DURHAM COLLEGE
SUCCESS MATTERS

WEBD 6201 – Client Side Scripting

Intro to jQuery

Last Week…

We covered:

- Test 1 – DOM Manipulation and Event Handling

Lesson Objectives

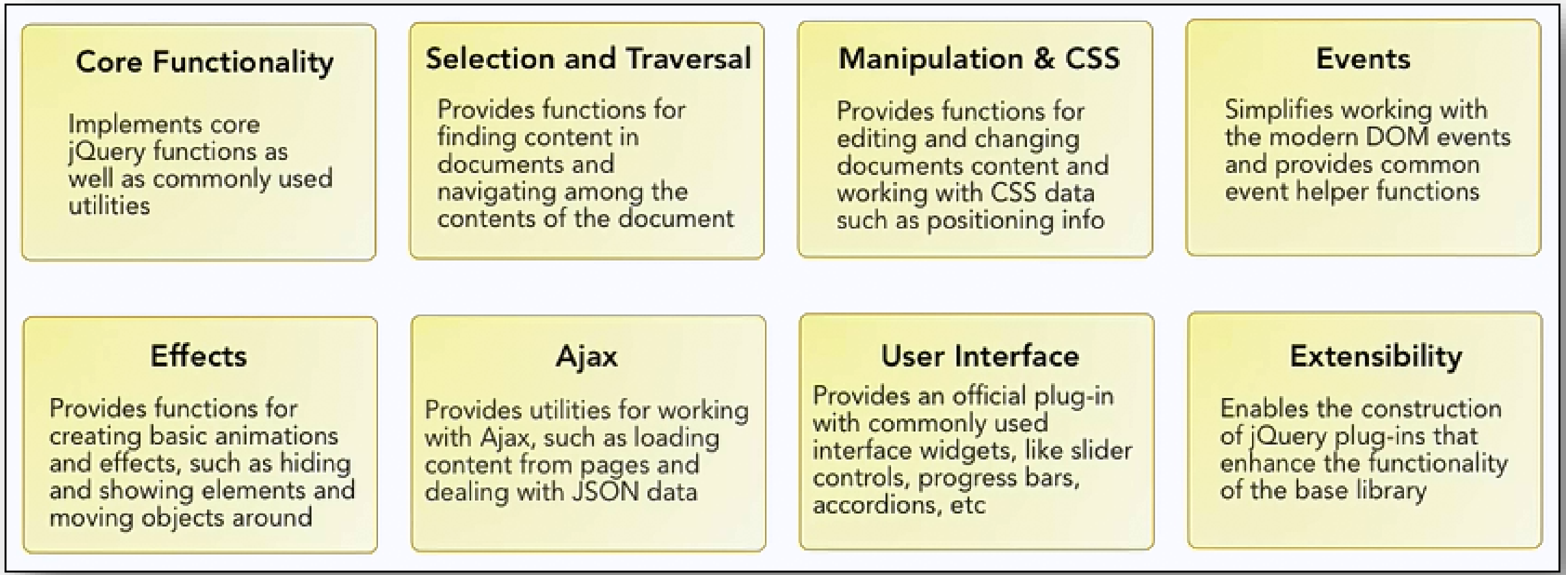
- Select elements using jQuery syntax
- Use built-in jQuery functions

What jQuery is

- **jQuery** is a free, open-source, **JavaScript library** that provides dozens of methods for common web features that make JavaScript programming easier.
- Beyond that, the jQuery functions are coded and tested for cross-browser compatibility, so they will work in all browsers.
- Those are two of the reasons why jQuery is used by more than half of the 10,000 most-visited web sites today, and that's why jQuery is commonly used by professional web developers.
- In fact, you can think of jQuery as one of the four technologies that every web developer should know how to use: **HTML**, **CSS**, **JavaScript**, and **jQuery**.
- But don't forget that **jQuery is actually JavaScript**.

Overview of jQuery Features

- jQuery's features break down across eight major categories:



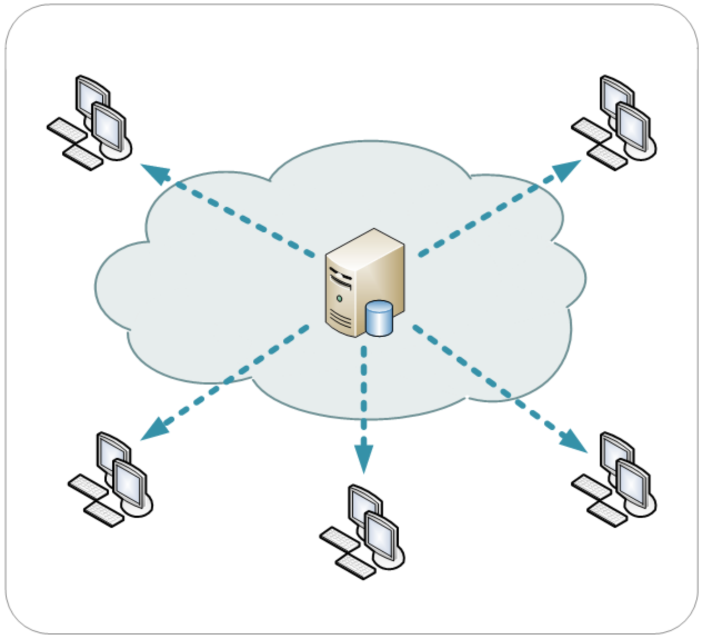
Core Functionality Implements core jQuery functions as well as commonly used utilities	**Selection and Traversal** Provides functions for finding content in documents and navigating among the contents of the document	**Manipulation & CSS** Provides functions for editing and changing documents content and working with CSS data such as positioning info	**Events** Simplifies working with the modern DOM events and provides common event helper functions
Effects Provides functions for creating basic animations and effects, such as hiding and showing elements and moving objects around	**Ajax** Provides utilities for working with Ajax, such as loading content from pages and dealing with JSON data	**User Interface** Provides an official plug-in with commonly used interface widgets, like slider controls, progress bars, accordions, etc	**Extensibility** Enables the construction of jQuery plug-ins that enhance the functionality of the base library

