BIT116: Scripting
Lecture 05

Using JQuery

jQuery: The Basics

jquery03.html

```
<!DOCTYPE html>
<html>
<head>
<script src="http://ajax.googleapis.com/ajax/libs/jquery/1.10.2/jquery.min.js">
</script>
<script>
$(document).ready(function(){
  $("p").click(function(){
    $(this).hide();
  });
});
</script>
</head>
<body>
<p>If you click on me, I will disappear.</p>
<p>Click me away!</p>
<p>Click me too!</p>
</body>
</html>
```

Why Use A JavaScript Library?

Okay, first things first.

Writing JavaScript applications from scratch can be difficult, time-consuming, frustrating, and a real pain in the, ahem, britches. They often require a great deal of knowledge of working with the DOM, CSS, JavaScript, and often server resources.

But don't fret!

Because of open source JavaScript Libraries (or "Toolkits" or "Frameworks"), you can take advantage of JavaScript features and functionalities that are already pre-written and programmed—all you have to do is know enough code to be able to insert these into your web pages.

There are many JavaScript Library collections available, and most of them are free. For this course, we'll be looking at the **jQuery** JavaScript library, a freely available open-source set of utilities and controls that help you to quickly build interactive web applications, or at least add cool and useful effects to your web pages.

Why jQuery Instead of Some Other JavaScript Library?

Some of jQuery's strengths are, besides being ubiquitous:

- **Lightweight**: It's considerably smaller than many other JavaScript libraries that are out there, which means that pages/sites using it load more quickly.
- **Active Development Community**: If you have a question, you can ask it in any of the several jQuery forums that are available and get a fast response, or you can search the forum archives to see if it is a FAQ.
- **Plugin Architecture**: If you need a feature that isn't in jQuery, there's a good chance that someone's written it as a plugin. An additional benefit of plugins is that you're only adding them to your site when they're needed—that is, you don't have their added weight on every page.
- **Speed**: jQuery utilities are often a lot faster than those of it's competitors. For example, see the following speed test: http://mootools.net/slickspeed/
- **Ease of Use for Non-Geeks**: Because its selection queries are based on *CSS*, someone who isn't a full-time professional programmer can easily drop into jQuery, add some functionality to their site, and have it work the way they expect.

For all these reasons, jQuery has become one of the most popular JavaScript libraries available.

Overview

- Sample file that loads jQ, has button, paragraph output, handler for the click
- Break that down into each piece
  - Loading jQ – use a template for loading CDN w/ local backup
  - $(document).ready()
  - Selectors - Changing paragraph
  - Forms, buttons, Selectors – responding to a button click

Adding jQuery to Your Web Pages/Web Site

In many cases, your first decision as a web developer using jQuery is deciding whether to

