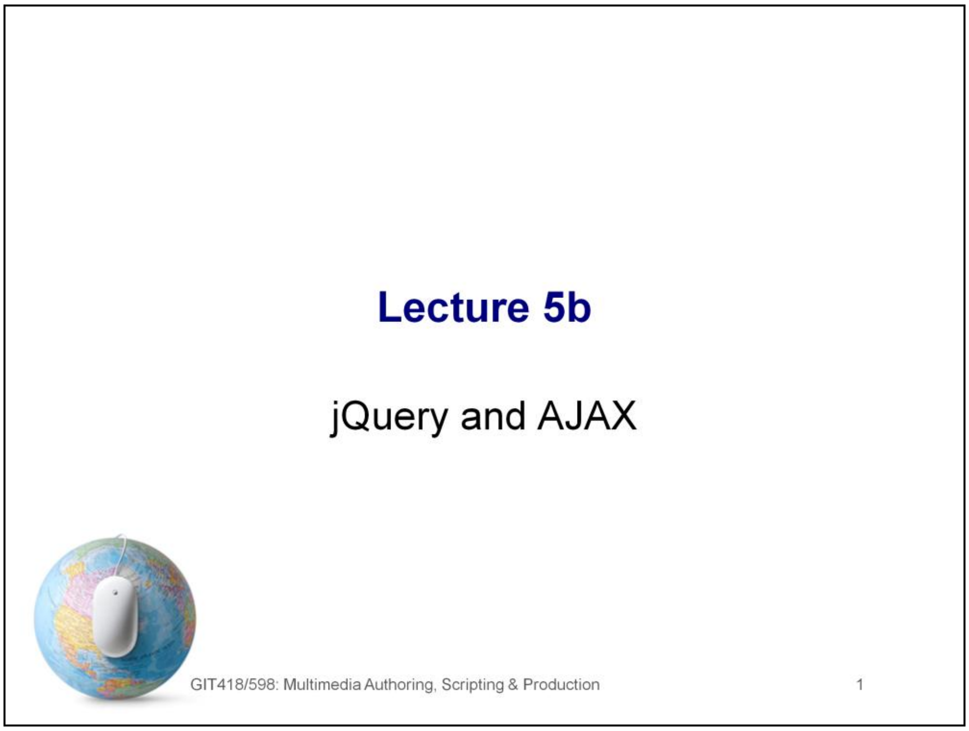
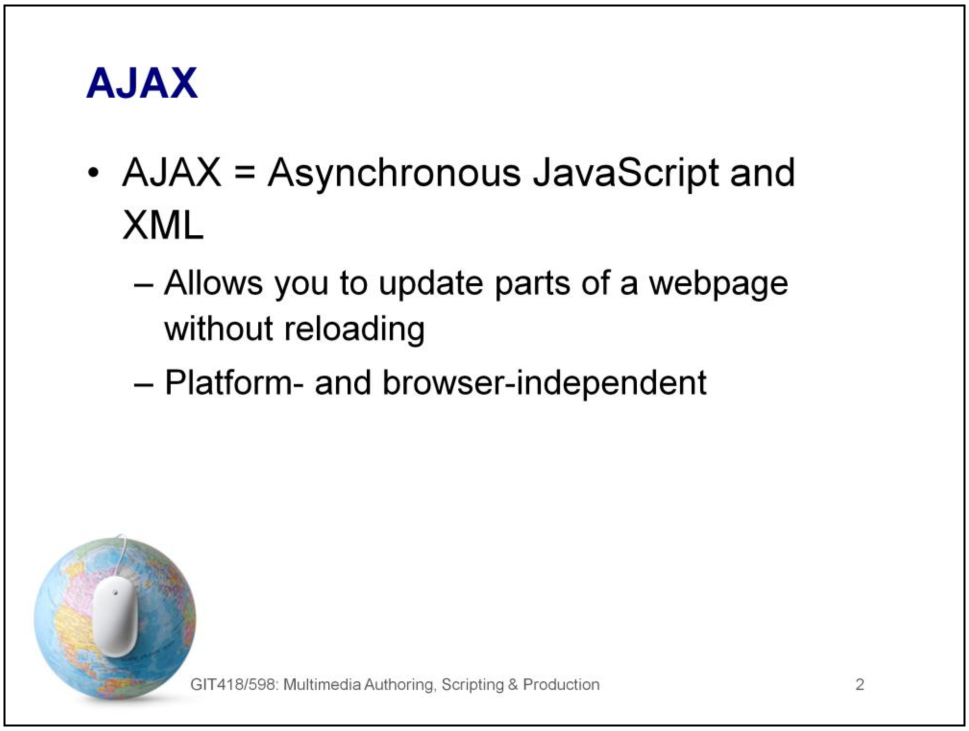


Lecture 5b

jQuery and AJAX



AJAX

- AJAX = Asynchronous JavaScript and XML
  - Allows you to update parts of a webpage without reloading
  - Platform- and browser-independent

GIT418/598: Multimedia Authoring, Scripting & Production 2

Ajax requests are triggered by JavaScript code; the code sends a request to a URL, and when it receives a response, a callback function can be triggered to handle the response. Because the request is asynchronous, the rest of your code continues to execute while the request is being processed, so it's imperative that a callback be used to handle the response.

