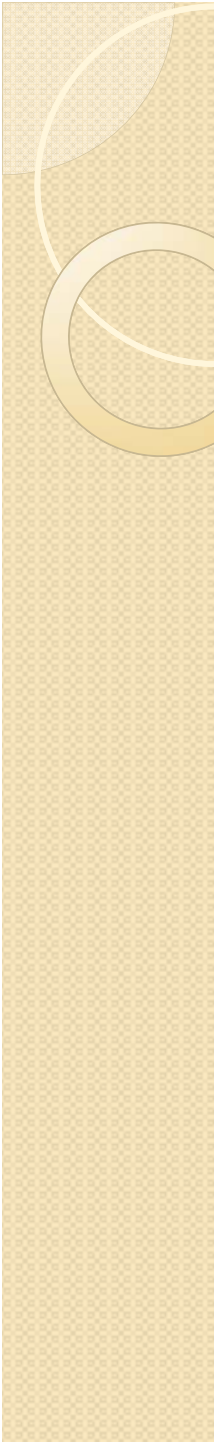




CGT 353 Lecture 2

Introduction to jQuery

Ronald J. Glotzbach



Introduction

Getting Started

- Start with a new .html that contains the following:

```
<?xml version="1.0" encoding="UTF-8" ?>
<!doctype html>
  <html xmlns="http://www.w3.org/1999/xhtml" xml:lang="en" lang="en">
    <head>
      <meta charset="utf-8">
      <title>Getting Started with jQuery</title>
    </head>
    <body>
      <a href="http://jquery.com/">jQuery</a>

      <!-- The script below points to your copy of jQuery -->
      <script src="jquery.js"></script>
      <script>

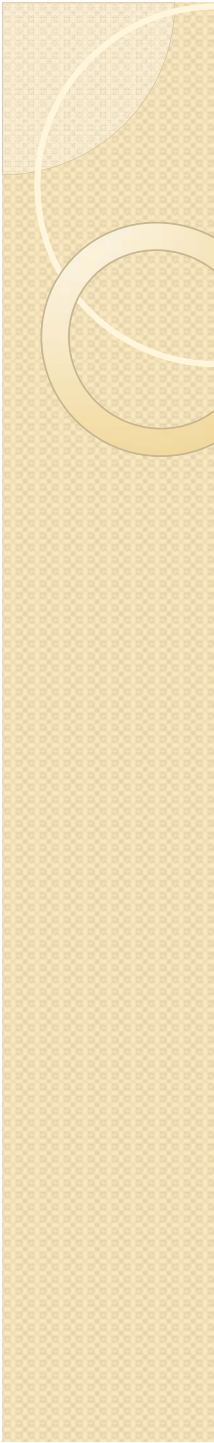
      </script>
    </body>
  </html>
```

Downloading jQuery

- jQuery can be downloaded from here:
 - http://docs.jquery.com/Downloading_jQuery
- You can download plugins developed by other contributors here:
 - <http://plugins.jquery.com/>
 - Don't do this to start... this is for after you have the hang of jQuery

Understanding jQuery syntax

- Basic syntax:
 - **`$(selector).action()`**
 - The dollar sign `$` defines it as being jQuery
 - The `(selector)` is used to query (or find) HTML elements on the page
 - The `.action()` is the jQuery action that will be performed on the element(s)



Case sensitivity

- Consider ***everything*** to be case sensitive and you'll be much better off.



BASIC SYNTAX

Examples of the syntax

- `$("#a").addClass("blue");`
 - Uses jQuery to add a CSS class named 'blue' to all `<a>` elements.
- `$("#a").removeClass("blue");`
 - Uses jQuery to remove the CSS class named 'blue' from all `<a>` elements.
- `$("#div").hide();`
 - Hides all `<div>` tags on the page
- `$("#div").hide("slow");`
 - Hides all `<div>` tags 'slowly' on the page

Examples of the syntax

- `$(this)`
 - Refers to the current HTML element
- `$("p")`
 - Refers to the paragraph element
 - Replace p with any HTML element
- `$("a.example")`
 - Refers to the `<a>` element with **class**="example"
- `$("#example")`
 - Refers to the element with **id**="example"
- `$("div#example")`
 - Refers to the `<div>` element with **id**="example"



ATTRIBUTE SELECTORS USING XPATH

Attribute Selectors using XPATH

- `$("[style]")`
 - Refers to all elements with a 'style' attribute
- `$("[style='text-align:center;']")`
 - Refers to all elements with **style="text-align:center;"**
- `$("[style!='text-align:center;']")`
 - Refers to all elements with a 'style' attribute NOT equal to **style="text-align:center;"**
 - Notice the use of the exclamation point !
- `$("[style*='text-align:center;']")`
 - Refers to all elements with a 'style' attribute that **contains text-align:center;** within it
 - Notice the use of the asterisk *

Attribute Selectors using XPATH

- `$("[src$='.jpg']")`
 - Refers to all elements with a 'src' attribute that **ends with .jpg**
 - Notice the use of the dollar sign \$
- `$("[name^='address']")`
 - Refers to all elements with a 'name' attribute (think form elements) that **starts with address**
 - Notice the use of the ^

More about attribute selectors

- Suppose you have
 - `<input name="address1" />`
 - `<input name="address2" />`
 - `<input name="saddle" />`
 - `<div name="addme" />`
- Then you have
 - `$(“input[name*='add']”).val(“has add in it!”);`
- Result
 - All 3 ***input*** elements will have the value set to: “has add in it!”