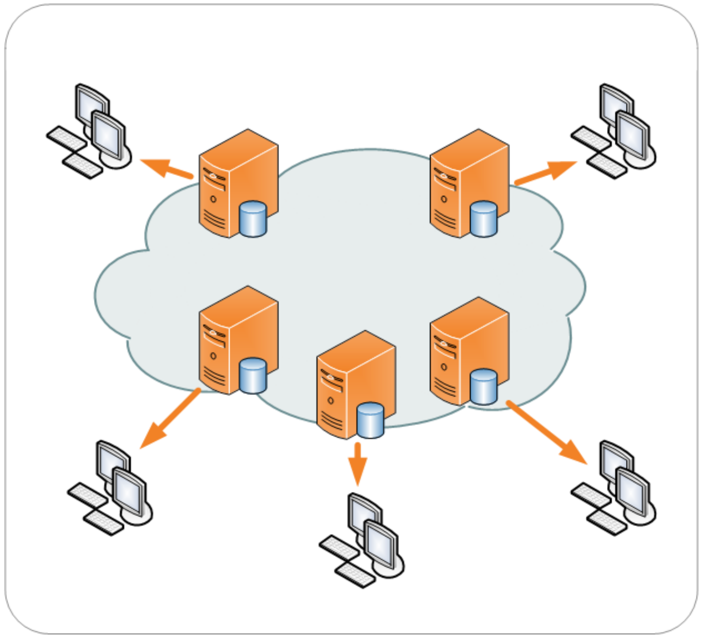
How to include jQuery in your web pages

- The jQuery download comes in two versions:
  - One version (min) is a compressed version that is relatively small and loads fast.
  - The other version is uncompressed so you can use it to study the JavaScript code in the library.
- If you include the jQuery file from a ***Content Delivery Network*** *(**CDN**),* you don't have to provide it from your own server, but then you can't work offline.
- The **jQuery CDN** now provides a link that will always deliver the latest version of jQuery.
- If you download the jQuery file to your system, you can change the filename so it's simpler, but then you may lose track of what version you're using.

Content delivery network (Wikipedia)

- A content delivery network (CDN) is a large distributed system of servers deployed in multiple data centers in the Internet.
- The goal of a CDN is to serve content to end-users with high availability and high performance.
- CDNs serve a large fraction of the Internet content today, including web objects (text, graphics, URLs and scripts), downloadable objects (media files, software, documents), applications (e-commerce, portals), live streaming media, on-demand streaming media, and social networks.





Using jQuery with a CDN

- **CDNs** can offer a performance benefit by hosting jQuery on servers spread across the globe.
- This also offers an advantage that if the visitor to your webpage has already downloaded a copy of jQuery from the same CDN, it won't have to be re-downloaded.
- Three main CDN providers:
  - **Using jQuery's CDN**

    ```
    <script src="http://code.jquery.com/jquery-1.9.0.min.js"></script>
    ```
  - **Using Google's CDN**

    ```
    <script src="https://ajax.googleapis.com/ajax/libs/jquery/1.9.0/jquery.min.js"></script>
    ```
  - **Using Microsoft's CDN**

    ```
    <script src="http://ajax.aspnetcdn.com/ajax/jQuery/jquery-1.9.0.min.js"></script>
    ```

How jQuery UI and plugins can simplify JavaScript development

- **jQuery UI** is a free, open-source, JavaScript library that provides higher-level effects, widgets, and mouse interactions that can be customized by using themes.
- A **plugin** is a JavaScript library that provides functions that work in conjunction with jQuery to make it easier to add features to your web applications.
- In general, if you can find a plugin or jQuery UI feature that does what you want it to do, you should use it.
- Often, though, you won't be able to find what you want so you'll need to develop the feature with just the core jQuery library.

Some typical plugin functions

- Data validation
- Slide shows
- Carousels

CENTENNIAL COLLEGE

The basics of jQuery programming

jQuery Selectors and Filters: Overview

- jQuery selectors and filters retrieve content from the document so it can be manipulated using other functions (think of this as the Query part of the jQuery)
  - jQuery **selectors** return an array of objects that match the selection criteria
  - jQuery **filters** operate on a selector to further refine the results array that the selector returns.
- The array that is returned is not a set of DOM elements.
  - It is a collection of jQuery objects that provide a large number of predefined functions for further operating on the objects

Using basic selectors

❖ CSS-style selectors and filters are based on familiar CSS syntax, and work pretty much the same way as CSS does

The CSS selectors listed here correspond directly to their CSS counterparts

Selector	Purpose
`tagname`	Finds all elements that are named `tagname`.
`#identifier`	Finds all elements with ID of `identifier`.
`.className`	Finds all elements that have class attribute with the value of `className`.
`tag.className`	Gets elements of type `tag` that have a class attribute with the value of `className`.
`tag#id.className`	Retrieves the tag element that has an ID of `id` and a class attribute with the value of `className`.
`*`	Finds all elements on the page.

How to code jQuery selectors

- When you use **jQuery**, the dollar sign (**$**) is used to refer to the **jQuery library**.
- Then, you can code selectors by using the **CSS syntax** within quotation marks within parentheses.

The syntax for a jQuery selector

```
$("selector")
```

How to select elements by element, id, and class

By element type: All <p> elements in the entire document

```
$("p")
```

By id: The element with "faqs" as its id

```
$("#faqs")
```

By class: All elements with "plus" as a class

```
$(".plus")
```

How to select elements by relationship

Descendants: All <p> elements that are descendants of the section element

```
$("#faqs p")
```

Adjacent siblings: All div elements that are adjacent siblings of h2 elements

```
$ ("h2 + div")
```

General siblings: All <P> elements that are siblings of ul elements

```
$("ul ~ p")
```

Children: All ul elements that are children of div elements

```
$("div > ul")
```

How to call jQuery methods

- To call a **jQuery** method, you code a **selector**, the **dot operator**, the **method name**, and **any parameters** within parentheses. Then, that method is applied to the element or elements that are selected by the selector.
- When you use **object chaining** with jQuery, you code one method after the other.
- This works because each method returns the appropriate object.
- If the selector for a method selects more than one element, jQuery applies the method to all of the elements so you don't have to code a loop to do that.

The syntax for calling a jQuery method

```
$("selector").methodName(parameters)
```

Some common jQuery methods

Method	Description
`val()`	Get the value of a text box or other form control.
`val(value)`	Set the value of a text box or other form control.
`text()`	Get the text of an element.
`text(value)`	Set the text of an element.
`next([type])`	Get the next sibling of an element or the next sibling of a specified type if the parameter is coded.
`submit()`	Submit the selected form.
`focus()`	Move the focus to the selected form control or link.

