AJAX with jQuery

ACM Webmonkeys 2011

What is AJAX?

- AJAX is a technology used to facilitate real-time data changes and updates on a page without requiring a page reload.
- AJAX stands for **A**synchronous **J**avascript **A**nd **X**ML.
- Let's break that down:
  - Asynchronous: The response from the server doesn't have to be immediate, like a page load does. Other stuff can happen in-between.
  - Javascript: The client-side language which you use to make requests to and handle responses from the server
  - XML: The format often used to pass data between Javascript and the server.

No, really, what is AJAX?

- AJAX is basically a general term for making your webpage able to do dynamic stuff on the server (like make a new post, remove a user, etc) without having the user click on a link which loads a new page.
- It works similar to how any fancy JavaScript effect works, except that the JavaScript makes a call to some page on the server in the middle which causes something "real" to happen.
  - Remember, JavaScript is client-side. It can't affect the database directly. It needs to use a technique like AJAX to cause an actual effect on the server.

A simple example

get_stuff.php

```
<?php
echo "This is the stuff that will go in
the box.";
?>
```

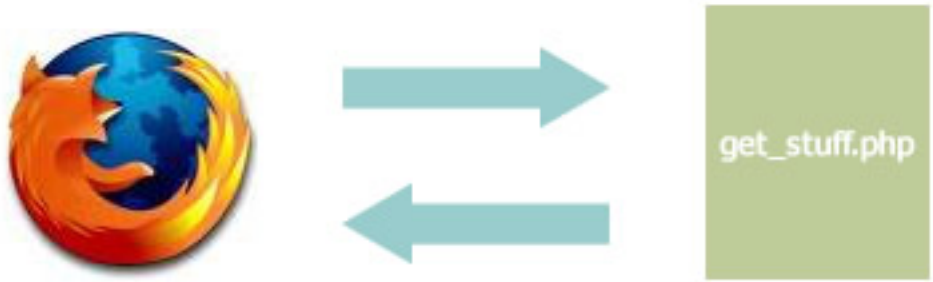
mypage.html

```
<h1>Stuff</h1>
<div id="box">
</div>
<a href='javascript:getStuff();'>Get stuff via AJAX.</a>
<script type='text/javascript'>
function getStuff() {
  // Pseudocode for simplicity
  request = makeHttpRequest("get_stuff.php");
  document.box.innerHTML = request.text;
}
</script>
```

What happens?

1) Server sends mypage.html to browser, which displays the page and the link.

2) User clicks the link, causing JavaScript to tell the browser to make a separate HttpRequest.



3) The output of get_stuff.php is returned to JavaScript eventually, and the result is put into the #box div.

Main idea

- With AJAX, we use JavaScript to load a page, but the output is not loaded directly onto the page.
  - The AJAX request will look the same to the server as if the user loaded the page in their browser.
  - The output must be handled in JS to do what you want.
- Don't confuse normal JavaScript functionality with AJAX -- it only becomes AJAX when a remote call is involved.
  - Kind of like RPCs for HTML.

So, how do we write pages with AJAX?

- There are two sides to an AJAX feature:
  - The JavaScript, which makes the request
  - The dynamic page, which does something and returns a response
- The dynamic side can be done in PHP, ASP, Ruby on Rails, C++, or anything that runs on your webserver.
- The JavaScript side needs to be done in JavaScript, of course, but we have a couple options:
  - Straight, platform-specific JS (painful)
  - Wrapper libraries (like Prototype or jQuery)

What is jQuery?

- http://www.jquery.com/
- jQuery is a JavaScript library based around the idea of attaching functionality to selected groups of elements.
  - The main function in jQuery is the selector, $, which works like:
    - $("#thisisanid") -- finds the element with that id
    - $("div") -- finds all divs
    - Many more syntaxes are supported.
  - Once a group of elements (or a single element) has been selected, you can call jQuery functions that act on the entire group.
- It also has some very easy-to-use AJAX wrappers.

