2><a href="mailto:kacoombs@uh.edu">Karen Coombs</a></h2>
<p>Head of Web Services, University of Houston Libraries</p>
<a href="http://librarywebchic.net/wordpress/">http://librarywebchic.net/wordpress/</a>
```

Code Sample #1: People @ the Library - Explanation

- One of our data files
- Various and sundry factoids about person, some associated urls
- Header and description element to populate the heading and description of the content
- Can pass any xHTML tags or markup - <form>, <ul>, <table>

Code Sample #2: People @ the Library - Web page for user interface and display

```
…
<div id="container">
<div id="main"><a name="mainContent"></a>
```

```
    <h1>People @ Your Library</h1>
    <p class="control"><a href="./" class="refresh">Reset the page</a></p>
    <ul id="people">
        <li id="first"><a href="?person=karen">Karen</a></li>
        <li><a href="?person=jason">Jason</a></li>
        <li><a href="?person=amy">Amy</a></li>
    </ul>
    <div id="details">
        <?php include "people.php"; ?>
    </div>
</div>
<!-- end main div -->
</div>
```

Code Sample #2: People @ the Library - Explanation

- xHTML that provides interface and gives action to our script
- Notice the query string value (?person=) on <a> tag
- <div id="details"> will be populated with script messages OR new xHTML tags received via our Ajax requests

Code Sample #3: People @ the Library - Using the XMLHttpRequest object

```
function getHTTPObject() {
  var xhr = false;
  if (window.XMLHttpRequest) {
    xhr = new XMLHttpRequest();
  } else if (window.ActiveXObject) {
    try {
      xhr = new ActiveXObject("Msxml2.XMLHTTP");
    } catch(e) {
      try {
        xhr = new ActiveXObject("Microsoft.XMLHTTP");
      } catch(e) {
        xhr = false;
      }
    }
  }
  return xhr;
}

function grabFile(file) {
  var request = getHTTPObject();
  if (request) {
    displayLoading(document.getElementById("details"));
    request.onreadystatechange = function() {
      parseResponse(request);
    };
    request.open("GET", file, true);
    request.send(null);
```

```
    return true;
  } else {
    return false;
  }
}
```

Code Sample #3: People @ the Library - Explanation

- First part of our javascript
- "getHTTPObject" function creates the XMLHttpRequest object
- Using the if and else statements to check for Web browsers' different implementations of XMLHttpRequest
- "grabFile" function makes request, gives us peek of Document Object Model (DOM) in action using "getElementById" to select piece of page to update
- Relies on two separate functions - a feedback function ("displayLoading") and a load request function ("parseResponse")

Code Sample #4: People @ the Library - Showing feedback to user

```
function displayLoading(element) {
  while (element.hasChildNodes()) {
    element.removeChild(element.lastChild);
  }
  var image = document.createElement("img");
  image.setAttribute("src","img/loading.gif");
  image.setAttribute("alt","Loading...");
  element.appendChild(image);
}

function fadeUp(element,red,green,blue) {
  if (element.fade) {
    clearTimeout(element.fade);
  }
  element.style.backgroundColor = "rgb("+red+","+green+","+blue+")";
  if (red == 255 && green == 255 && blue == 255) {
    return;
  }
  var newred = red + Math.ceil((255 - red)/10);
  var newgreen = green + Math.ceil((255 - green)/10);
  var newblue = blue + Math.ceil((255 - blue)/10);
  var repeat = function() {
    fadeUp(element,newred,newgreen,newblue)
  };
  element.fade = setTimeout(repeat,100);
}
```

Code Sample #4: People @ the Library - Explanation

- The two functions that show visual cues to the user after action
- "displayLoading" shows status messages and images to user
- "fadeUp" highlights where the page update is taking place

Code Sample #5: People @ the Library - Communicating status and loading the response

```
//checks state of HTTP request and gives brief status note to user
function parseResponse(request) {
  if (request.readyState == 4) {
    if (request.status == 200 || request.status == 304) {
      var details = document.getElementById("details");
      details.innerHTML = request.responseText;
      fadeUp(details,255,255,153);
    }
  }
}
```

Code Sample #5: People @ the Library - Explanation

- Next part of our javascript
- Displays different messages and cues to the user based on the status of the request on the server
- Uses "innerHTML" and "responseText" to target and write new data into <div id="details">
- Second peek at Document Object Model (DOM) in action using "getElementById"

Code Sample #6: People @ the Library - Set up client side scripting