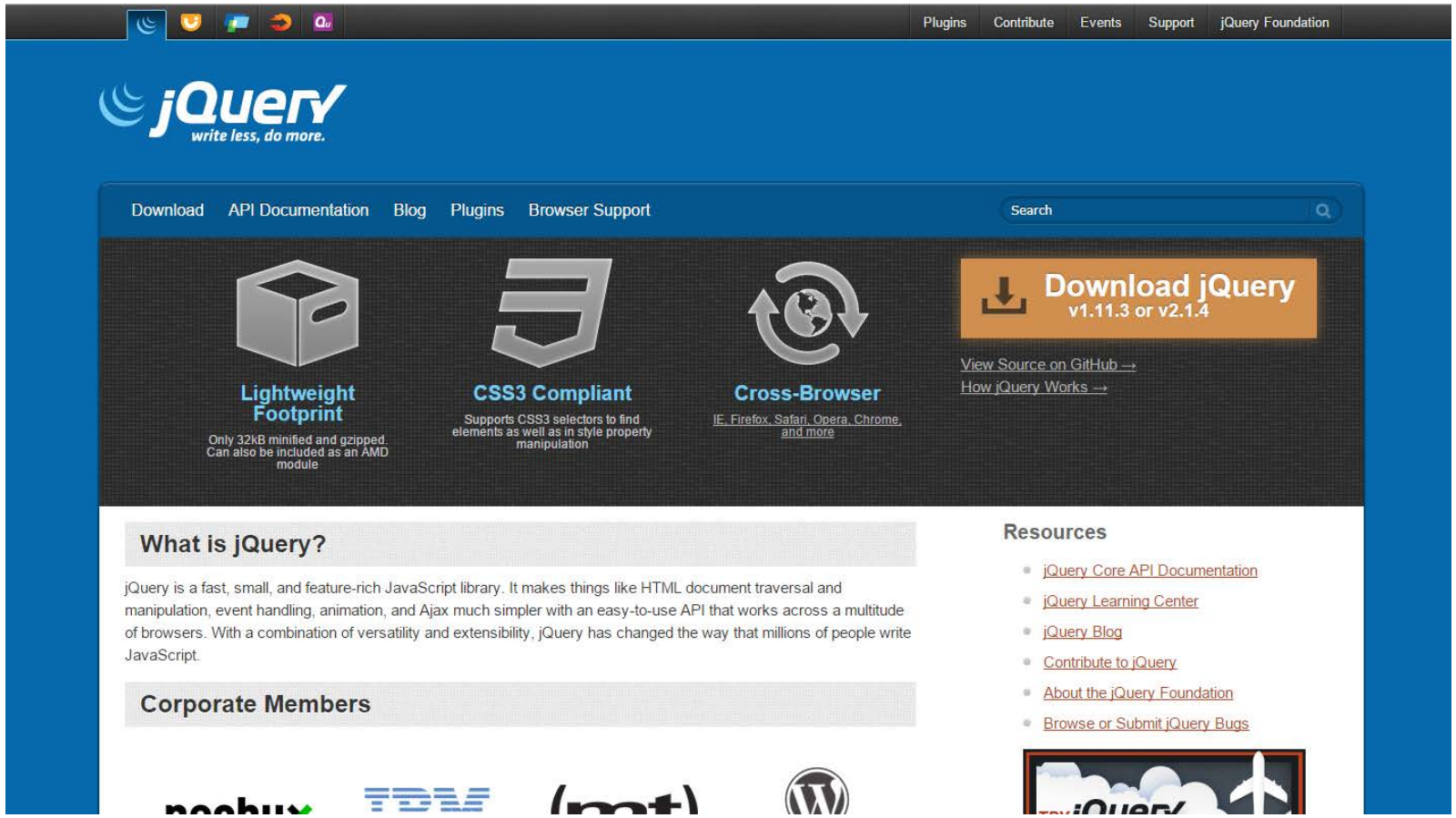
1. download the **jQuery core** to make it available from your web server

   or

2. use a hosted service (a **CDN -Content Delivery Network**).