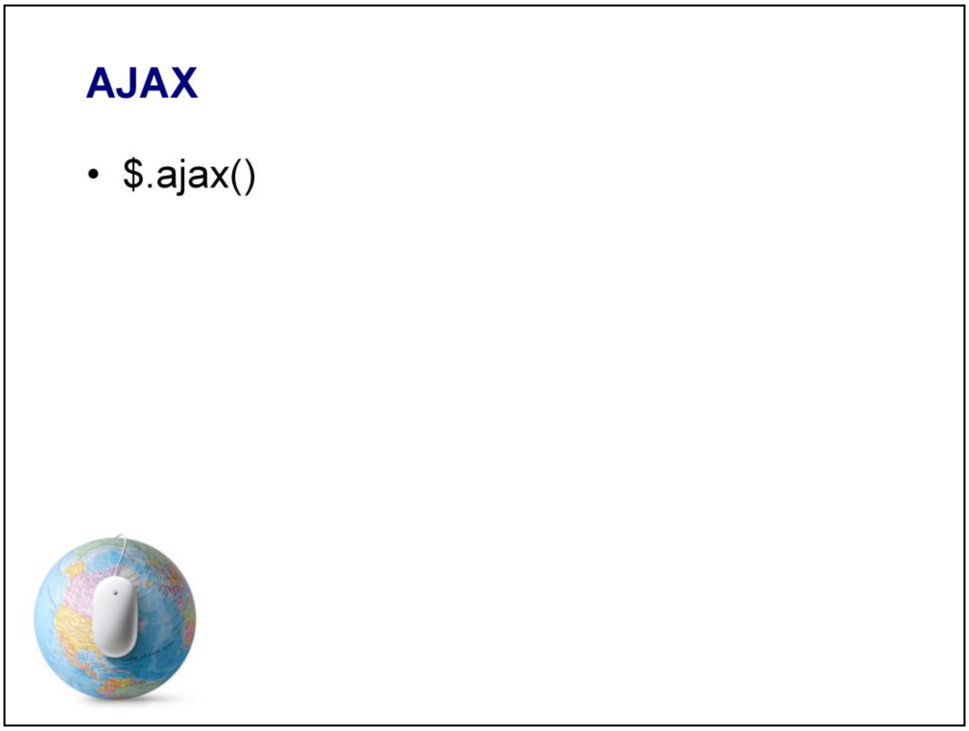
What I'm going to show you is just a small part of what AJAX can do.

AJAX

- What AJAX uses:
  - – Method to request data from server
  - – jQuery
  - – XML, TXT, JSON, HTML, script
  - – CSS

The method is used to talk with the server asynchronously. With JavaScript, this would be the XMLHttpRequest. jQuery uses the DOM to interact with the data. CSS is used to style the information. XML is usually the format for transferring the data, but JSON is common as well.

