jQuery, Ajax & the Same Origin Policy
Homework review

- What was your experience with Ajax?
  - Via XMLHttpRequest?
  - Via $.getJSON()?

- jQuery DOM manipulation
  - Which methods did you use?
  - Which did you find the most useful / surprising / interesting?

- Any creative solutions to demonstrate for the 5%?
  - Post the URL on Now
Ajax utility functions

- jQuery.getJSON()
  - or equivalently $.getJSON()
  - Arguments:
    - URL (required)
    - data arguments (optional)
    - callback function (necessary!)
  - The callback gets arguments
    - returned data
    - status code string
    - XMLHttpRequest
- jQuery Ajax Status Code Strings
  - "success" (HTTP 200 OK)
  - "notmodified" (HTTP 304 Not Modified)
  - "error" (one of the HTTP error codes)
  - "timeout" (if you set a timeout interval, default is none)
  - "parsererror" (if received malformed XML or JSON)
Ajax utility functions

• $.get ()
  – get request similar to $.getJSON() but without JSON.parse() of the response data

• $.post()
  – same as $.get(), except uses HTTP POST
Does Ajax request, puts returned data (typically HTML) into a matched element

E.g.
```
$('#responseArea').load('response.html');
```

Notice that .load() is a method, and can work on a set of selected elements.
- If the selection is null, the load is not done.

E.g. `$('div.foo').load('divContent.html')`
- What does this do?

The XMLHttpRequest is done asynchronously
• $.ajax()
  – The most complex, but offers the most control
  – Necessary when the simple cases don't apply
  – We will use this version shortly...
Browsing to a site is only as risky as the site you are visiting.

(You should only visit sites and give them information if you trust them.)

Using Ajax, the site can make additional requests within the context of the page.
Similarly, you can view your bank balance with limited risk.
The Cross Domain Problem

Criminal.com

HTTP Request

Bank.com

XMLHttpRequest
Directed by code
from Criminal.com
Using your credentials
to view balance at bank.com

Bank.com does not want other sites to initiate and control requests with your credentials.

This is risky for the user, and the browser will not allow the JavaScript from Criminal.com retrieve info from another site.
Same Origin Policy

• A broad security policy enforced by all browsers
• JavaScript from one domain cannot receive information via an XMLHttpRequest to another domain.
  – Regardless of whether it is a direct XMLHttpRequest or one done via jQuery.

• The request will be sent, but the response will be blocked by the browser.
  – Sending malicious actions is another problem called cross-site request forgery (CSRF)
  – The server (bank.com) can use other means to foil CSRF
Same Origin Policy

• Only applies to XMLHttpRequests
• You can access information from a 2nd domain using
  – img tags
  – script tags
  – iFrames

• Image & script tags only fetch information
  – The browser will display the image, but your JavaScript can't read the contents
  – Similarly, script will be interpreted, but you can't read it.

• iFrames are isolated from the original domain document.
  – It is like opening a separate window
  – JavaScript cannot reach across into the iFrame

• !!There is a nice summary discussion of this at:
Same origin policy

• Ajax can only be used for URLs that use:
  – The same protocol
    • e.g. http, https
  – And the same domain name
    • e.g. foo.com vs bar.com
    • e.g. 1.foo.com vs 2.foo.com
  – And the same port number
    • e.g. foo.com:3333 vs foo.com:4444
# Same origin or different?

<table>
<thead>
<tr>
<th>Source URL:</th>
<th>Same or Different?</th>
</tr>
</thead>
<tbody>
<tr>
<td><a href="http://foo.com/index.html">http://foo.com/index.html</a></td>
<td></td>
</tr>
<tr>
<td><a href="https://foo.com/page/start.html">https://foo.com/page/start.html</a></td>
<td></td>
</tr>
<tr>
<td><a href="http://foo.com:4532/index.html">http://foo.com:4532/index.html</a></td>
<td></td>
</tr>
<tr>
<td><a href="http://foo.com:80/index.html">http://foo.com:80/index.html</a></td>
<td></td>
</tr>
<tr>
<td><a href="http://www.foo.com/index.html">http://www.foo.com/index.html</a></td>
<td></td>
</tr>
<tr>
<td>Others?</td>
<td></td>
</tr>
</tbody>
</table>
## Same origin or different?