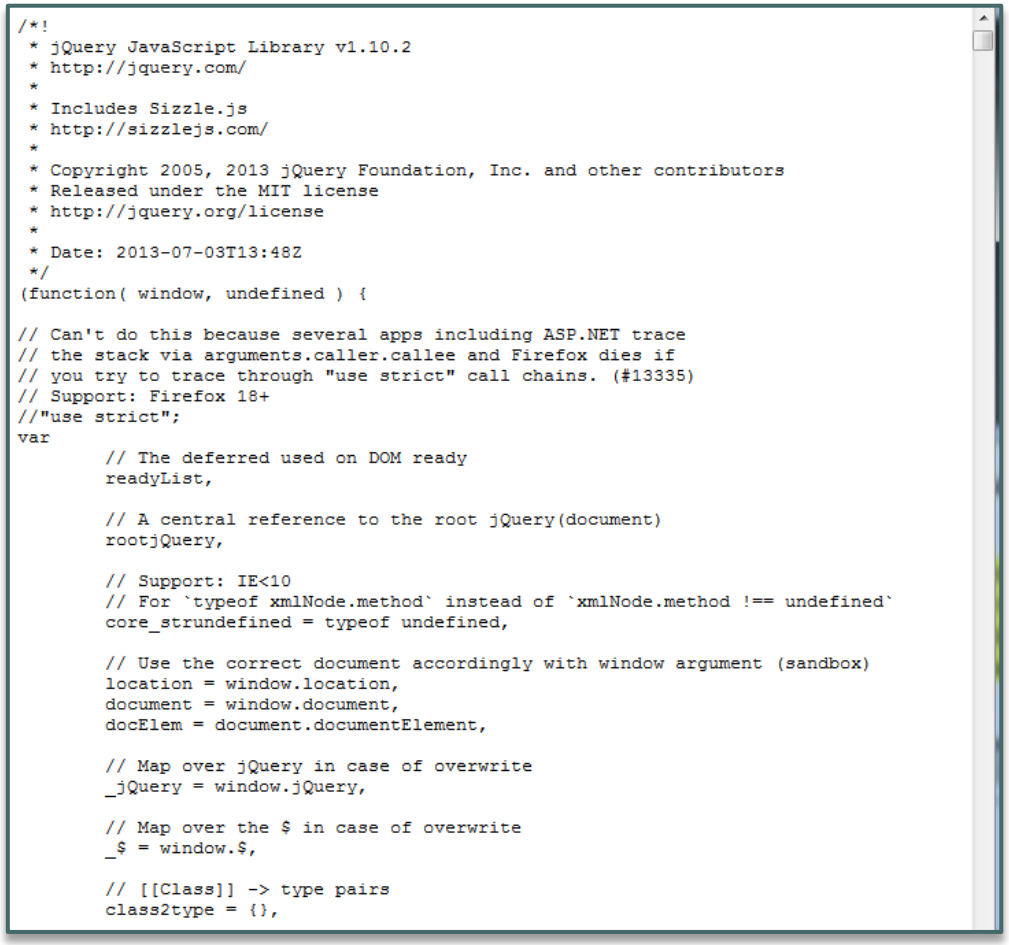
To Download jQuery from jquery.com

1. Open a browser and visit www.jquery.com and click the **Download jQuery** button.



To Download jQuery from jquery.com CONTINUED

2. Choose the version of jQuery you'd like to download, either **production** or **development**. The production version is compressed and **minified** (white spaces and comments stripped out) to provide the smallest possible footprint and overhead. For this class we'll be using the uncompressed **development** version, **3.2.1** when using a local file.
3. The jQuery file will appear in your browser
4. **Save** the file to your computer and then move it to the proper place on your web site (whether your local class or development folder or a "live" production web space).

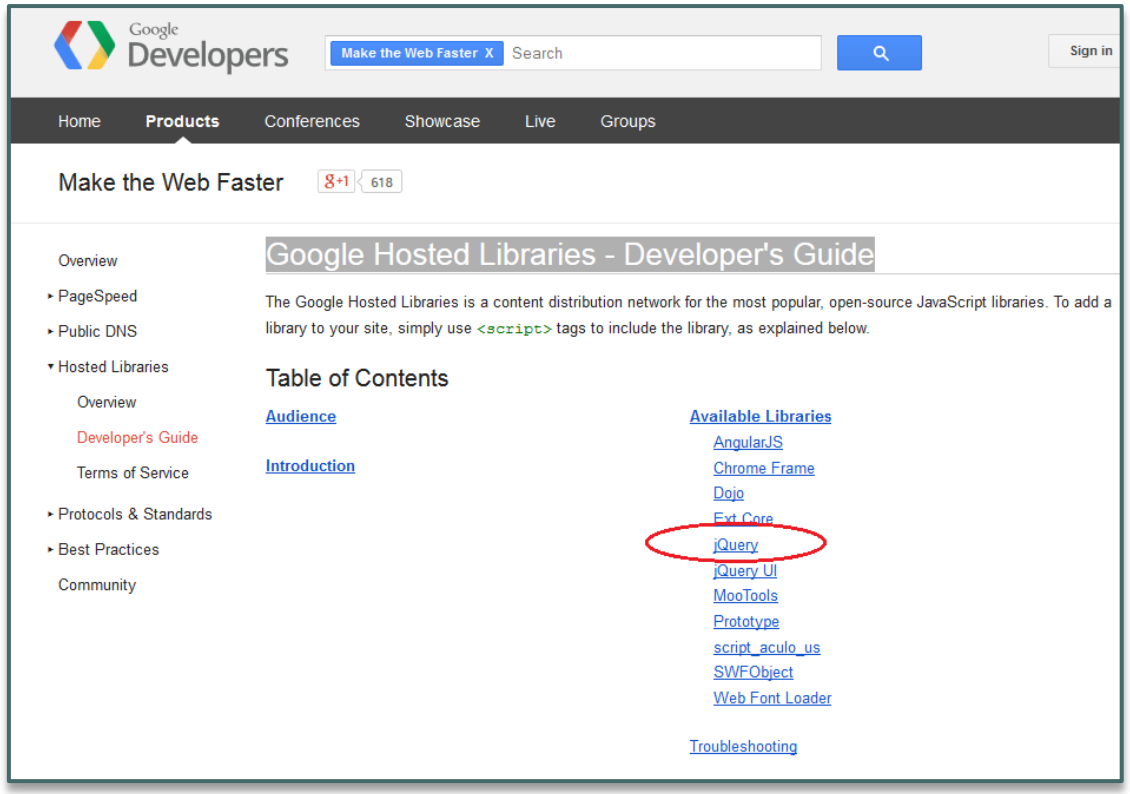


```
/*!
 * jQuery JavaScript Library v1.10.2
 * http://jquery.com/
 *
 * Includes Sizzle.js
 * http://sizzlejs.com/
 *
 * Copyright 2005, 2013 jQuery Foundation, Inc. and other contributors
 * Released under the MIT license
 * http://jquery.org/license
 *
 * Date: 2013-07-03T13:48Z
 */
(function( window, undefined ) {

// Can't do this because several apps including ASP.NET trace
// the stack via arguments.caller.callee and Firefox dies if
// you try to trace through "use strict" call chains. (#13335)
// Support: Firefox 18+
//"use strict";
var
        // The deferred used on DOM ready
        readyList,

        // A central reference to the root jQuery(document)
        rootjQuery,

        // Support: IE<10
        // For `typeof xmlNode.method` instead of `xmlNode.method !== undefined`
        core_strundefined = typeof undefined,

        // Use the correct document accordingly with window argument (sandbox)
        location = window.location,
        document = window.document,
        docElem = document.documentElement,

        // Map over jQuery in case of overwrite
        _jQuery = window.jQuery,

        // Map over the $ in case of overwrite
        _$ = window.$,

        // [[Class]] -> type pairs
        class2type = {},
```