Some jQuery examples

How to get the value from a text box

```
var gallon = $("#gallons").val();
```

How to set the value for an input element

```
$("#gallons").val("");
```

How to set the text in an element

```
$("#email_address_error").text(“Email address is
required");
```

How to set the text for the next sibling with object chaining

```
$("#last_name").next().text("Last name is required");
```

How to submit a form

```
$("#join_list").submit();
```

How to move the focus to a form control or link

```
$("#email_address").focus();
```

How to use jQuery event methods

- To code a jQuery **event handler**, you code a **selector**, the **dot operator**, the **name** of the jQuery **event method**, and an **anonymous function** that handles the event within parentheses.
- The **event handler** for the ready event will run any methods that it contains as soon as the **DOM** is ready, even if the browser is loading images and other content for the page.
  - This works better than the JavaScript **onload event**, which doesn't occur until all of the content for the page is loaded.
- When coding one event handler within another, the use of the closing braces, parentheses, and semicolons is **critical**.
  - To help get this right, many programmers code inline comments after these punctuation marks to identify the ends of the handlers.

The syntax for a jQuery event method

```
$(selector).eventMethodName(function() {
  //the statements of the event handler
});
```

Two common jQuery event methods

Event Method	Description
`ready(handler)`	The event handler runs when the DOM is ready.
`click(handler)`	The event handler runs when the selected element is clicked.

Two ways to code an event handler for the jQuery ready event

The long way

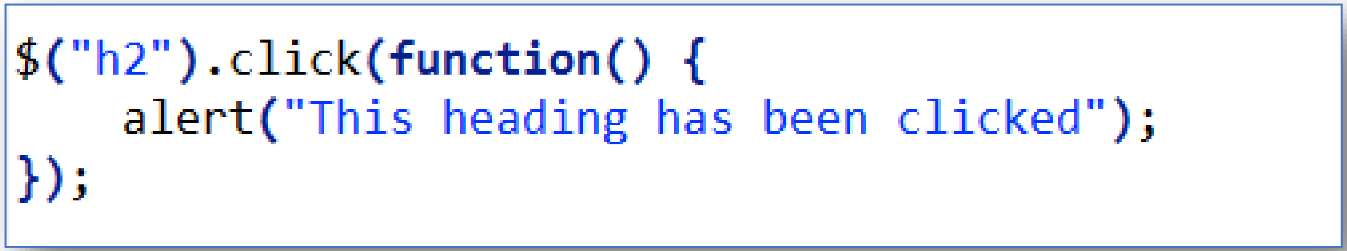
```
$(document).ready(function() {
    alert("The DOM is ready");
});
```

The short way

```
$(function() {
    alert("The DOM is ready");
});
```

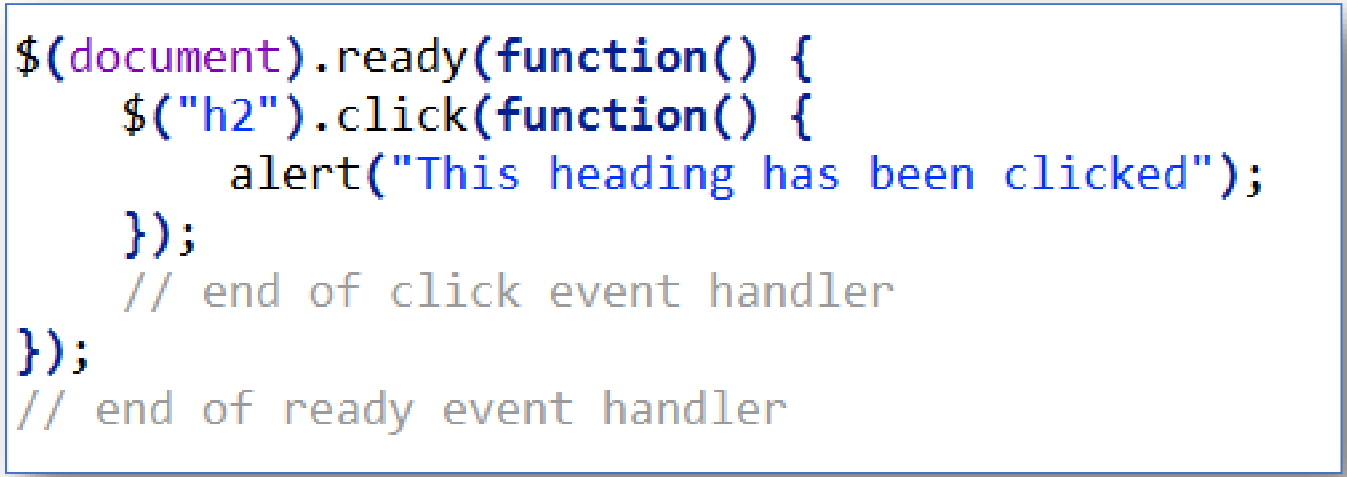
A couple of event handler examples

An event handler for the click event of all h2 elements



```
$("h2").click(function() {
    alert("This heading has been clicked");
});
```