CSS SELECTORS

CSS Selectors

- `$(“div”).css(“background-color”,“#ff0000”);`
 - Sets the background-color of all `<div>` elements on the page to red
 - Use `#id` to identify an individual `<div>` by its id
- In your `<head>` element, you have:
 - `<style type="text/css">`
 `a.test { font-weight:bold; }`
 `</style>`
- Then
 - `$(“a”).addClass(“test”);`
 - Adds the CSS class ‘test’ to all `<a>` elements
 - `$(“a”).removeClass(“test”);`
 - Removes the CSS class ‘test’ from all `<a>` elements



READY EVENT

CGT 353 - Material Copyright © 2009-
Present Ronald J. Glotzbach, Purdue

1/24/2012 University

Ready event

- `$(document).ready(function(){
 //add code here
});`
- This is the code that will execute as soon as the document is loaded in the browser and ready to be worked on.
- Replaces `window.onload`

Ready event

- ```
$(document).ready(function(){
 $("a").addClass("test");
 $("div#wrapper").css("background-color", "#ff0000");
});
```
- When the page is finished loading in the browser, all `<a>` elements will have the CSS class 'test' applied to them and the `<div>` with id="wrapper" element will have a red background color applied to it.



# EVENT FUNCTIONS

# Event functions

- `$("#div#wrapper").click(function(event){  
 event.preventDefault();  
 $(this).hide("slow");  
});`
- When the `<div>` with `id="wrapper"` is clicked, prevent any default functionality and hide the `<div>`
- 'event' is the variable name we used as a parameter: `function(event)...` sometimes you might see `function(e)` instead, and then: `e.preventDefault()`;
- `hide()` is a built-in **effect** provided by jQuery
- This has to be launched from the ready event.

# Event functions

- Thus, the entire code would be:

```
<script type="text/javascript">
 $(document).ready(function(){
 $("div#wrapper").click(function(e){
 e.preventDefault();
 $(this).hide("slow");
 }); //end of click event
 }); //end of ready event
</script>
```



# **CALLBACK AND FUNCTIONS**

CGT 353 - Material Copyright © 2009-  
Present Ronald J. Glotzbach, Purdue

1/24/2012 University

# Callback

- A callback is a function that is passed as an argument to another function and is executed after its parent function has completed.
- `$.get("somePage.html", callbackFunction);`
  - First parameter is the URL to get
  - Second parameter is the name of the function to call after getting the URL
  - Notice that there are **not** `()` after the function

# Callback with arguments

- You're familiar with passing arguments (also called parameters) the traditional way
- `arg1 = "fu";`
- `arg2 = "bar";`
- `someFunction(arg1, arg2);`
  - The two parameters are passed into the function

# Callback with arguments

- You **cannot** follow that example in jQuery
- Here is the **WRONG** way to do it – this will **not** work
- `$.get("somePage.html", callbackFunction(arg1, arg2));`
  - *I make this mistake all the time*

# Callback with arguments

- Here is the **correct** way to pass arguments into the function that you want to call.
- `$.get("somePage.html", function(){  
    callbackFunction(arg1, arg2);  
});`



# EFFECTS

# .hide() effect

- .hide() is easy to use
- \$(this).hide("slow");
  - Hides the current element
- \$("#wrapper").hide();
  - Hides the element with id="wrapper"
- \$("p").hide("fast");
  - Hides all the <p> elements

# .animate() effect

```
<div id="clickme" style="width:300px; text-align:center; height:150px; background-color:#ccc;">
```

click here

```
</div>
```

```

```

```
<script type="text/javascript">
```

```
$(document).ready(function(){
```

```
 $("#clickme").click(function(){
```

```
 $("#pic").animate({
```

```
 opacity:0.6,
```

```
 left:"+=25",
```

```
 height:"toggle"
```

```
 }, 3000, function(){
```

```
 //animation complete
```

```
 });
```

```
 }); //end clickme
```

```
});
```

```
</script>
```

## .animate()

- On the previous slide, if you have an image with id="img1"
- Clicking the div will shrink the image over 3 seconds until it disappears
- Clicking the div again will expand the image over 3 seconds until it is full size again.
- Opacity and position will remain adjusted, however.

# .animate()

- **This is a particularly useful example of .animate(), relatively easy also**

```

```

```
<script type="text/javascript">
```

```
$(document).ready(function(){
```

```
 $('#pic2').animate({'opacity' : 1}).hover(function() {
```

```
 //on hover, change opacity to 50%
```

```
 $(this).animate({'opacity' : .5});
```

```
 }, function() {
```

```
 //on exit hover, change opacity back to 100%
```

```
 $(this).animate({'opacity' : 1});
```

```
 });
```

```
});
```

```
</script>
```