To Add jQuery from a CDN (Content Delivery Network) like Google

1. **Google** a search for "**Google Host Libraries**" or you can access this direct link: https://developers.google.com/speed/libraries/devguide

2. From the "**Google Hosted Libraries - Developer's Guide**" page, select the **jQuery** link



To Add jQuery from a CDN (Content Delivery Network) like Google CONTINUED

3. Under **jQuery snippet**, highlight and copy the **<script>** tags that contain the path to the minified version of jQuery hosted by Google

4. **Paste** this into a file for later use, or into the head of an HTML file that you could use as a template.

FILE:jquery01.html

Setting Up a "Fallback" of Both Local and CDN Versions of jQuery

- Add the following code within the **<head></head>** tags in your web pages:

```
<script src="//ajax.googleapis.com/ajax/libs/jquery/3.2.1/jquery.min.js"></script>

<script type="text/javascript">
  if (typeof jQuery == 'undefined') {
  document.write(unescape("%3Cscript src='jquery-3.2.1.js' type='text/javascript'%3E%3C/script%3E")); }
</script>
```

The first script is an attempt to include the jQuery core in your web project from the CDN. The second script then tests to see if the jQuery object is available. If the jQuery object isn't available, a line is written to the HTML file to load the jQuery file from the local source (make sure the **src** path points to the actual directory/name of the local jQuery file and not simply the example jquery.js file name used here). *If you're using HTML5 it isn't necessary to include the type attribute in your script tags (as used in the code above).*

FYI: The **%3C** and **%3E** are **URL Encoded Characters**. **%3C** represents a '**<**' character and **%3E** represents a '**>**' character. The reason that **%3C** and **%3E** are used instead of '**<**' and '**>**' is because browsers may have a problem parsing '**<**' and '**>**' when included in a string inside **document.write()** but will know how to interpret the **%3C** and the **%3E** .

Include both CDN and local copy in this class

- For this class, use the pattern on the prior slide to first try to load jQuery from a CDN, and then to try and load it from your local backup if the CDN doesn't work.
- **➔ ➔ This is required for homework assignments. ➔ ➔**

FYI: "Minifying" Your Code

As you're developing your markup, style sheets, and jQuery code, you'll leave a lot of **whitespace**, **tabs**, and **comments** in your files to make the files easy to maintain and read. This is great during **development**, but for **production** you might consider minifying your code.