The click event handler within the ready event handler



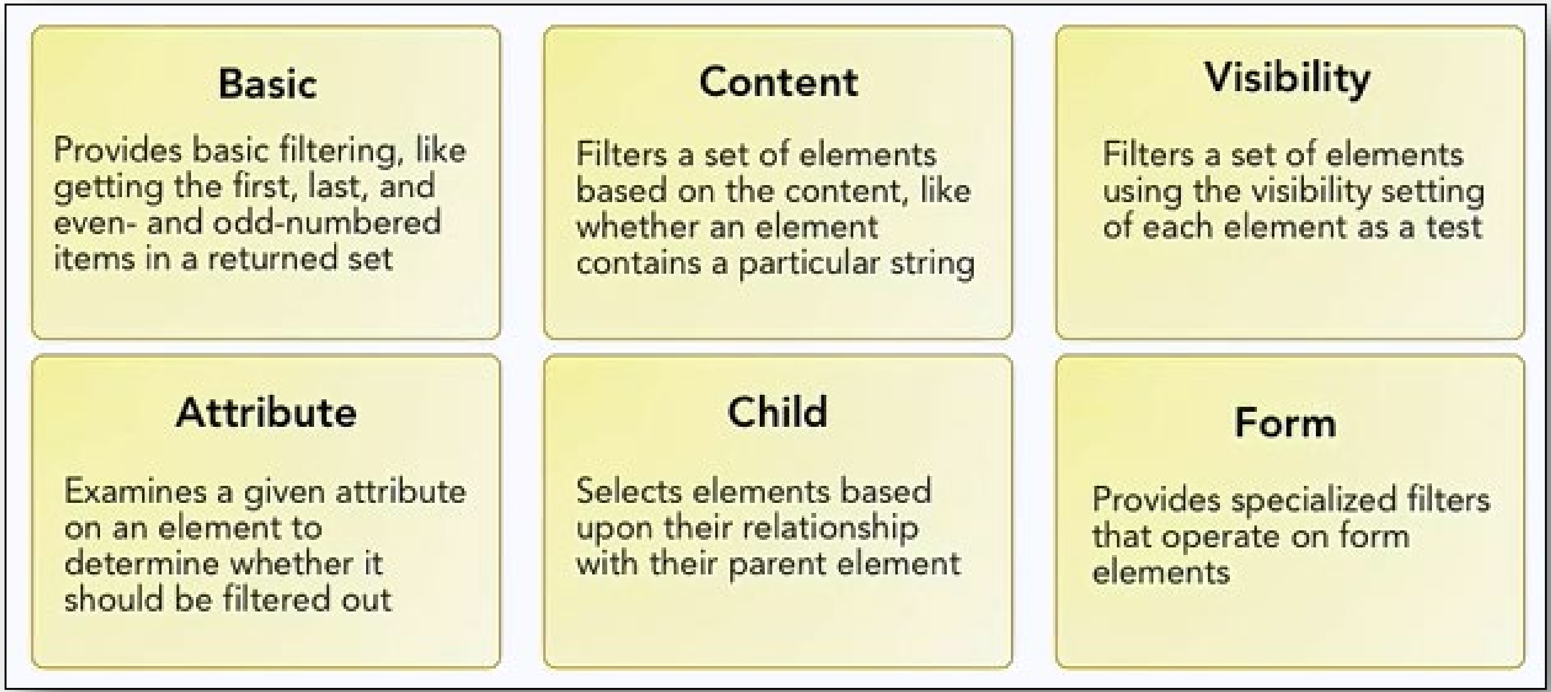
```
$(document).ready(function() {
    $("h2").click(function() {
        alert("This heading has been clicked");
    });
    // end of click event handler
});
// end of ready event handler
```

DURHAM COLLEGE SUCCESS MATTERS

Using basic jQuery filters

- Filters work selectors to provide even more fine-grained control over how elements are selected in the document

jQuery filters fall into six different categories



Basic Provides basic filtering, like getting the first, last, and even- and odd-numbered items in a returned set	**Content** Filters a set of elements based on the content, like whether an element contains a particular string	**Visibility** Filters a set of elements using the visibility setting of each element as a test
Attribute Examines a given attribute on an element to determine whether it should be filtered out	**Child** Selects elements based upon their relationship with their parent element	**Form** Provides specialized filters that operate on form elements

The most useful filters

Selector	Selects
`[attribute]`	All elements with the named attribute.
`[attribute=value]`	All elements with the named attribute and value.
`:contains(text)`	All elements that contain the specified text.
`:empty`	All elements with no children including text nodes.
`:eq(n)`	The element at index n within the selected set.
`:even`	All elements with an even index within the selected set.
`:first`	The first element within the set.
`:first-child`	All elements that are first children of their parent elements.
`:gt(n)`	All elements within the selected set that have an index greater than n.
`:has(selector)`	All elements that contain the element specified by the selector.
`:header`	All elements that are headers (hl, h2, ...).
`:hidden`	All elements that are hidden.
`:last`	All elements that are the last children of their parent elements.
`:lt(n)`	All elements within the selected set that have an index less than n.
`:not(selector)`	All elements that aren't selected by the selector.
`:nth-child`	All elements that are the nth children of their parent elements.
`:odd`	All elements with an odd index within the selected set.
`:only-child`	All elements that are the only children of their parent elements.
`:parent`	All elements that are parents of other elements, including text nodes.
`:text`	All input elements with the type attribute set to "text".
`:visible`	All elements that are visible.

jQuery filter Examples

How to select the li elements that are the first child of their parent element

```
$("li:first-child")
```

How to select the even tr elements of a table

```
$("table > tr:even")
```

How to select the third descendant <p> element of an element

```
$("#faqs p:eq(2)")
```

How to select all input elements with "text" as the type attribute

```
$(":text")
```

The most useful methods

Method	Description
`next([selector])`	Get the next sibling of an element or the first sibling of a specified type if the parameter is coded.
`prev([selector])`	Get the previous sibling of an element or the previous sibling of a specified type if the parameter is coded.
`attr(attributeName)`	Get the value of the specified attribute.
`attr(attributeName,value)`	Set the value of the specified attribute.
`css(propertyName)`	Get the value of the specified property.
`css(propertyName,value)`	Set the value of the specified property.
`addClass(className)`	Add one or more classes to an element and, if necessary, create the class. If you use more than one class as the parameter, separate them with spaces.
`removeClass([className])`	Remove one or more classes. If you use more than one class as the parameter, separate them with spaces.
`toggleClass([className])`	If the class is present, remove it. Otherwise, add it.
`hide([duration])`	Hide the selected element. The duration parameter can be "slow", "fast", or a number giving the time in milliseconds. By default, the duration is 400 milliseconds, "slow" is 600 milliseconds, and "fast" is 200 milliseconds.
`show([duration])`	Show the selected element. The duration parameter is the same as for the hide method.
`each(function)`	Run the function for each element in an array.