More on jQuery

- The general structure of writing jQuery is to define *events* on elements; for example:
  - $("#somelink").click(function() { /*...*/ });
- You typically pass in a function to the event function, which specifies what happens when that event is triggered.
  - The function passed in is a closure, meaning it can refer to any variable in the current scope.

```
$("somelink").ready(function() {    // EVENT
   $("somelink").fadeOut();         // ACTION
});
```

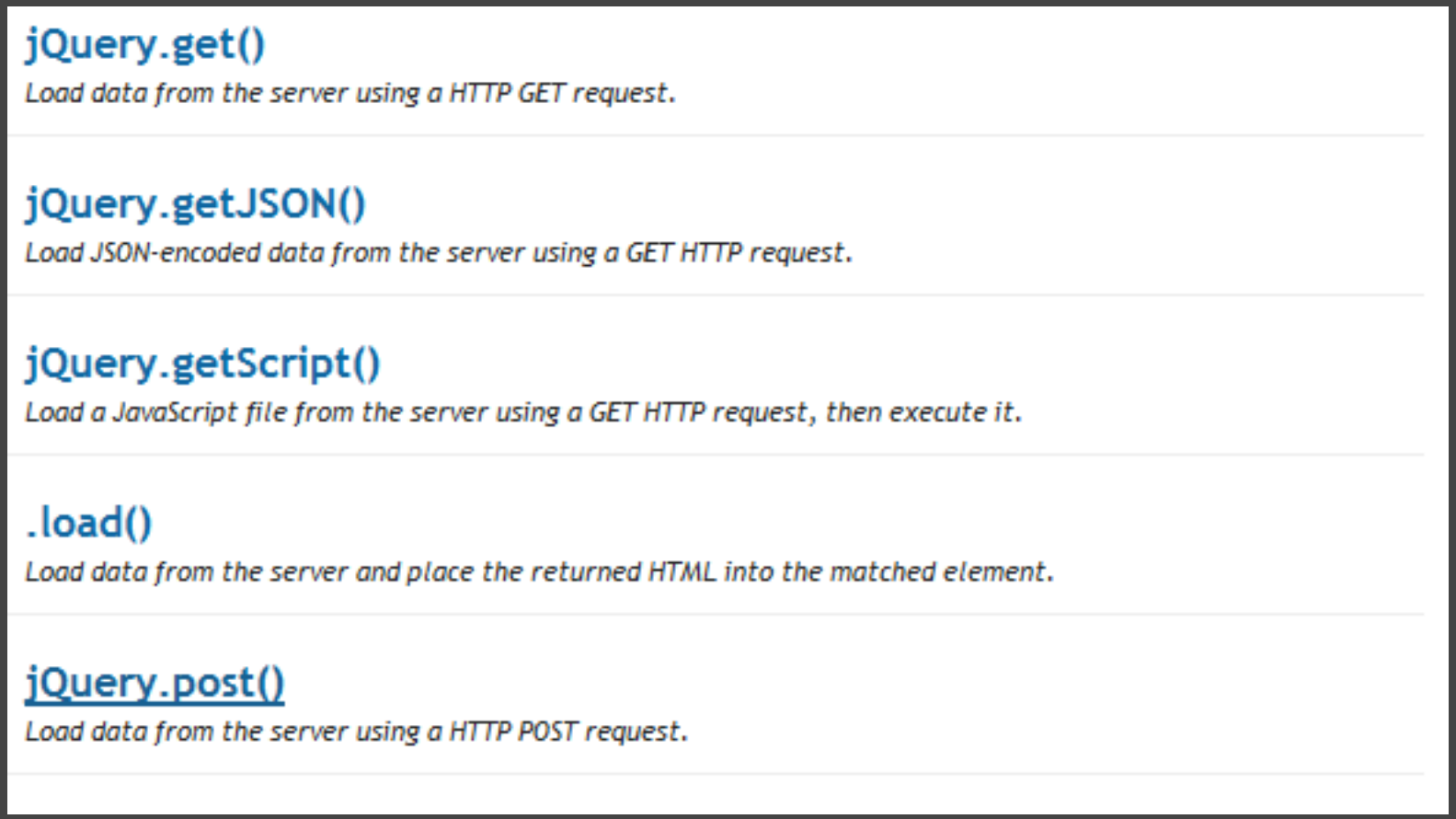
A simple example of jQuery

```
<script type="text/javascript" src="jquery.js"></script>

<body>
<div id="thediv" style="display:none;">
Content not yet loaded.
</div>

<script type="text/javascript">
$(document).ready(function() {
    $("#thediv").fadeIn();
});
</script>
</body>
```