```
window.onload = prepareLinks;

function prepareLinks() {
  if (!document.getElementById || !document.getElementsByTagName) {
    return;
  }
  if (!document.getElementById("people")) {
    return;
  }
  var list = document.getElementById("people");
  var links = list.getElementsByTagName("a");
  for (var i=0; i<links.length; i++) {
    links[i].onclick = function() {
      var query = this.getAttribute("href").split("?")[1];
      var url = "people.php?"+query;
      return !grabFile(url);
    };
  }
}
```

Code Sample #6: People @ the Library - Explanation

- Last part of our javascript - "hijacks" server side scripting

- Earmark and traverse xHTML data elements - <a> and <ul id="people">
- Rewrite it to be used and available for javascript functions such as "onclick"
- More DOM functions like "getElementsByTagName"

Code Sample #7: People @ the Library - 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;}
/* =people
---------------------------------------------- */
ul#people {display:inline;}
ul#people li {margin-left:0;padding-left:30px;border:none;list-style:none;display:inline;}
ul#people li#first {margin-left:0;padding-left:0;border:none;}
/* =details
---------------------------------------------- */
div#details {margin-top:30px;}
```

Code Sample #7: People @ the Library - Explanation

- Part of our CSS file
- Means of passing style rules for different pieces of the Web page
- <div> tags are given specific, relative widths, <ul> and <li> tags are styled to be listed inline

Code Sample #1: BrowseSearch 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>
```

Code Sample #1: BrowseSearch LOC Subject Headings -

Explanation

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

Code Sample #2: BrowseSearch 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 + ' ';
         } else {
            //If the PHP script fails to send a response, or kicks an error, display a simple
user-friendly notification
            document.getElementById('result').innerHTML = 'Error: preSearch Failed!';
         }
      };
      xmlhttp.send(null);
   }
}
```

Code Sample #2: BrowseSearch LOC Subject Headings - Explanation

- Piece of javascript that creates instant search
- Talks to server-side PHP script - browseSearch.php
- Uses DOM to populate <div id="result"> with search results

Code Sample #3: BrowseSearch LOC Subject Headings - Using

PHP and MySQL to search database

```
<?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;

?>
```

Code Sample #3: BrowseSearch LOC Subject Headings - Explanation

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

Code Sample: "Hello User" - Prototype Example 1

The HTML and Javascript

```
<html>
<head>
   <title>Test Form</title>
   <script type="text/javascript" src="prototype.js"></script>
   <script type="text/javascript">
      Event.observe(window, 'load', init, false);
```

```
        function init(){
            $('greeting-submit').style.display = 'none';
            Event.observe('greeting-name', 'keyup', greet, false);
        }



        function greet(){
            var url = 'prototype_example.php';
            var pars = 'greeting-name='+escape($F('greeting-name'))+'&ajax=true';
            var target = 'greeting';
            var myAjax = new Ajax.Updater(target, url, {    method: 'get',    parameters:
pars});
        }
    </script>
</head>
<body>
    <form method="get" action="prototype_example.php" id="greeting-form">

    <div>
        <label for="greeting-name">Enter your name:</label>
        <input id="greeting-name" name="greeting-name" type="text" />
        <input id="greeting-submit" name="greeting-submit" type="submit" value="Greet
me!" />
    </div>
    <div id="greeting"></div>
    </form>

 </body>
 </html>
```

The HTML and Javascript explanation

Form to submit name data
Javascript takes name data as it is keyed in (keyup) and uses the PHP file to write Hello plus name
<div id="greeting"> is where response goes
AJAX variable set to true with Javascript so that HTML snippet is returned

PHP

```
<?php
     $the_name = htmlspecialchars($_GET['greeting-name']);

     if (! isset($_REQUEST['ajax'])) {
         echo '<html>';
        echo '<head>';
        echo '<title>Hello' . $the_name . '</title>';
        echo '</head>';
        echo '<body>';

     }
     echo "<p>Hello, $the_name!</p>";
```

```
    if (! isset($_REQUEST['ajax'])) {
        echo '</body>';
       echo '</html>';
    }
?>
```