DURHAM COLLEGE SUCCESS MATTERS

jQuery method examples

Get the value of the src attribute of an image

```
$("#image").attr("src");
```

Set the value of the src attribute of an image to the value of a variable

```
$("#image").attr("src", imageSource);
```

Set the value of the color property of the h2 elements to blue

```
$("h2").css("color", "blue");
```

Add a class to the h2 descendants of the "faqs" element

```
$("#faqs h2").addClass ("minus");
```

Run a function for each <a> element within an "image_list" element

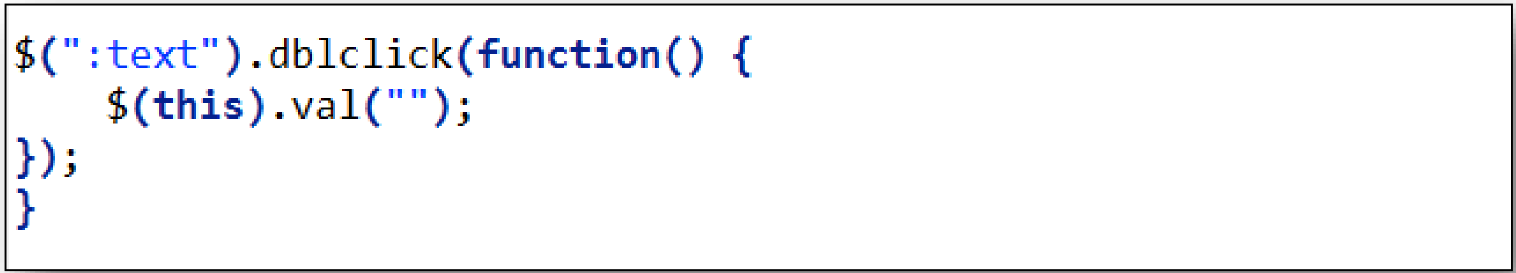
```
$("#image_list a").each(function(){
  // the statements of the function
});
```

The most useful event methods

Method	Description
`ready(handler)`	The handler runs when the DOM is ready.
`unload(handler)`	The handler runs when the user closes the browser window.
`error(handler)`	The handler runs when a JavaScript error occurs.
`click(handler)`	The handler runs when the selected element is clicked.
`toggle(handlerEven, handlerOdd)`	The first handler runs on even clicks of the element, starting with 0. The second handler runs on odd clicks of the element.
`dblclick(handler)`	The handler runs when the selected element is doubleclicked.
`mouseenter(handler)`	The handler runs when the mouse pointer enters the selected element.
`mouseover(handler)`	The handler runs when the mouse pointer moves over the selected element.
`mouseout(handler)`	The handler runs when the mouse pointer moves out of the selected element.
`hover(handler)`	The first event handler runs when the mouse pointer moves into an element. The second event handler runs when the mouse pointer moves out.

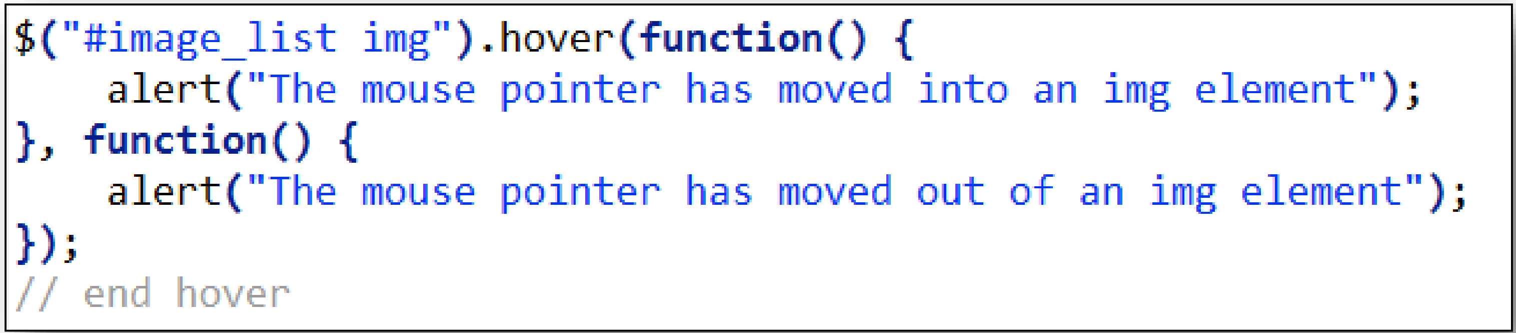
jQuery event method examples

A handler for the double-click event of all text boxes that clears the clicked box



```
$(":text").dblclick(function() {
    $(this).val("");
});
}
```

A handler for the hover event of each img element within a list



```
$("#image_list img").hover(function() {
    alert("The mouse pointer has moved into an img element");
}, function() {
    alert("The mouse pointer has moved out of an img element");
});
// end hover
```

Other event methods that you should be aware of

- When you store an event handler in a variable, you can use the **bind method** to attach it to more than one event.
- When you use the shortcut method to attach an event handler to an event, you're actually using the bind method.
- The **submit** and **focus** methods are actually event methods. When you use the shortcut method to trigger them, they trigger the submit and focus events.

Event Method	Description
`bind(event, handler)`	Attach an event handler to an event.
`unbind(event, handler)`	Remove an event handler from an event.
`one(event, handler)`	Attach an event handler and remove it after it runs one time.
`trigger(event)`	Trigger the event for the selected element.

Event Handler tips

How to store an event handler in a variable

```
var clearClick = function() {
    // the statements for the event handler
}
```

How to attach an event handler to an event

With the bind method

```
$("#clear").bind(click, clearClick);
```

With the shortcut method

```
$("#clear").click(clearClick);
```

How to attach an event handler to two different events

```
$("#clear").click(clearClick);
$(":text").dblclick(clearClick);
```

Event Handler tips (continued)

How to remove an event handler from an event

```
$("#clear").unbind("click", clearClick);
```

How to attach and remove an event handler so it runs only once

```
$("#clear").one("click", confirmClick);
```

How to trigger an event

With the trigger method

```
$("#clear").trigger("click");
```

With the shortcut method

```
$("#clear").click();
```

How to use effects and animations

The jQuery methods for effects

- The **duration** parameter can be "slow", "fast", or a number giving the time in milliseconds.
- By default, the duration is 400 milliseconds, "slow" is 600 milliseconds, and "fast" is 200 milliseconds.
- The callback parameter is for a function that is called after the effect has finished.
- If more than one element is selected, the callback junction is run once for each element.
- **Chaining** is commonly used with effects. This works because each effect method returns the object that it performed the effect on.