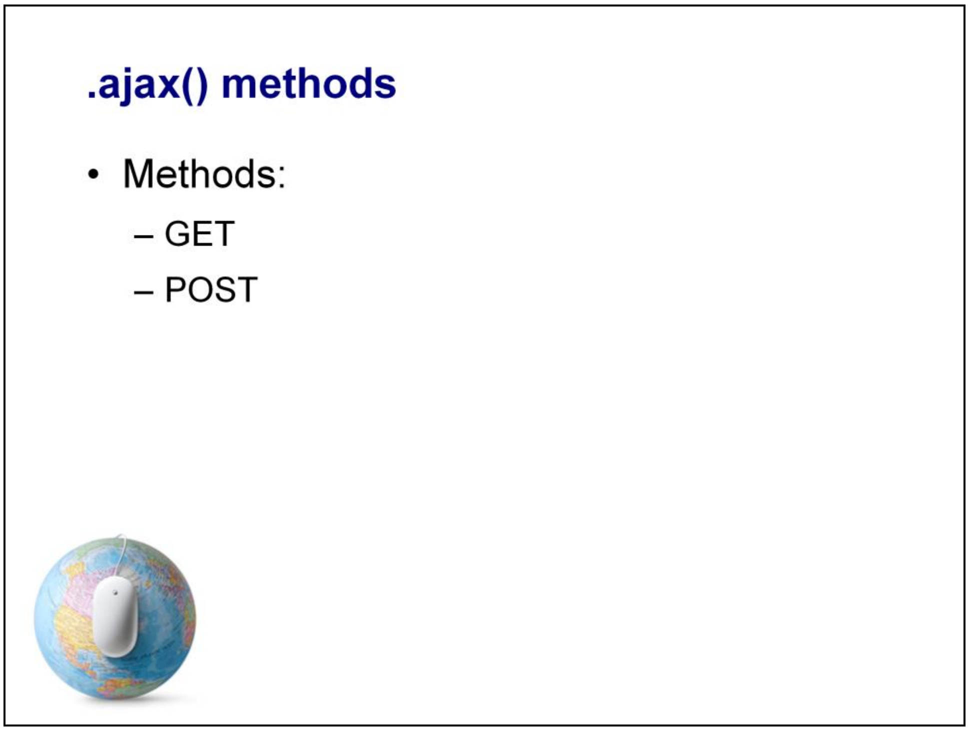
JSON is similar to XML but it's taking over as far as the format of data for AJAX calls. See, there's something called same-domain issue, and JSON gives us a way around that. We'll talk about that in a few slides.



AJAX

- $.ajax()

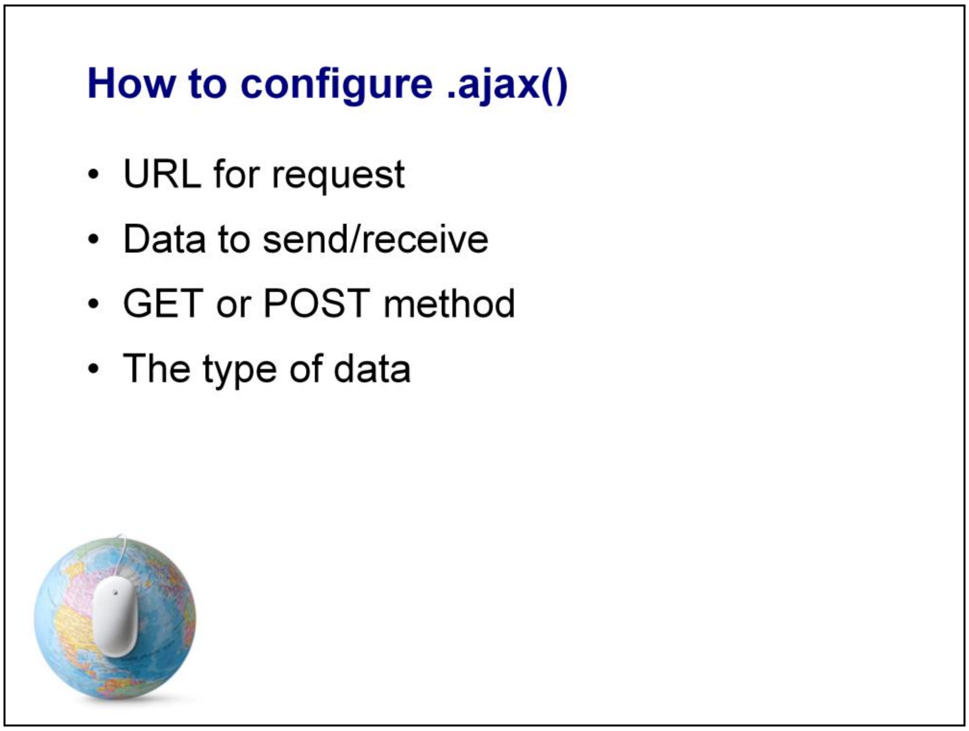
The method that is used is the .ajax method.If you’ve written JavaScript code to accomplish this, you can see that typing .ajax is much easier than the XMLHttpRequest and all of the associated code with that.



