AQUENT

GYMNASIUM

JQUERY: BUILDING BLOCKS

Lesson 6

MATERIALS & ASSIGNMENTS

THIS LESSON'S CORE CONCEPTS

In this lesson you will learn how to work with dynamic page content.

1. "AJAX" is the outdated buzzword for what is now the universal process of web pages reaching out to servers after load, and fetching and using additional data. Facebook and Google Maps are canonical examples of web pages which make very heavy use of AJAX to load news and chats, or map tiles and search results. Even mostly static pages like Yahoo! or the New York Times have a few dynamic widgets. You will never in your life need to know what "AJAX" stands for.
2. Data exchange works best when it's structured, and the state of the art data structure format is called JSON, which stands for JavaScript Object Notation, and looks very much like the JavaScript objects you're familiar with. This makes it very easy to use in JavaScript, and has largely replaced the more verbose XML as the Internet's data format of choice.
3. JSON is made up of objects and arrays full of values. Those values may be other objects and arrays, strings, numbers, true, false and null. Object keys must be in quotation marks, like this:
   ```
   { "firstKey": "value", "secondKey": 42 }
   ```
4. Once we have our distilled data in hand, we have to use it to dynamically generate HTML for the user to see, via a process called "templating". Templating is the name for any system that takes partial HTML strings (called templates) and adds data into them.
5. There are many templating systems, including some jQuery plugins. If you have complex needs, I recommend checking out Mustaches. If your needs are simple, you can roll your own rudimentary system very easily using string replacement:
   ```
   var myTemplate = '<div>%{greeting}!</div>',
         myHtml = myTemplate.replace('%{greeting}', 'Hello');
   ```
   Note however that JavaScript's "replace" method only replaces the first instance it finds, so beware if you're trying to drop a piece of data into multiple places in a string.
6. JSON is what our data looks like, and templating is how we get it into the HTML so the user can see it; now we need to reach out and fetch it from the server. AJAX is powered by "XHR", which stands for "XML-HTTP Request", even though you'll rarely use it with XML. WIth XHR objects, you can request data from a server, and use it when it returns. Working with XHR directly is fairly complex, and is a notorious hive of cross-browser compatibility, but jQuery takes care of all of that for us with the `$.ajax() method.`
7. The `$.ajax()` method takes as an argument a single hash of options. See the Further Reading sections for a link to the full documentation. Here are the most important ones.
   a. Data on the web lives at specific addresses, just like web pages, images and stylesheets.You specify the data's address with the "url" option.

   b. Requesting data is no use if you don't do something with it when it comes back; you can specify a callback function for this via the "success" option. (This only gets called if the server successfully responds to your request; you can also specify an "error" callback for when the server fails to respond, and a "complete" callback which is called whether the request succeeds or fails. For advanced needs, note for your Googling pleasure that the `$.ajax()` method returns a "jqXHR" object, which conforms to the "Promise" interface.)
   c. You may optionally specify a "dataType" option, describing what format of data you expect to receive. By default, jQuery does an excellent job guessing; you can speed things up slightly by specifying "json" or (less often) "xml". Another option is "jsonp", which we will cover below.
8. So to make a dynamic web page, you use `$.ajax(options)` to request data from a server at a particular URL; you specify a callback to be run when the data returns; your day will likely come down in the JSON format; and you will use some form of templating to turn the raw data into visible HTML.
9. A "web API" is any set of URLs which systematically expose data. For example, in the process described above, your server is exposing your data via an API. There also exist public web APIs for many different kinds of data. For example, openweathermap.org and forecast.io are two weather APIs which are free for a small number of requests. (You must sign up with them for a free "API key", which allows your site to identify itself.) The World Bank exposes a comprehensive set of global demographic data via its API (see "The World Bank API Call" below), which does not require an API key, and is great for experimenting with even if you don't care about percentage access to clean water by country.
10. To access data from other servers like these, your `$.ajax()` call must specify a dataType of "jsonp". This safely bypasses "same-origin" restrictions (a frustrating but absolutely essential internet security precaution). APIs which do not support JSONP are not generally accessible from other websites.

FURTHER READING:

- Full `$.ajax()` documentation: http://api.jquery.com/jQuery.ajax/
- Experiment with dynamic data via the World Bank API: http://data.worldbank.org/node/9

The World Bank API Call:

```
$.ajax({
   url: "http://api.worldbank.org/countries/indicators/1.1_ACCESS.ELECTRICITY.TOT",
   data: { per_page: 100, date: "2000:2013" },
   success: function(data) { console.log(data); /* Play here! */ },
   dataType: 'jsonp',
   jsonp: 'prefix' // (WB API uses a non-standard callback name)
});
```

ASSIGNMENTS:

1. Quiz
2. Remove our last piece of hard-coding: get the category to fetch and display from the URL.