The basic syntax for all of the methods except the fadeTo method

methodName ([duration] [, *callback]*)

The basic syntax for the fadeTo method

```
fadeTo(duration, opacity[, callback])
```

The basic methods for jQuery effects

Method	Description
`show()`	Display the selected elements from the upper left to the lower right.
`hide()`	Hide the selected elements from the lower right to the upper left.
`toggle()`	Display or hide the selected elements.
`slideDown()`	Display the selected elements with a sliding motion.
`slideUp()`	Hide the selected elements with a sliding motion.
`slideToggle()`	Display or hide the selected elements with a sliding motion.
`fadeIn()`	Display the selected elements by fading them in to opaque.
`fadeOut()`	Hide the selected elements by fading them out to transparent.
`fadeToggle()`	Display or hide the selected elements by fading them in or out.
`fadeTo()`	Adjust the opacity property of the selected elements to the opacity set by the second parameter. With this method, the duration parameter must be specified.

jQuery effects examples

HTML for a heading that is animated after the web page is loaded

```
<h1 id="startup_message">Temporarily under construction!</h1>
```

jQuery that fades the heading out over 5 seconds

```
$("#startup_message").fadeOut(5000);
```

jQuery that uses chaining to fade the heading out and slide it back down

```
$("#startup_message").fadeOut(5000).slideDown(1000);
```

Chaining with fadeTo methods

```
$("#startup_message").fadeTo(5000, .2).fadeTo(1000, 1);
```

jQuery with a callback function that gets the same result as the chaining

```
$("#startup_message").fadeTo(5000, .2,
    function() { // start callback fUDction
    $(this).fadeTo(1000, 1);
} // end callback function
);
```

How to use the basic syntax of the animate method

- We refer to custom effects as **animation.**
- When the animate method is run, the CSS properties for the selected elements are changed to the properties in the properties map that is coded as the **first parameter** of the method. The animation is done in a **phased transition** based on the **duration** parameter.
  - To specify a **property name** in the properties map, you can use camel casing instead of the CSS hyphenation or you can enclose the property name in quotation marks.
  - To specify a non-numeric property value, enclose the value in quotation marks.
  - To specify a numeric property value, just code the value. For measurements, pixels are assumed. You can also use the+= and-= operators with numeric values, but these expressions must be enclosed in quotation marks.
- For some properties, like width, height, and opacity, you can use "show", "hide", or "toggle" as the property values. These will show, hide, or toggle the element by setting the property appropriately.
- Although color transitions aren't handled properly by jQuery, they are by jQuery UI.

The basic syntax for the animate method

```
animate ( {properties}) [, duration] [, callback])
```

Animation Basics Examples

The CSS for the h1 heading

```css
#faqs h1 {
    position: relative;
    left: -175px;
    font-size: 75%;
    opacity: 0.2;
}
```

An animate method for the h1 heading without a callback function

```javascript
$("#faqs h1").animate({
    fontSize : "275%",
    opacity : 1,
    left : 0
}, // the properties map
2000);
// end animate
```

An animate method for the h1 heading with a callback function

```javascript
$("#fags h2").animate({
    fontSize : "275%",
    opacity : 1,
    left : "+=175"
}, 2000, function() {
    $("#faqs h2").next().fadeIn(1000).fadeOut(1000);
});
// end animate
```

How to chain animate methods

- When you chain the effects and animations for an element, they are placed in a ***queue*** for that element and run in sequence, not at the same time.
- When separate effects and animations are started for an element, they are also placed in a queue for that element and run in sequence.
- When you use a **callback function** with an animate method, the callback function is run **after** the animation is finished.
- In some cases, a problem will occur if the user starts a second animation for an element before the callback function for the first animation has finished.

Animation Chaining examples

Chained animations

```
$("#faqs h1").click(function() {
    $(this).animate({
        fontSize : "650%",
        opacity : 1,
        left : "+=275"
    }, 2000).animate({
        fontSize : "175%",
        left : "-=275"
    }, 1000)
});
// end click
```

Queued animations

```
$("#faqs h1").click(function() {
    $(this).animate({
        fontSize : "650%",
        opacity : 1,
        left : "+=275"
    }, 2000);
    $(this).animate({
        fontSize : "175%",
        left : "-=275"
    }, 1000);
});
// end click
```

An animation with a second animation in its callback function

```
$("#faqs h1").click(function() {
    $(this).animate({
        fontSize : "650%",
        opacity : 1,
        left : "+=275"
    }, 2000, function() {
        $(this).animate({
            fontSize : "175%",
            left : "-=275"
        }, 1000)
    })
});
// end click
```

How to use the delay and stop methods

- The delay method in the example below works as an alternative to the use of a one-time timer.
- If you use the stop method as shown in the example above, you can stop the current animation for each <a> element and clear the queue. This will stop the bouncing effect that occurs when the user moves the mouse pointer rapidly back and forth over the thumbnails.

Method	Description
`delay(duration)`	Delay the start of the next animation in the queue.
`stop([clearQueue][,jumpToEnd])`	Stop the current animation for the selected element. The two parameters are Boolean with false default values. If set to true, the first parameter clears the queue so no additional animations are run. The second parameter causes the current animation to be completed immediately.

DURHAM COLLEGE
SUCCESS MATTERS

Delay and Stop examples

HTML for a heading that is displayed when the web page is loaded

```
<h1 id="startup_message">Temporarily under construction!</h1>
```

JQuery that fades the heading out after 5 seconds

```
$("#startup_message").delay(5000).fadeOut(1000);
```

The HTML for the thumbnail Images

```
<ul id="image_list">
  <li> <a href="images/h1.jpg" title="James Allison: 1-1"><img src="thumbnails/t1.jpg" alt=""></a> </li>
  <li> <a href="images/h2.jpg" title="James Allison: 1-2"><img src="thumbnails/t2.jpg" alt=""></a> </li>
  <li> <a href="images/h3.jpg" title="James Allison: 1-3"><img src="thumbnails/t3.jpg" alt=""></a> </li>
  <li> <a href="images/h4.jpg" title="James Allison: 1-4"><img src="thumbnails/t4.jpg" alt=""></a> </li>
  <li> <a href="images/h5.jpg" title="James Allison: 1-5"><img src="thumbnails/t5.jpg" alt=""></a> </li>
  <li> <a href="images/h6.jpg" title="James Allison: 1-6"><img src="thumbnails/t6.jpg" alt=""></a> </li>
</ul>
```

The css for the <a> elements

```
a {
    position: relative;
}
```

The stop method stops the queued animations before starting a new one

```
$("#image_list a").hover(function(evt) {
    $(this).stop(true).animate({
        top : 15
    }, "fast");
}, function(evt) {
    $(this).stop(true).animate({
        top : 0
    }, "fast");
});
// end hover
```

How to use easings with effects and animations

- ***Easing*** refers to the way an animation is performed. jQuery provides only two **easings**: **swing** and **linear**. Swing is the default, and it's the animation that you usually want.
- Plugins, including **jQuery UI**, provide many other types of easings.
- To use an easing:
  - you code a **script element** for the plugin library or jQuery UI.
  - Then, you code the **easing parameter** for a method with the name of any easing that the plugin or jQuery UI supports.