.ajax() methods

- Methods:
  - GET
  - POST

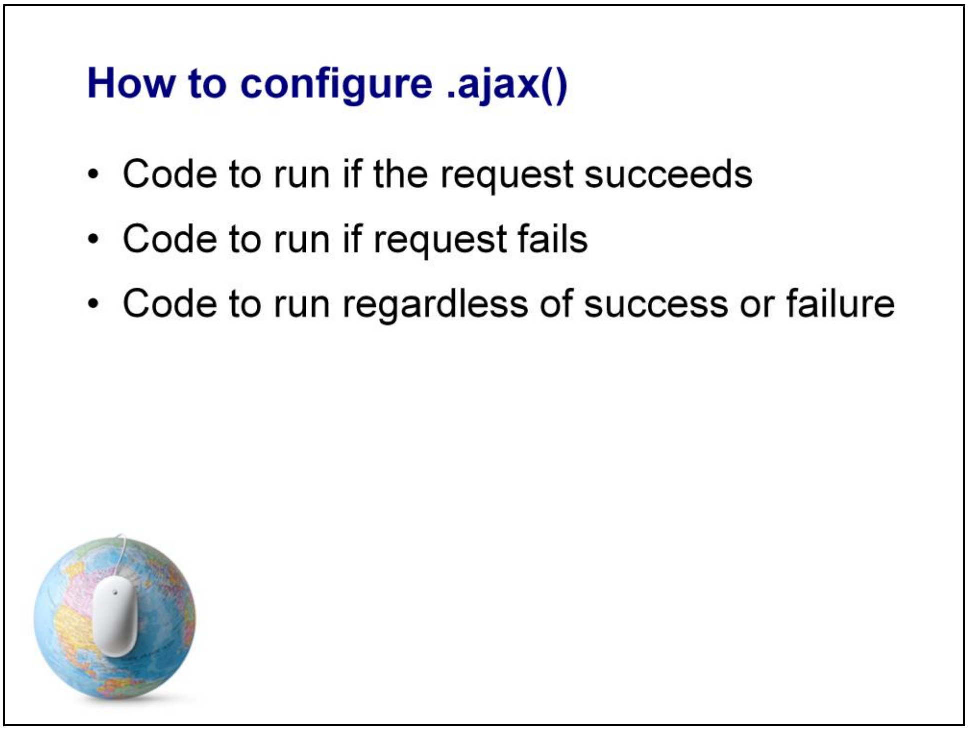
The GET method is used for getting data from a server. The POST method is used to get data but you also would be changing the data. For example, if you want to save a blog post to the server, you would use the POST method.



How to configure .ajax()

- URL for request
- Data to send/receive
- GET or POST method
- The type of data

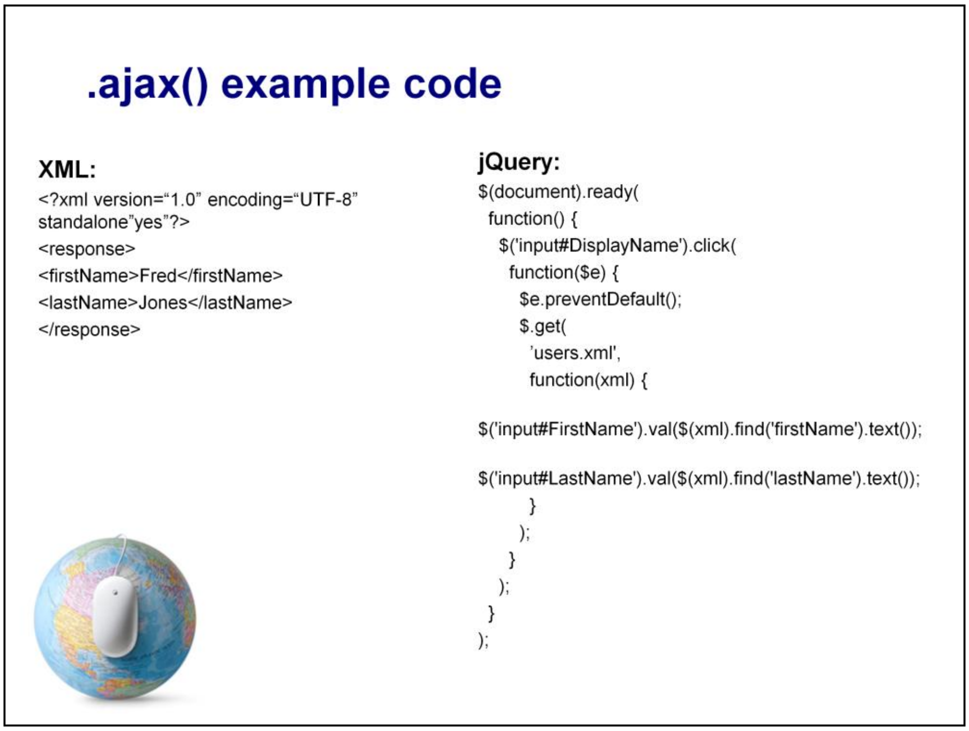
The GET method is used for getting data fro a server. The POST method is used to get data but you also would be changing the data. For example, if you want to save a blog post to the server, you would use the POST method. The POST method also prevents the form from resubmitting if the user navigates back using the back button on the browser.



