Working with Javascript: Building Responsive Library apps
Code Samples, Explanations, and Downloads

Jason A. Clark
Head of Digital Access and Web Services
Montana State University Library
jaclark@montana.edu
www.jasonclark.info
twitter.com/jaclark

Tips – Common Javascript App Actions

1. Create markup
2. Call supplementary script
3. Customize javascript

Tips – Calling a Javascript Library

- Call as any other script on page
- Think procedural - request before starting to use script functions
- Use Google Hosted Libraries - Google Developers

```
<script src="http://ajax.googleapis.com/ajax/libs/jquery/2.0.0/jquery.min.js"></script>
```

Further Reading

- JavaScript: The Good Parts by Douglas Crockford
  http://www.amazon.com/JavaScript-Good-Parts-Douglas-Crockford/dp/0596517742/

- jQuery Cookbook: Solutions & Examples for jQuery Developers by Cody Lindley
  http://www.amazon.com/jQuery-Cookbook-Solutions-Examples-Developers/dp/0596159773/

- 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?

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 Hosted Libraries - Google Developers
  https://developers.google.com/speed/libraries/

Tutorials

- Javascript - Mozilla Developer Network
  https://developer.mozilla.org/en-US/docs/Web/JavaScript
- W3 Schools - Javascript Tutorials
  http://www.w3schools.com/js/default.asp/
- 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
- jQuery Documentation
  http://api.jquery.com/
- How jQuery Works
  http://learn.jquery.com/about-jquery/how-jquery-works/

Javascript in Libraries Examples

- NCSU Libraries Bento Box Search
  http://www.lib.ncsu.edu/search/?q=laser
- Guesstimate Virginia Tech Libraries
  http://addison.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
- VuFind Demo
  http://www.vufind.org/demo/
- NINES
  http://www.nines.org/collex
- Project Blacklight
  http://blacklight.betech.virginia.edu

Javascript – Sample Applications (Workshop Demos)

http://www.lib.montana.edu/~jason/files/javascript/

People @ Your Library (Ajax)
www.lib.montana.edu/~jason/files/javascript/showpeople/

Flickr API - Display Photos (Javascript)
www.lib.montana.edu/~jason/files/api/flickr/

Dynamic Definition List (toggle with jQuery)
www.lib.montana.edu/~jason/files/javascript/togglelist/

Livesearch of a Department/Subject List (jQuery)
www.lib.montana.edu/~jason/files/javascript/livesearch/

Toggle Search Form (Javascript)
www.lib.montana.edu/~jason/files/javascript/togglesearch/

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
- 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
- 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
- tags are given specific, relative widths, <ul> and <li> tags are styled to be listed inline

Code Sample #1: Flickr API - Display Photos (Javascript) – The URL Request

http://api.flickr.com/services/feeds/photos_public.gne?tags=cil2008&format=json

Code Sample #1: Flickr API - Display Photos (Javascript) - Explanation

• HTTP Request to Flickr API
http://www.flickr.com/services/api/
• API provides data as XML feeds (RSS, ATOM)
• Requesting "/feeds/" with a "format" of JSON (Javascript Object Notation)
• Querying API for all public photos tagged "cil2008" with the "tags" parameter

Code Sample #2: Flickr API - Display Photos (Javascript) – The URL Request in Javascript

```
<!-- use script tag to make request to flickr api, specify json format and tag to search -->
<script type="text/javascript"
src="http://api.flickr.com/services/feeds/photos_public.gne?tags=cil2008&format=json">
</script>
```

Code Sample #2: Flickr API - Display Photos (Javascript) - Explanation

• JSON is actually javascript and to make JSON output available we must call it on the page via the <script> tag
• After <script> tag is run, JSON output exists as javascript object ready to be parsed

Code Sample #3: Flickr API - Display Photos (Javascript) – JSON Response

```
jsonFlickrFeed({
"title": "Photos from everyone tagged cil2008",
"link": "http://www.flickr.com/photos/tags/cil2008/",
"description": "",
"modified": "2008-04-07T18:43:16Z",
"generator": "http://www.flickr.com/",
"items":
```

```
[

{
"title": "So many floors",
"link": "http://www.flickr.com/photos/nengard/2395908509/",
"media": {"m":"http://farm4.static.flickr.com/3182/2395908509_d6452e2d56_m.jpg"},
"date_taken": "2008-04-07T13:07:53-08:00",
"description": "So many floors",
"published": "2008-04-07T18:43:16Z",
"author": "nobody@flickr.com (nengard)",
"author_id": "10137764@N00",
"tags": "hyatt cil2008 cil08"
},
...

]
})
```


Code Sample #3: Flickr API - Display Photos (Javascript) - Explanation

- More structured data ready to be parsed
- We'll extract the values and format for display using the second javascript

Code Sample #4: Flickr API - Display Photos (Javascript) – Parse and display with Javascript

```
<script type="text/javascript">
//run function to parse json response, grab title, link, and media values - place in html tags
function jsonFlickrFeed(fr) {
for (var i = 0; i < fr.items.length;i++) {
document.write('<a title="' + fr.items[i].title + '" href="' + fr.items[i].link + '"><img src="'
+ fr.items[i].media.m + '" alt="' + fr.items[i].title + '"></a>');
}
}
</script>
```

Code Sample #4: Flickr API - Display Photos (Javascript) - Explanation

- Create javascript function "jsonFlickrFeed" to parse JSON response returned from first javascript
- Loop statement: "for (var i = 0; i < fr.items.length;i++)" runs through all JSON data nodes
- "document.write" – native javascript function prints out values from JSON in xHTML markup