<table>
<thead>
<tr>
<th>Source URL:  <a href="http://foo.com/page/start.html">http://foo.com/page/start.html</a></th>
<th>Same or Different?</th>
</tr>
</thead>
<tbody>
<tr>
<td><a href="http://foo.com/index.html">http://foo.com/index.html</a></td>
<td>Same</td>
</tr>
<tr>
<td><a href="https://foo.com/page/start.html">https://foo.com/page/start.html</a></td>
<td>Different protocol</td>
</tr>
<tr>
<td><a href="http://foo.com:4532/index.html">http://foo.com:4532/index.html</a></td>
<td>Different port</td>
</tr>
<tr>
<td><a href="http://foo.com:80/index.html">http://foo.com:80/index.html</a></td>
<td>Same (port 80 is the default http port)</td>
</tr>
<tr>
<td><a href="http://www.foo.com/index.html">http://www.foo.com/index.html</a></td>
<td>Different domain</td>
</tr>
</tbody>
</table>

Others?
Proxy alternative

• As an alternative, server-side programming, acting as a proxy, can be used to:
  – interact with one or more web services
  – process the responses
  – communicate information to the browser

• In this case, the site cannot make use of the user's cookies for accessing the 2nd domain in their name (i.e. using their credentials).
Cross Origin Resource Sharing

• There is a W3C recommendation for allowing the 2^nd domain to decide whether it will allow cross domain access
World Wide Web Consortium (W3C)

International body of members/staff/public for developing web standards

– HTML
– CSS
– JavaScript
– PNG
– Accessibility
– Internationalization
– Privacy
– Mobile
– And more...
Cross Origin Resource Sharing

- There is a W3C recommendation for allowing the 2nd domain to decide whether it will allow cross domain access

- Essentially a new HTTP request header is sent indicating that this is a cross domain access
  
  Origin: "http://www.andrew.cmu.edu"

- And the server can choose to allow or deny cross origin sharing in the response header
  
  - Access-Control-Allow-Origin: "http://www.andrew.cmu.edu"
  - Access-Control-Allow-Origin: *

- Latest browsers implement it
Cross Origin Resource Sharing

• Slow acceptance – only a few public APIs implement it
• Some use OAUTH to do authentication first
  (We will discuss OAUTH much later)

APIs that support CORS

- Amazon S3
- DBpedia Spotlight
- Dropbox API
- Facebook Graph API
- Flickr API
- FourSquare API
- Google APIs
- Google Cloud Storage
- GitHub v3 API
- MediaWiki API
- prefix.cc
- PublishMyData
- sameAs
- SoundCloud API
- Spotify Lookup API
- Sunlight Congress API
- URIBurner
- YouTube API (blog post)
- doctape API

Source: http://enable-cors.org/resources.html
CORS Demo

- Show sameOriginPolicyExample.html
One way around the restriction is to use JSONP
- Stands for JSON-Padded
- JSON is padded with (wrapped within) a function call
- This is a misuse of the script tag to get around a useful browser security policy ... use with caution.

E.g.
- If the JSON to be returned is: \{x: 1, y:2\}
- With JSONP, this is returned: \pcallback\(\{x: 1, y:2\}\)
  - I.e., the JSON is an argument to a function call

This can be put in a \(<script>\) tag
- \(<script src=\"jsonpexample.json\">\</script>\)

After the JSONP is loaded, the function is called with the data as the function parameter
- \pcallback\() // You develop \pcallback\() and use the JSON
Walk through alFlickrbet
Examples of another site that uses jsonp

• Geonames
JSONP Public APIs

• List of APIs that make JSONP available
  – http://www.programmableweb.com/category/all/apis?keyword=jsonp
BIG CAVEAT – POSSIBLE DANGER

• You need to trust the 3rd party site you are requesting JSONP from.

• JSONP blindly executes code that a 3rd party is providing
  – So make sure you (designer of the web site) trusts the 3rd party site you are taking the user to.
  – Else it can do nasty things
The safest way...

- The safest way to use 3rd party APIs is to do it on a server
  1. User (aka client or browser) makes request to server
  2. Server *middleware* makes a request to the 3rd party API
  3. Server middleware processes the response
     - Processing may be minimal or complex
  4. Server middleware passes a response back to the browser
  5. Browser uses the response.
- This is what we will begin in a few weeks.
Homework

• Monday – Demonstrate calling an API
• Following Monday (3Oct) – Mini Client Project

See web course schedule for details.