How to configure .ajax()

- Code to run if the request succeeds
- Code to run if request fails
- Code to run regardless of success or failure

You also could add the following to the code. It’s not required but it’s a good idea because it lets you know when or if your code works.



.ajax() example code

XML:

```
<?xml version="1.0" encoding="UTF-8"
standalone"yes"?>
<response>
<firstName>Fred</firstName>
<lastName>Jones</lastName>
</response>
```

jQuery:

```
$(document).ready(
  function() {
    $('input#DisplayName').click(
      function($e) {
        $e.preventDefault();
        $.get(
          'users.xml',
          function(xml) {

$('input#FirstName').val($(xml).find('firstName').text());

$('input#LastName').val($(xml).find('lastName').text());
          }
        );
      }
    );
  }
);
```

Here is an example of an AJAX call. The first line is our standard document.ready statement. Then we write a function that is a click event. When the user clicks on the input field with the ID of DisplayName, it will trigger another function.

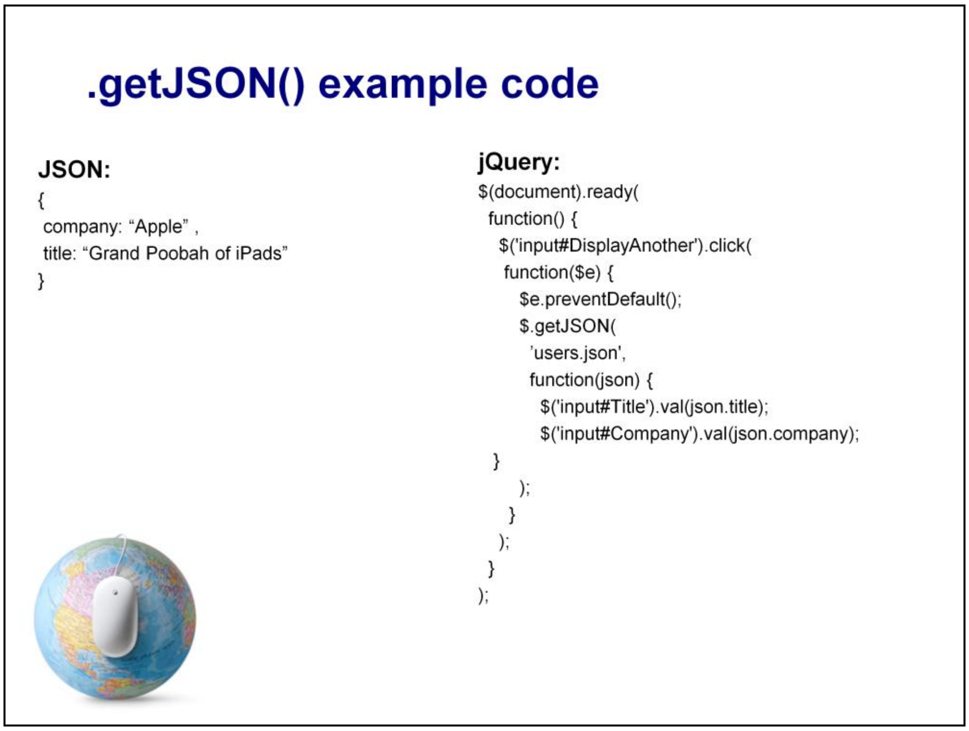
The next two lines are another function that sets up to keep the default action to happen. The default action is to have the user type text into the field. We don't want them to do that because we are going to provide the text for them via XML.

So the next one is the get, which is asking for the XML file with the name of users. Then we have a function set to grab the data from the XML file that is requested. The next line with input#FirstName.val is asking to find the text for the XML tag firstName, and then input that text, or value, into the field with the ID of firstName. The second one does the same, but grabs the text for LastName and inputs that text into the field.

.getJSON()

- Use when you want to grab JSON data

You can use .get in order to get it, but this allows for less code to be written, and it lets the browser know that it’s definitely going to be JSON data that is retrieved from the server.