The process of minifying code removes all of the unnecessary whitespace, comments, and formatting characters, which makes the download of the code much faster. Some minifying applications will even check your code for errors and perform other actions to reduce your code to the smallest possible size without losing functionality.

Links to some online minifying/code compression tools:

- Google Closure Compiler (one of my favorites)
- Online JavaScript Compression Tool
- JavaScript Compressor (it can obfuscate too)

Exercise!

- Do exercise #1, in order to make sure that you can add jQuery to your file

jQuery: $(document).ready()

Getting Started with jQuery: document.ready()

Using jQuery's ready() Handler to Check the DOM

Because jQuery typically interacts with elements in your web pages, you'll want to make sure all of the elements are loaded into the browser before any of the jQuery methods are called.

To accomplish this, you'd use the **ready()** event handler to make sure all of the DOM elements are loaded.

There are a couple of ways to capture the event, and you can use either one of these methods—directly in your page's markup or in a separate file.

FILE: jquery02.html

Getting Started with jQuery: document.ready() CONTINUED

NOTE: for demonstrative purposes I will be including both JavaScript and jQuery in <script> tags in the web page's <head> because it is easier to see what is going on with the ids and classes in the <body>. Usually I would put all my JavaScript/jQuery code in separate .js files and then link to them in the <head>.

In either you're the **head** of your **HTML** page or a separate **.js** file attach the **ready()** handler to the document:

```
$(document).ready(function() {
    // your JavaScript/jQuery code goes here
     alert('Gadzooks! It  worked!');
});
```

NOTE: the first opening curly brace '{' (or "squiggle") should be on the same line as `function()` as shown here. Putting it on a separate line can produce irregular behavior in some browsers.

Getting Started with jQuery: document.ready() CONTINUED

Let's dissect this single line of code step-by-step to see what it is doing. Similar syntax will be seen with other **jQuery** code, so understanding it here will carry over there.

```
$(document).ready(function() {
    // your JavaScript/jQuery code goes here
     alert('Gadzooks! It  worked!');
});
```

The dollar sign '**$**' is the standard way to access **jQuery** (it's actually a shortcut for '**jQuery**'). It is the same as if you would have used the following (which also works and is completely interchangeable; in fact, if you are using another **library** that uses the dollar sign, then you can replace jQuery's dollar sign shortcut with the word '**jQuery**'):

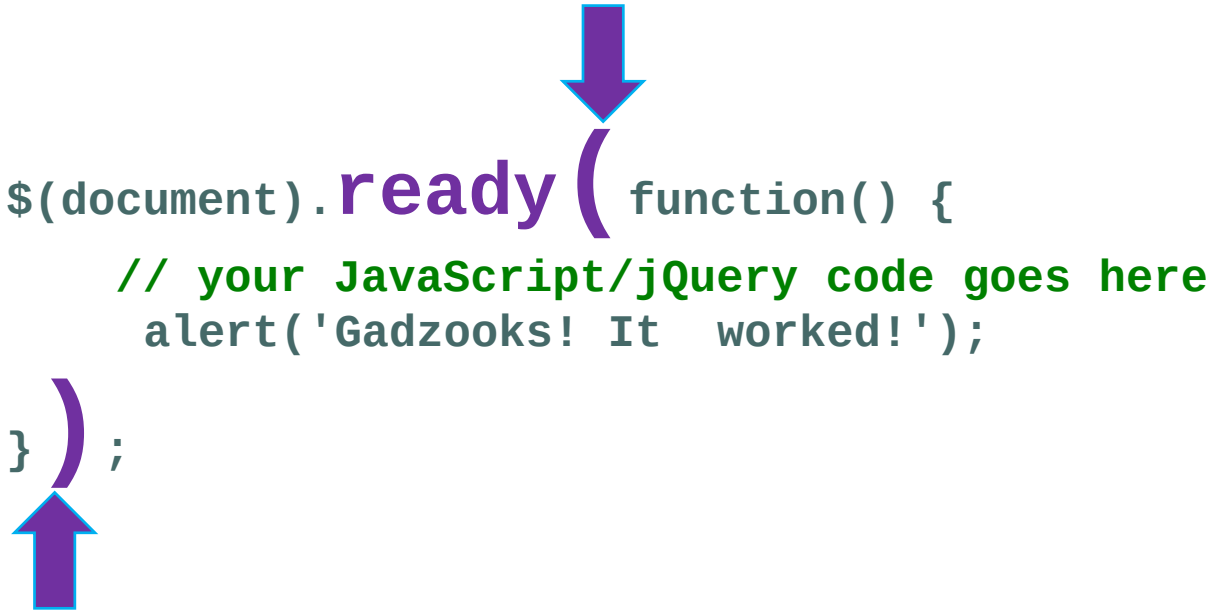
```
jQuery(document).ready(function() {
    // your JavaScript/jQuery code goes here
     alert('Gadzooks! It  worked!');
});
```