The syntax for using easing with effects and animations

The syntax for all of the basic methods except the fadeTo method

```
methodName([duration] [, easing] [, callback])
```

The syntax for the fadeTo method

```
fadeTo(duration, opacity [, easing] [, callback])
```

The syntax for the basic animate method

```
animate({properties}[, duration] [, easing] [,
callback])
```

Easinigs examples

A script element for getting the jQuery UI library from the Google CDN

```
<!-- the script element for jquery ui must come after the one for jquery -->
<script src="//ajax.googleapis.com/ajax/libs/jqueryui/1.10.0/jquery-ui.min.js"></script>
```

Easings Example 1

```
$("#faqs h2").toggle(function() {
    $(this).toggleClass("minus");
    $(this).next().slideDown(1000, "easeOutBounce");
}, function() {
    $(this).toggleClass("minus");
    $(this).next().slideUp(1000, "easeInBounce");
});
// end toggle
```

Easings Example 2

```
$("faqs h1").click(function() {
    $(this).animate({
        fontSize : "650%",
        opacity : 1,
        left : "+=275"
    }, 2000, "easeInExpo").animate({
        fontSize : "175%",
        left : "-=275"
    }, 1000, "easeOutExpo");
}) // end click
```

How to use the DOM manipulation and traversal methods

The methods for DOM replacement

- We’ve learned how to use the val and text methods to get and set values and text. Note, however, that we can also use **functions** to supply the values and text.
- If you use a function with the **val**, **text**, or **html** method, the function receives two parameters.
  - The first is the **index** of the current element in the set
  - The second is the old value, text, or html of the element.
- The replaceWith and replaceAll methods provide two different ways to replace one set of elements with another set.

The methods for DOM replacement (continued)

Method	Description
`val()`	Gets the value of the first selected control on a form.
`val(value)`	Sets the value of the first selected control on a form.
`val(function)`	Sets the value of the first selected control on a form to the value returned by the function.
`text()`	Gets the combined text contents of all selected elements including their descendants.
`text(textString)`	Sets the contents of each selected element to the specified text.
`text(function)`	Sets the contents of each selected element to the text returned by the function.
`html()`	Gets the HTML contents of the first selected element.
`html(htmlString)`	Sets the HTML contents of the first selected element to the specified HTML string.
`html(function)`	Sets the HTML contents of the first selected element to the HTML string returned by the function.
`replaceWith(content)`	Replaces each selected element with the specified content. This could be an HTML string, a DOM element, or a jQuery object.
`replaceAll(target)`	Replaces each target element with the selected elements. This is the reverse of how the replace With method is coded.

DOM replacement examples

Display all of the HTML in the aside element

```
alert($("aside").html());
```

Put an h2 element into an aside element

```
$("aside").html("<h2>Table of contents</h2>");
```

Replace all elements that have a class named "old., with an h2 element

```
$(".old").replaceWith("<h2></h2>");
```

Replace all elements that have a class named "old., with an h2 element

```
$("<h2></h2>").replaceAll(".old");
```

The methods for DOM insertion and cloning

- ❖ The **prepend** and **append** methods insert the content at the start or end of the selected elements, but within the elements.
- ❖ The **before** and **after** methods insert the content before or after the selected elements, not within the selected elements.
- ❖ If you use a function with the prepend or append method, it receives the index of the current element in the set and the old html for the element. If you use a function with the before or after method, it receives just the index of the current element.
- ❖ If you want to copy an element before you insert it somewhere else in the document, you need to use the **clone** method. Otherwise, the element is moved from its old location to its new one.

The methods for DOM insertion and cloning (continued)

Method	Description
`prepend(content)`	Inserts the specified content at the start of each selected element.
`prepend(function)`	Inserts the content returned by the function at the start of each selected element.
`prependTo(target)`	Inserts all of the selected elements at the start of each target element.
`append(content)`	Inserts the specified content at the end of each selected element.
`append(function)`	Inserts the content returned by the function at the end of each selected element.
`appendTo(target)`	Inserts all of the selected elements at the end of each target element.
`before(content)`	Inserts the specified content before each selected element.
`before(function)`	Inserts the content returned by the function before each selected element.
`insertBefore(target)`	Inserts all of the selected elements before each target element.
`after(content)`	Inserts the specified content after each selected element.
`after(function)`	Inserts the content returned by the function after each selected element.
`insertAfter(target)`	Inserts all of the selected elements after each target element.
`clone([withEvents])`	Creates a copy of the selected elements. The parameter is a Boolean that indicates if the event handlers are also copied.

DOM insertion and cloning methods examples

Insert an h2 element at the end of an aside element

```
$("aside").append("<h2>Table of contents</h2>");
```

Insert an <a> element after the last <p> element in an article

```
$("article p:last").after("<a href='#top'>Back to top</a>");
```

Insert the <a> elements in an article after the h2 elements in an aside

```
$("article a").insertAfter($("aside h2"));
```

Clone the <a> elements and insert them after the h2 element in an aside element

```
$("article a").alone().insertAfter($ ("aside h2"));
```

The methods for DOM wrapping and removal

- The DOM elements for **wrapping** let you wrap elements like <a> elements around other elements or text that you want to treat in a different way.
- If you use a function with the **wrap** or **wrapInner** method, it receives a single parameter with the index of the current element in the set.
- The DOM elements for removal and unwrapping let you remove or unwrap elements.

The methods for DOM wrapping and removal (continued)

Method	Description
`wrap(element)`	Wraps the specified element around each selected element.
`wrap(function)`	Wraps the element returned by the function around each selected element.
`wrapAll(element)`	Wraps the specified element around all of the selected elements.
`wrapInner(element)`	Wraps the specified element around the content of each selected element.
`wrapInner(function)`	Wraps the element returned by the function around the content of each selected element.
`empty()`	Removes all of the child nodes of the selected elements.
`remove([selector])`	Removes the selected elements from the DOM. If a selector is specified, it filters the selected elements.
`unwrap()`	Removes the parents of the selected elements.