PHP Explanation

Takes form data submitted and writes Hello plus the name.
If not an ajax request a full HTML page is written, rather than HTML snippet

Code Sample: WorldCat Search API - Prototype Example 2

The HTML and Javascript A

```
<html>
<head>
   <title>Search WorldCat</title>
   <script type="text/javascript" src="prototype.js"></script>
   <script type="text/javascript">
      function submitSearch(){
         var url = 'prototype_example2.php';
         var pars = 'search='+escape($F('search'))+'&ajax=true';
         var target = 'results';
         new Ajax.Updater(target, url, {method: 'get',    parameters: pars, evalScripts: 
true});
         return false;
      }
   </script>
</head>
<body>
   <form method="get" action="prototype_example2.php" id="search-form" 
onSubmit="return submitSearch();">

   <div>

      <label for="search">Search:</label>
      <input id="search" name="search" type="text" />
      <input id="search-submit" name="search-submit" type="submit" value="Search"/>

   </div>
   <div id="results"></div>
   </form>

 </body>
 </html>
```

The HTML and Javascript explanation A

If Javascript enabled form onSubmit sends search via Javascript with a variable that tells PHP that the search is an AJAX one
return false prevents form from being submitted in traditional manner
Javascript passes search term and ajax hidden variable generated via Javascript to PHP in background and returns results. Results go into <div id=results>

The HTML and Javascript B

```
<html>
<head>
  <title>Search WorldCat</title>
  <script type="text/javascript" src="prototype.js"></script>
  <script type="text/javascript">
    document.observe('dom:loaded', function() {
      $('search-form').insert('<input type="hidden" name="ajax" value="true" />');

      $('search-form').observe('submit', function(e) {
        e.stop();

        this.request({
          onComplete: function(response){
            $('results').innerHTML = response.responseText;
          }
        });
      });

    });

  </script>
</head>
<body>
  <form method="get" action="prototype_example2.php" id="search-form">

  <div>
    <label for="search">Search:</label>
    <input id="search" name="search" type="text" />
    <input id="search-submit" name="search-submit" type="submit" value="Search"/>

  </div>
  <div id="results"></div>
  </form>

 </body>
```

The HTML and Javascript explanation B

If Javascript enabled form sends search via Javascript with a hidden field that tells PHP that the search is an AJAX one
e.stop(); prevents form from being submitted in traditional manner
Results go into <div id=results>
If javascript it off form is submitted in traditional matter and hidden field for ajax is not sent.

PHP

```
<?php
   $search = $_REQUEST['search'];
```

```
    $searchURL = 'http://worldcat.org/webservices/catalog/search/worldcat/opensearch?q='
. urlencode($_REQUEST['search']);

   // Format (Atom) WorldCat Service Level and wskey
   $searchURL .= '&format=atom&servicelevel=full&wskey=12345kac';
      $xml = simplexml_load_file($searchURL);

    if (! isset($_REQUEST['ajax'])) {
       echo '<html>';
       echo '<head>';
       echo '<title>WorldCat Search Results</title>';
       echo '</head>';
       echo '<body>';
    }
      $xml->registerXPathNamespace("opensearch", "http://a9.com/-/spec/opensearch/
1.1/");
      $xml->registerXPathNamespace("atom", "http://www.w3.org/2005/Atom");
      foreach($xml->xpath('//atom:entry') as $book ) {
         $field = simplexml_load_string($book->asXML());
         $title = $field->title;
         $author = $field->author->name;
         $link = $field->link['href'];
         $summary = $field->summary;

         echo '<p><a href="' . $link . '">' . $title . '</a> by ' . $author . '</p>';
         echo '<p>Summary: ' . $summary . '</p>';

      }
    if (! isset($_REQUEST['ajax'])) {
       echo '</body>';
       echo '</html>';
    }
?>
```

PHP Explanation

Form information is processed
Request sent to WorldCat Search API
Results returned from WorldCat Search API and formatted.
If form is being submitted in traditional manner (ie. ajax variable not set), then the results are generated in a full HTML page. Otherwise if ajax variable set then only an HTML snippet is created and passed back to HTML page

Code Sample: WorldCat Search API - MooTools Example

The HTML and Javascript

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN"
   "http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd">
<html xmlns="http://www.w3.org/1999/xhtml" xml:lang="en" lang="en">
<head>
   <meta http-equiv="Content-Type" content="text/html; charset=utf-8"/>
```