Getting Started with jQuery: document.ready() CONTINUED

```
$(document).ready(function() {
    // your JavaScript/jQuery code goes here
     alert('Gadzooks! It  worked!');
});
```

The `(document)` is a special case and represents an all-encompassing document object that points to the web page document loaded into the browser. Typically a selector would go here in between the two parenthesis (or **parens** in programmer-speak). This might be a **CSS3** selector and/or some of jQuery's improvements. This is basically telling jQuery to go find something, and in this case it is looking for the web page document.

Getting Started with jQuery: document.ready() CONTINUED



```
$(document).ready( function() {
    // your JavaScript/jQuery code goes here
     alert('Gadzooks! It  worked!');

} );
```

The `.ready()` is an event handler that checks to see whether the document—and the DOM—is properly loaded and ready before running any more jQuery, particularly the function() code inside of the .ready's parens.

Getting Started with jQuery: document.ready() CONTINUED

```
$(document).ready(function() {
    // your JavaScript/jQuery code goes here
     alert('Gadzooks! It  worked!');
});
```

The `(function()` is an inline "anonymous" function that hasn't been specifically named. It is a JavaScript shortcut that cuts down on code.

If we wanted to we could have created a named function outside of this, and called it instead of the "anonymous" function:

```
function someFunction(){
    // your JavaScript/jQuery code goes here
     alert('Gadzooks! It  worked!');
};

$(document).ready(someFunction);
```

Getting Started with jQuery: document.ready() CONTINUED

```
$(document).ready(function() {
    // your JavaScript/jQuery code goes here
    alert('Gadzooks! It  worked!');
});
```

The opening and closing curly braces (or "squiggles" as I like to call them) `{` and `}` are going to enclose the inner working code of the function.

The `)` at the end closes `.ready`'s parens, and the semicolon `;` at the very end closes the entire statement (just like we've seen in Java).

jQuery Selectors

Getting Started with jQuery: selecting an element

```
<script>
    $(document).ready(function() {
        // Since this page loads so fast
        // you probably won't see the original text
        $("#paraToChange").html("This is the <b>NEW</b> paragraph");
    });
</script>
</head>
<body>
<h2>jQuery 03 Example File</h2>
<p id="paraToChange">Some Fascinating Text!</p>
</body>
</html>
```

FILE: jquery03.html

Getting Started with jQuery: selecting an element

```
<script>
    $(document).ready(function() {
        // Since this page loads so fast
        // you probably won't see the original text
        $("#paraToChange").html("This is the <b>NEW</b> paragraph");
    });
</script>
</head>
<body>
<h2>jQuery 03 Example File</h2>
<p id="paraToChange">Some Fascinating Text!</p>
</body>
</html>
```

- This will use the jQuery function (`$(…)`) in order to find one or more things on the page
  - For right now we're going to select INDIVIDUAL items
- The jQuery function is given a string – note the red double quotes!
- The string contains a CSS selector for the element with the id - `#paraToChange`
- Note that we've assigned an id to the paragraph we want to change
  - `id="paraToChange"`, on the **p** element

FILE: jquery03.html

Getting Started with jQuery: selecting an element

```
<script>
    $(document).ready(function() {
        // Since this page loads so fast
        // you probably won't see the original text

        $("#paraToChange").html("This is the <b>NEW</b> paragraph");
    });
</script>
</head>
<body>
<h2>jQuery 03 Example File</h2>
<p id="paraToChange">Some Fascinating Text!</p>
</body>
</html>
```

- We then call the **html()** method to change the contents of the paragraph to be the new string
  - Don't forget the red dot!
  - Don't forget the dark-red double-quotes!