.getJSON() example code

JSON:

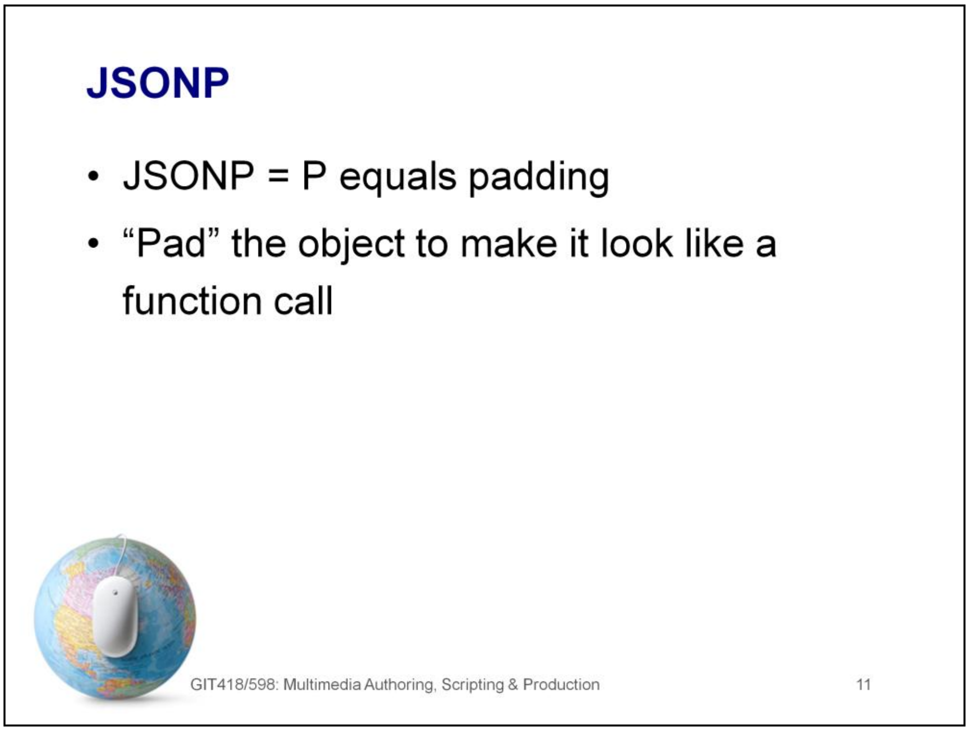
```
{
 company: “Apple” ,
 title: “Grand Poobah of iPads”
}
```

jQuery:

```
$(document).ready(
  function() {
    $('input#DisplayAnother').click(
     function($e) {
        $e.preventDefault();
        $.getJSON(
          'users.json',
          function(json) {
            $('input#Title').val(json.title);
            $('input#Company').val(json.company);
   }
        );
      }
    );
  }
);
```

Here is an example of an AJAX call using JSON. The first line is our standard document.ready statement. The next two lines are another function that sets up to keep the default action to happen, just like the XML example.

Then it gets different. We use getJSON method, request the JSON file again, then we get the data to input into the input fields, just like before. But notice that the code is shorter. We just need to specify json and the specific element from json that we want. The first one is the title, and second is the company.



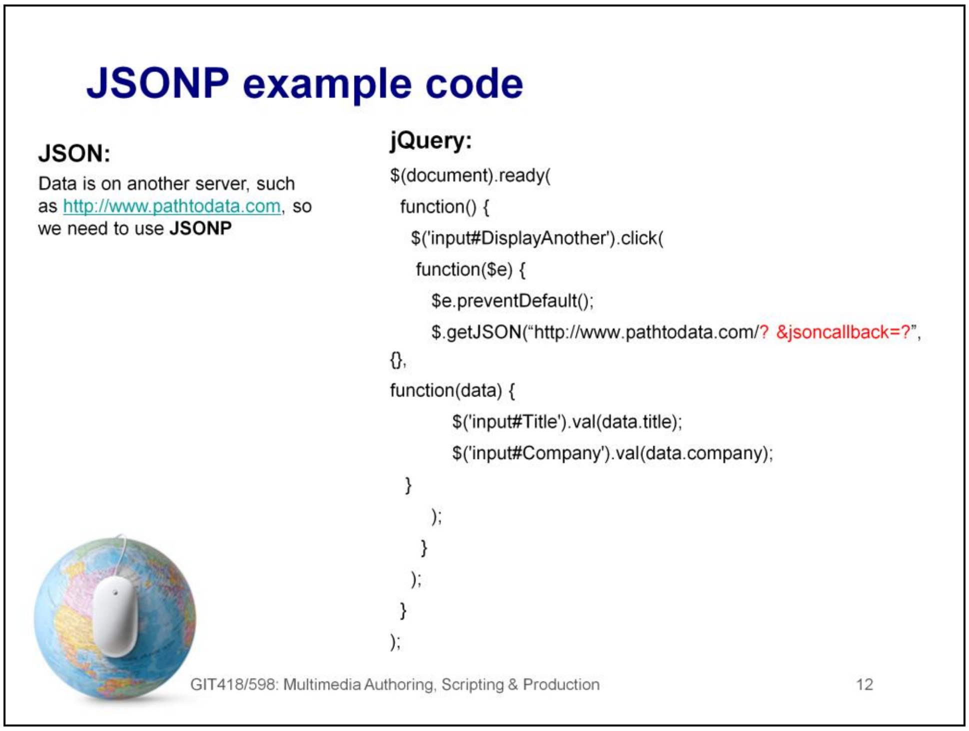
JSONP

- JSONP = P equals padding
- "Pad" the object to make it look like a function call