```
    <title>WorldCat Search</title>
    <script src="/~coombsk/mootools.js" type="text/javascript"></script>
    <script type="text/javascript">
        window.addEvent('domready', function() {

            var el = new Element('input', {
                'type': 'hidden',
                'name': 'ajax',
                'value': '1'
            }).inject($('search-form'));

            $('search-form').addEvent('submit', function(e) {
                e.stop();

                this.set('send', {
                    'onComplete': function(response) {
                        $('results').innerHTML = response;
                    }
                });

                this.send();
            });

        });
</script>
<body>
    <form method="get" action="mootools_example.php" id="search-form" name="search-
form">

        <label for="search">Search:</label>
        <input id="search" name="search" type="text" />
        <input id="search-submit" name="search-submit" type="submit" value="Search" />
    </form>
    <div id="results"></div>
</body>
</html>
```

The HTML and Javascript explanation
If Javascript enabled form sends search via Javascript with a hidden field (generated via Javascript) that tells PHP that the search is an AJAX one
e.stop(); prevents form from being submitted in traditional manner
Results go into <div id=results>
If javascript it off form is submitted in traditional matter and hidden field for ajax is not sent.

PHP

```
<?php
    $search = $_REQUEST['search'];

    $searchURL = 'http://worldcat.org/webservices/catalog/search/worldcat/opensearch?q='
. urlencode($_REQUEST['search']);
```

```
    // Format (Atom) WorldCat Service Level and wskey
    $searchURL .= '&format=atom&servicelevel=full&wskey=12345kac';
        $xml = simplexml_load_file($searchURL);

     if ($_REQUEST['static'] = 'Yes') {
         echo '<html>';
         echo '<head>';
         echo '<title>WorldCat Search Results</title>';
         echo '</head>';
         echo '<body>';
     }
        // Format (Atom) WorldCat Service Level and wskey
    $searchURL .= '&format=atom&servicelevel=full&wskey=12345kac';
        $xml = simplexml_load_file($searchURL);

        $xml->registerXPathNamespace("opensearch", "http://a9.com/-/spec/opensearch/
1.1/");
        $xml->registerXPathNamespace("atom", "http://www.w3.org/2005/Atom");
        foreach($xml->xpath('//atom:entry') as $book ) {
            $field = simplexml_load_string($book->asXML());
            $title = $field->title;
            $author = $field->author->name;
            $link = $field->link['href'];
            $summary = $field->summary;

            echo '<p><a href="' . $link . '">' . $title . '</a> by ' . $author . '</p>';
            echo '<p>Summary: ' . $summary . '</p>';

        }
     if ($_REQUEST['static'] = 'Yes') {
         echo '</body>';
         echo '</html>';
     }
?>
```

PHP Explanation

Form information is processed
Request sent to WorldCat Search API
Results returned from WorldCat Search API and formatted.
If form is being submitted in traditional manner, then the results are generated in a full HTML page

Additional Code Samples and Downloads

- People @ the Library (XML)
  http://www.lib.montana.edu/~jason/files/ajax/list/

- Yahoo! News Search @ the Library (JSON)
  http://www.lib.montana.edu/~jason/files/ajax/search/

- Contact the Library (Feedback and Validation) http://www.lib.montana.edu/~jason/files/ajax/validate/

- PageInsert - WorldCat Form
http://www.lib.montana.edu/~jason/files/ajax/

AJAX in Libraries Examples

- SingleSearch - Curtain University Library
  http://apps.library.curtin.edu.au/singlesearch/search.cgi

- Content Panes TERRA:The Nature of Our World - Montana State University Libraries
  http://lifeonterra.com

- Guesstimate Virginia Tech Libraries
  http://addision.vt.edu

- TAMU Geological Atlas of the United States
  http://repository.tamu.edu/handle/1969.1/2490

- Plymouth State University Lamson Library Catalog
  http://library.plymouth.edu/read/184908

- National Library of Australia Library Labs
  http://ll01.nla.gov.au/search.jsp?searchTerm=enigma

Additional Links and Examples of AJAX in Action

- VuFind Demo
  http://www.vufind.org/demo/

- NINES
  http://www.nines.org/collex

- Project Blacklight
  http://blacklight.betech.virginia.edu