Adding AJAX

- That example doesn't use AJAX, just normal JavaScript.
- Let's add some AJAX by making the DIV load its contents from another page.
  - For now, let's just use a static page. It makes no difference to JavaScript whether the response comes from a truly dynamic page, so using static pages is useful for testing your AJAX code.
- Let's first take a look at what jQuery has to offer for AJAX...

jQuery's shorthand AJAX methods



Let's look more in depth at the get() method.

jQuery.get()

The get() function is a method of the jQuery object, so we might call it like $.get('blah.php'). The full arguments are:

Description: *Load data from the server using a HTTP GET request.*

jQuery.get(url, [data], [success(data, textStatus, jqXHR)], [dataType]) version added: 1.0

url A string containing the URL to which the request is sent.

data A map or string that is sent to the server with the request.

success(data, textStatus, jqXHR) A callback function that is executed if the request succeeds.

dataType The type of data expected from the server. Default: Intelligent Guess (xml, json, script, or html).

Making a GET request with AJAX

- As a reminder, GET data is passed as URL arguments (e.g., ?foo=bar&a=c&b=2). So, a GET request is just loading a specific URL with those arguments holding the request data.
- So, if we want to load the URL data.txt (no arguments being passed) and put the response (just the file's text) into the div, we could do:

```
$.get('data.html', function(response) {
 alert("Data incoming...");
 $("#thediv").html(response);
});
```

Even simpler

- If all you need to do is make a simple GET request, you can just use the load function (applied to an element directly). For example, we could just do:
  - `$("#thediv").load('data.html');`

Applying AJAX

- Now that we've seen how to make an AJAX request, let's take a look at how to use it to improve our web pages.
- Remember, AJAX is just a way to make a web request without reloading the page.
  - We can't do anything with it that we couldn't do without it, but we can make things seem faster, and not require reloading the page.

Checking username availability

- Suppose we have a site that lets users create accounts with custom usernames. However, no two users can share the same username, so during the registration process we need to let them know if there's a conflict.
- We could do this by checking after they submit the form, but let's write a system that automatically checks availability.
- The process we want is:
  - User tabs out of the username field
  - Make an AJAX request to see if username is taken
  - Update form to display whether or not username is taken

The easy part -- server side

- We'll need a server-side script to check whether or not a username is taken. Let's write a username_check.php page which takes one argument, username, that we'll call like:
  - username_check.php?username=blah
- It'll look something like:

```
<?php

// connect to database somehow...

$sql = "SELECT  1 FROM users WHERE username='" . mysql_real_escape_string($_GET['username']) ."'
LIMIT 1";
$qry = mysql_query($sql);
if (mysql_num_rows($qry) > 0) {
    echo "TAKEN";
} else {
    echo "AVAILABLE";
}

?>
```

Adding an event to the text field

- Back on the client side, we need to add an event for when the user is done typing in their username. This event should be called when the field loses focus, so we use the .focusout() event handler in jQuery.
- So, our code might be like:

```
<input id='username' type='text' name='username' />
<div id='username_error_box'></div>
<script type='text/javascript'>
$("#username").focusout(function() {
    // ajax call will go here
});
</script>
```

Handling the AJAX request

- We know how to make an AJAX request, so let's do it:

```
$.get( 'username_check.php',
      { username: $("#username").val() },
      function(response) {
          if (response == "TAKEN") {
              $('#username_error_box').html("This username is taken.");
          }
       });
```

Security concerns

- You might have noticed that an AJAX request just loads a page on your website.
  - That is, anyone can load the same page, and supply their own arguments to make their own request.
  - Even if you could enforce only requests coming from your page, it's easy for malicious users to modify the JavaScript on your page.
- So, you shouldn't pass sensitive information via AJAX, unless you take proper precautions in encrypting it (and remember, all encryption has to be done server-side; you can't do it via JavaScript).
- Also, if you're using a login system, make sure to check the session for proper credentials before fulfilling an AJAX request. This is a common attack vector.

Summary

- AJAX is a technique that lets you make your pages seem more responsive and flashier.
- If you use AJAX to perform sensitive tasks (like deleting data), make sure your server-side pages can't be exploited.