GIT418/598: Multimedia Authoring, Scripting & Production 11

So we have JSON working really well when we call a JSON file from our own web server or from our local computer. But what happens when we try to call a file from an external server. If you use the code on the previous screens for JSON, it may not work. It's because some browsers have something called a same-domain policy. In other words, you can only request data from the same server where your file is located. If you want to request data from a server other than your own, then you can use the JSONP method. It's JSON with a P, and the P stands for Padding. What happens is that with JSONP, what we are doing is essentially tricking the browser to think the files are on the same domain because scripts do not have that same-domain policy. It's just a script and can be called from anywhere We're calling a script, so the browser has no problem with the same-domain issue.

We also can use this to grab data from publicly available APIs or those where we can get a key, such as Google API.


JSONP example code

JSON:

Data is on another server, such as http://www.pathtodata.com, so we need to use **JSONP**

jQuery:

```
$(document).ready(
  function() {
    $('input#DisplayAnother').click(
     function($e) {
        $e.preventDefault();
        $.getJSON("http://www.pathtodata.com/? &jsoncallback=?",
{},
function(data) {
            $('input#Title').val(data.title);
            $('input#Company').val(data.company);
   }
        );
      }
    );
  }
);
```

GIT418/598: Multimedia Authoring, Scripting & Production 12

We create an anonymous function that will handle the response we get back from the server with the getJSON line. We will get what is called a "callback" with the data. There's a second parameter to that getJSON function that needs to be an object literal representing additional data passed to the API. And the 3rd parameter is an anonymous success function, and it's what we write to actually do something with the data we get if we successfully get it.

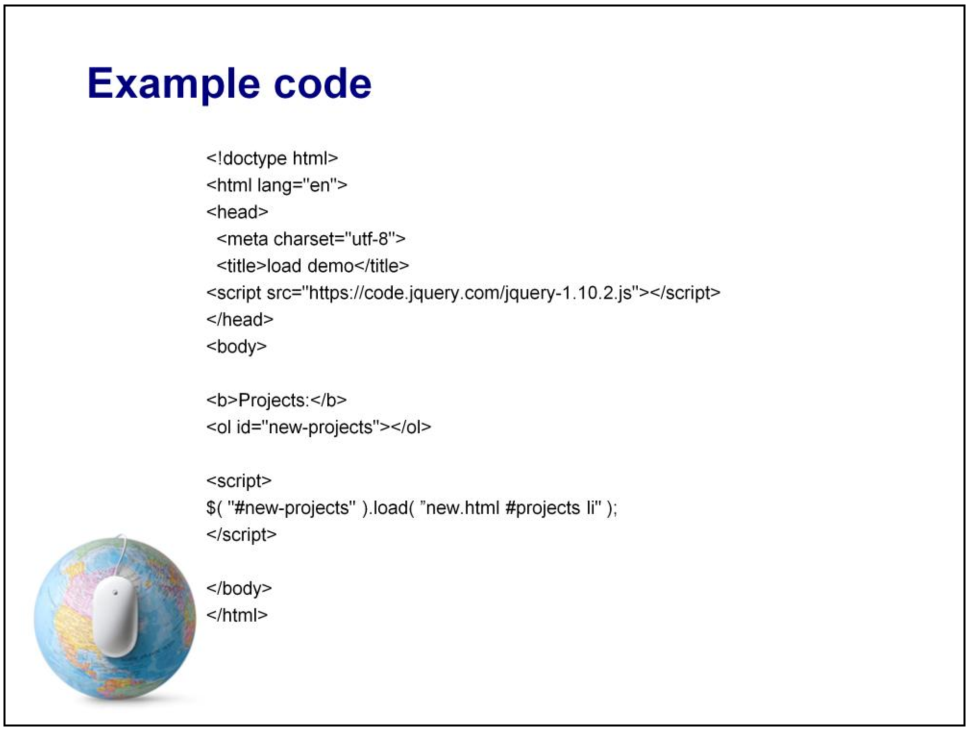
The main thing I want you to understand is that JSONP is more or less a 'hack' to get around the different domain issues when no authentication for using the API is necessary. This is because web browsers are inherently trusting of remote JavaScript files downloaded from a different domain. Proof of this is evident when a web developer chooses to make use of the jQuery JavaScript file being served from another server like the jQuery.com content distribution network (CDN) or Google's CDN. The browser doesn't care where the code is coming from as long as it's valid JavaScript code. We are inherently trusting that no malicious code is coming from those sites. For other types of hosted code, we probably wouldn't want to do this because someone could use their server to hack the JavaScript code and insert all sorts of things into a web page to link to, manipulate the web document, or otherwise directly download malicious software to the user’s machine or device.

