AJAX & JSON

Let's talk to someone

Today’s Agenda

- Resume chat
- AJAX
- JSON

My Resume Tricks

1. Make an impossibly large resume that documents everything you've done.
2. I mean everything. Volunteer stuff, work stuff, everything. This is your MEGA resume.
3. Whenever you apply for a job or position, build a one-page resume specifically for that position, pulling from your MEGA resume.
4. Never remove anything from your MEGA resume – always add to it.

AJAX

Asynchronous Javascript And XML

A technique for loading data into part of a page without having to refresh the entire page.

This is the secret behind the single-page web application.

Places You've Seen AJAX

- Google Autocomplete
- Shopping Carts that don't do page refreshes
- username availability
- Social Media widgets on webpages
- live editing of content
- calendar systems
- Google Docs
- (mostly just all of Google, actually)

AJAX is Asynchronous

- the browser does not hold while the AJAX call is processing
- allows you to update just specific parts of your site that you want to update
- You make the call, go on with your site load, the response from the server shows up eventually and you deal with it.

Making the Call

You use an XMLHttpRequest object to make the call for you.

```
var xhr = new XMLHttpRequest();
xhr.open('GET', 'data/test.json', true);
xhr.send('search=arduino');
```

open()

xhr.open(‘GET’, ‘data/test.json’, true);

open() has 3 parameters:

1. HTTP Method
2. url of the page handling your request
3. bool for if it should be asynchronous

send()

send() is what actually sends the request, and can accept any extra info you want to send with your call.

This is where you send any additional data you might need to send with your request

onload()

xhr.onload

- onload is set to a function, establishing the event handler for when the AJAX call returns.
- the XMLHttpRequest object will have a *status* property, which will determine what the status of the call was
- xhr.status === 200 { //all's good! Do something! }
- other statuses (304: Not modified; 404: Page not found; 500: Internal error on the server)

Data Formats

HTML
- Same ol' HTMl, generated and passed from the server to be added directly to your page

XML
- a mark-up language similar to HTML, but it describes data and is much more strict than HTML

JSON
- JavaScript Object Notation
- uses object literal notation to represent data

HTML & XML can only come from within the same domain!

XML

Extensible Markup Language

- XML can be used to create markup languages for any type of data
- the creator of the data file decides the name of tags
- XML works on any platform!
- it is extremely flexible and can represent complex data easily
- you can navigate XML the same way you navigate the DOM

XML Example

```
<menu id="file" value="File">
  <popup>
    <menuitem value="New" onclick="CreateNewDoc()" />
    <menuitem value="Open" onclick="OpenDoc()" />
    <menuitem value="Close" onclick="CloseDoc()" />
  </popup>
</menu>
```

JSON

Pronounced “Jason”, JavaScript Object Notation is very similar to object literal syntax, but it is *not* an object.

- JSON statements are made up of key and value pairs
- keys must be in double quotes

JSON Example

```
{"menu": {
  "id": "file",
  "value”: "File",
  "popup": {
    "menuitem": [
      {"value": "New", "onclick": "CreateNewDoc()", “onStatus”:true},
      {"value": "Open", "onclick": "OpenDoc()", “numClicked”: 3},
      {"value": "Close", "onclick": "CloseDoc()"}
    ]
  }
}}
```

JSON Continued

- JSON returned from a server is just a string, but you can use the *JSON object* to convert into Javascript objects
  - JSON.parse()
- likewise, you can use the JSON object to convert Javascript into a JSON string
  - JSON.stringify()

GET vs POST

GET - get data from the server

- GET requests can be cached
- GET requests remain in the browser history
- GET requests can be bookmarked
- GET requests should never be used when dealing with sensitive data
- GET requests have length restrictions
- GET requests should be used only to retrieve data

POST – post data to the server

- POST requests are never cached
- POST requests do not remain in the browser history
- POST requests cannot be bookmarked
- POST requests have no restrictions on data length