FILE: jquery03.html

Getting Started with jQuery: selecting an element

```
<script>
    $(document).ready(function() {
        // Since this page loads so fast
        // you probably won't see the original text
        $("#paraToChange").html("This is the <b>NEW</b> paragraph");
    });
</script>
</head>
<body>
<h2>jQuery 03 Example File</h2>
<p id="paraToChange">Some Fascinating Text!</p>
</body>
</html>
```

- Because we're using the .html method we can use HTML inside the string.
  - In this case everything outside the <b> element will be formatted 'normally' (`This is the … paragraph`)
  - Everything inside the <b> element will be bold (`<b>NEW</b>`)

FILE: jquery03.html

Getting Started with jQuery: Selectors

One of the strengths of the **jQuery** library is that it gives you the ability to easily select and interact with **Document Object Model (DOM)** elements in your web pages.

The **selectors** will be familiar to you if you've done any web development with CSS, because jQuery's selector syntax is nearly a duplicate of the selectors you'd use when preparing style properties for your websites.

jQuery makes use of the **Sizzle JavaScript CSS Selector Engine**, which was developed by John Resig, the creator of jQuery.

The jQuery library also adds several specific selector extensions to enhance the library's syntax and make the library easier to use.

In this section, we'll learn how to choose and use jQuery's selectors to gain control over DOM elements as well as groups of elements in your web pages. You'll also learn how to **combine selectors** and then apply **filters** to your selectors for greater flexibility and control.

Getting Started with jQuery: Selectors CONTINUED

Using Basic Selectors

Although jQuery provides a host of selector options, you'll find yourself using one of the basic selectors (element, id, or class) most of the time. Listed below are the basic selectors and what they do:

Selector Name	What It Does
`all $('*')`	Selects every element within a web page, beginning with the body element. The **all** selector is rarely used and can be slow if the web page contains a large number of elements.
`element $('element')`	Selects all of this element type, such as **div**, **p**, or **h1**.
`id $('#id_name')`	Selects the one element in a web page having the ID. The ID must be preceded with a hash or pound sign. (Each ID in a web page must be unique!)
`class $('.class_name')`	Selects all elements having a particular class. Class names are preceded by a period.

Getting Started with jQuery: Selectors CONTINUED

As you're creating the **HTML** markup for your web pages, you should plan **classes** and **ID**s carefully. Good planning will make your life much easier when you get ready to apply jQuery to your website. It's easy to get tripped up and wonder why your jQuery code isn't working when you don't keep your ids unique or you fail to define them properly.

In HTML5, most of the restrictions on IDs have been relaxed. You can use any character in an ID and it must contain at least one character, but each ID within a web page must be unique.

If you mistakenly have two elements with the same ID, you may get odd results when applying jQuery code to those elements and the results may differ from browser to browser.

When you wish to add jQuery code designed to have an effect on several or a group of elements, I recommend you use **classes** for those elements. First of all, it's usually easier to remember named groups, and second, you can apply jQuery methods to all sorts of elements that have the same class.

Exercises

- Do exercise #2

jQuery: Adding a Button

Getting Started with jQuery: Adding a button

```
<script>
    $(document).ready(function() {
        $("#theButton").click( function() {
            alert("button clicked!");
        });
    });
</script>
</head>
<body>
<h2>jQuery 03 Example File</h2>
<form>
<input type="button" value="Click here" id="theButton">
</form>
</body>
```

FILE: jquery04.html

Getting Started with jQuery: Adding A Button – first, the HTML

```
</script>
</head>
<body>
<h2>jQuery 03 Example File</h2>
<form>
<input type="button" value="Click here" id="theButton">
</form>
</body>
```

- We should put the button inside a form
  - This is the **<form> … </form>** stuff
- Note that the **<input** element does NOT need a closing tag
  - **type="button"** tells the browser to make it a button
  - **value="Click here"** tells the browser what to show on the button
  - **id="theButton"** tells the browser what id to give the button. DON'T FORGET THIS!!! ☺

FILE: jquery04.html

Getting Started with jQuery: Adding A Button – first, the HTML

```
<script>
    $(document).ready(function() {
        $("#theButton").click( function() {
            alert("button clicked!");
        });
    });
</script>
</head>
<body>
<h2>jQuery 03 Example File</h2>
<form>
<input type="button" value="Click here" id="theButton">
</form>
</body>
```