We’re essentially tricking the browser into receiving JSON data from a remote unauthenticated source provided that the API can wrap it in a function call. The code you see on the screen is the way the Flickr API is set up (HINT, HINT!). You can either provide a callback function name or send the callback function anonymously in a JSON object to the API. Other RESTful web services may have documentation to support a JSONP hack like Flickr’s but one would have to consult it. That’s important – it all depends on the way the API provider has set up their API, and what type of information they provide, and how you have to “ask” for it. There are probably as many different ways of using JSONP as there are remote web services that supply data in JSON format. The public API provider needs to set up their code to allow for JSONP usage if there’s no authentication needed.

.load()

- Allows you to load data from server and place returned HTML into element

One last useful way to get information via AJAX is through the load method.



Example code

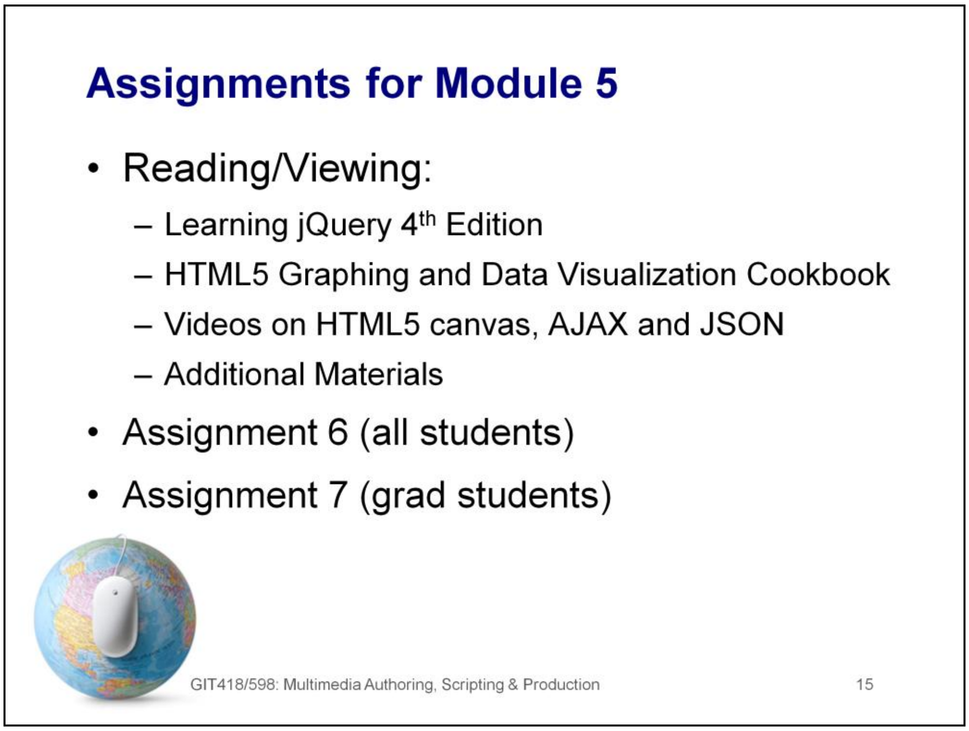
```
<!doctype html>
<html lang="en">
<head>
  <meta charset="utf-8">
  <title>load demo</title>
<script src="https://code.jquery.com/jquery-1.10.2.js"></script>
</head>
<body>

<b>Projects:</b>
<ol id="new-projects"></ol>

<script>
$( "#new-projects" ).load( ”new.html #projects li" );
</script>

</body>
</html>
```

What is happening is that the load is asking for the file new.html. Then it’s going to parse that file and only get the data from the element with the ID of projects. It’s going to take that data and add it as a list to the ID on this page with the name of new-projects. The load has a built-in innerHTML of sorts.



Assignments for Module 5

- Reading/Viewing:
  - Learning jQuery 4th Edition
  - HTML5 Graphing and Data Visualization Cookbook
  - Videos on HTML5 canvas, AJAX and JSON
  - Additional Materials
- Assignment 6 (all students)
- Assignment 7 (grad students)