- When the document is ready we tell jQuery to find the button (using it's ID)
  - `$("#theButton")`
- When then tell jQuery to attach an event handler so respond to the button being clicked
  - `.click(…);`
- When the button is clicked, run the function that we define here:
  - `function() {… }`

FILE: jquery04.html

Getting Started with jQuery: Adding A Button – first, the HTML

```
<script>
    $(document).ready(function() {
        $("#theButton").click( function() {
            alert("button clicked!");
        });
    });
</script>
</head>
<body>
<h2>jQuery 03 Example File</h2>
<form>
<input type="button" value="Click here" id="theButton">
</form>
</body>
```

- When the button is finally clicked (and the anonymous function is run) pop up an alert box
  - **`alert("button clicked!");`**

FILE: jquery04.html

Exercises

- Do exercise #3

jQuery: Getting User Input From A Textbox

Getting Started with jQuery: Getting User Input From A Textbox

```
<script>
    $(document).ready(function() {
        $("#theButton").click( function() {
            var input = $("#input").val();
            alert( "You typed: " + input );
        });
    });
</script>
</head>
<body>
<h2>jQuery 03 Example File</h2>
<form>
<p>Please type your name here: <input type="text" id="input"></p>
<input type="button" value="Click here" id="theButton">
</form>
</body>
</html>
```

FILE: jquery05.html

Getting Started with jQuery: Getting User Input From A Textbox - HTML

```
<body>
<h2>jQuery 03 Example File</h2>
<form>
<p>Please type your name here: <input type="text" id="input"></p>
<input type="button" value="Click here" id="theButton">
</form>
</body>
</html>
```

- We should put the textbox inside a form
  - This is the `<form> … </form>` stuff
- Start by telling the user what to type
  - **`<p>Please type your name here:… </p>`**
- Note that the **`<input`** element does NOT need a closing tag
  - **`type="text"`** tells the browser to make it a button
  - **`id="input"`** tells the browser what id to give the button. DON'T FORGET THIS!!! ☺
- You can pre-load the text box with text, if you want, using the value= attribute
  - Ex:**`value="Type here"`**

FILE: jquery05.html

Getting Started with jQuery: Getting User Input From A Textbox

```
<script>
    $(document).ready(function() {
        $("#theButton").click( function() {
            var input = $("#input").val();
            alert( "You typed: " + input );
        });
    });
</script>
</head>
```

- When the button is finally clicked (and the anonymous function is run) we'll first get the value of the text box.
- First we'll create a new variable
  - **`var input…`**
- Then we'll get the current value in the text box:
  - **`$("#input").val();`**
  - Specifically, the **.val()** part will retrieve the current value
- Lastly we'll assign the value to the variable:
- **`var input = $("#input").val();`**

FILE: jquery05.html

Getting Started with jQuery: Getting User Input From A Textbox

```
<script>
    $(document).ready(function() {
        $("#theButton").click( function() {
            var input = $("#input").val();
            alert( "You typed: " + input );
        });
    });
</script>
</head>
```

- Finally, we'll pop up an alert containing whatever was in the text box
  - `alert( "You typed: " + input );`

FILE: jquery05.html

Exercises

- Do exercise #4

Putting it all together

Complete example #2

- jquery_complete_2.html, on the course home page

- Note that you should memorize this pattern:
  1. Loading jQuery
  2. Adding a button, and a paragraph for 'output'
  3. Setting up document.ready
  4. Setting up the click event handler

jQuery Library and Assorted Links

jQuery Library

- http://jquery.com/

Google jQuery CDN (Content Delivery Network)

- https://developers.google.com/speed/libraries/devguide#jquery

jQuery Links/Tutorials

- How jQuery Works
- jQuery Learning Center
- jQuery Tutorials (W3Schools)
- jQuery Examples (W3Schools)
- jQuery for Beginners (Impressive Webs)
- jQuery (Tutorials Point)
- jQuery (Code Academy)
- jQuery (jQuery-Tutorial.net)
- jQuery (MP3, Polymorphic Podcast)
- jQuery Fundamentals (jqfundamentals)
- jQuery for Absolute Beginners Video Tutorials (Lost in the Woods)
- LearningJQuery.com
- jQuerify (Bookmarklet)
- SelectorGadget (CSS → jQuery)
- Firebug and jQuery (Video Screencast)