DOM wrapping and removal methods examples

Wrap all <a> elements in h2 elements

```
$("a").wrap("<h2></h2>");
```

Wrap the h1 text in an <a> element

```
$("article h1").wraplnner("<a id='top'></a>");
```

Remove the first <a> element in an article

```
$("article a:first").remove();
```

The methods for working with styles

- These properties make it easy to get and set the properties for an element and also to get and set the height and width of an element.
- When you get a height or width value, it is returned as a number and pixels are assumed.
- When you specify a number for a height or width value, pixels are assumed. If you want to include the unit of measurement, enclose the value in quotation marks.
- If you use a function with the **css** method, it receives a parameter with the index of the current element in the set and a parameter with the old property value.

The methods for working with styles (continued)

Method	Description
`css(name)`	Gets the value of the named property.
`css(name, value)`	Sets the value of the named property.
`css(map)`	Sets the values of multiple properties specified by the name/value pairs in the properties map.
`css(name, function)`	Sets the value of the named property to the value returned by the function.
`height()`	Gets the height of the first selected element. This height doesn't include padding, margins, or borders.
`height(value)`	Sets the height of the first selected element.
`innerHeight()`	Gets the height of the first selected element including padding but not margins or borders.
`outerHeight([includeMargin])`	Gets the height of the first selected element including padding and borders. If the parameter is set to true, it also includes margins.
`width()`	Gets the width of the first selected element. This width doesn't include padding, margins, or borders.
`width(value)`	Sets the width of the first selected element.
`innerWidth()`	Gets the width of the first selected element including padding but not margins or borders.
`outerWidth([includeMargin])`	Gets the width of the first selected element including padding and borders. If the parameter is set to true, it also includes margins.

Methods for working with styles examples

Set the CSS color property for all h2 elements to blue

```
$(h2).css("color", "blue");
```

Use a map to set two CSS properties for all h2 elements

```
$(h2).css(("color": "blue", "font-size": "150%" ));
```

Get the height of an article element

```
var height = $("article").height();
```

The methods for positioning elements

- The scroll methods apply to window objects, elements with the overflow CSS property set to scroll, and elements with the overflow property set to auto if the height of the elements are smaller than their contents.
- The **scrollTop** method returns the number of pixels that are hidden from view above the scrollable area. If the scroll bar is at the top or if the element isn't scrollable, this number is 0.
- The **scrollLeft** method returns the number of pixels that are hidden from view to the left of the scrollable area. If the scroll bar is all the way to the left or if the element isn't scrollable, this number is 0.

The methods for positioning elements (continued)

Method	Description
`offset()`	Gets the coordinates of the first selected element and returns them in an object with top and left properties. These coordinates are relative to the document.
`offset(coordinates)`	Sets the coordinates of the first selected element relative to the document. The parameter is an object with top and left properties.
`position()`	Gets the coordinates of the first selected element and returns them in an object with top and left properties. These coordinates are relative to the parent element.
`scrollTop()`	Gets the position of the vertical scroll bar for the first selected element.
`scrollTop(value)`	Sets the position of the vertical scroll bar for the first selected element.
`scrollLeft()`	Gets the position of the horizontal scroll bar for the first selected element.
`scrollLeft(value)`	Sets the position of the horizontal scroll bar for the first selected element.

The methods for positioning elements examples

Get the top offset for an article element

```
var offsetTop = $("article").offset().top;
```

Set the offset coordinates for an aside element

```
var asideCoordinates = new Object();
asideCoordinates.top = 200;
asideCoordinates.left = 100;
$("aside").offset(asideCoordinates);
```

Move the scroll bar for the window to the top

```
$(window).scrollTop(0);
```

How to work with forms and data validation

How forms work (review)

- A ***form*** contains one or more *controls* (or *fields)* like text boxes, radio buttons, lists, or check boxes that can receive data.
- When you click on a ***submit button*** for a form (type is .. submit"), the form data is sent to the server as part of an **HTTP request**. When you click on a ***reset button*** for a form (type is .. reset"), the form data is reset to its default values.
- When a form is submitted to the server for processing, the data in the controls is sent along with the HTTP request.
- When you use the **get method** to submit a form, the URL that requests the file is followed by a **question mark** and name/value pairs that are separated by ampersands. These pairs contain the **name attributes** and **values** of the data that is submitted.
- When you use the **post method** the data is hidden.
- ***Data validation*** refers to checking the data collected by a form to make sure it is valid, and complete data validation is always done on the server. Then, if any invalid data is detected, the form is returned to the client so the user can correct the entries.
- To save round trips to the server when the data is invalid, some validation is usually done on the **client** before the data is sent to the server. However, this validation doesn't have to be as thorough as the validation that's done on the server.

The jQuery selectors and methods for forms

❖ jQuery provides special selectors for selecting the controls on a form; the **val** method for getting and setting the value in a control; and a **trim** method that can be used to trim a user's entry.

Method	Description
`:input`	All form elements: input, select, textarea, button.
`:text`	All text boxes: input elements with type equal to "text".
`:radio`	All radio buttons: input elements with type equal to "radio".
`:checkbox`	All check boxes: input elements with type equal to "checkbox".
`:file`	All file upload fields: input elements with type equal to "file".
`:password`	All password fields: input elements with type equal to "password".
`:submit`	All submit buttons and button elements: input elements with type equal to "submit" and button elements.
`:reset`	All reset buttons: input elements with type equal to "reset".
`:image`	All image buttons: input elements with type equal to "image".
`:button`	All buttons: button elements and input elements with type equal to "button".
`:disabled`	All disabled elements: elements that have the disabled attribute.
`:enabled`	All enabled elements: elements that don't have the disabled attribute.
`:checked`	All check boxes and radio buttons that are checked.
`:selected`	All options in select elements that are selected.

The jQuery event methods for forms

- You can use event handlers for the focus, blur, change, and select events to process data as the user works with individual controls.
- You can use an **event handler** for the click event of a regular button, not a submit button, to validate the data in a form. Then, if the data is valid, you can use the **submit method** to submit the form.
- You can also use an **event handler** for the submit event of a form to validate data before it is sent to the server. Then, if any of the data is invalid, you must issue the **preventDefault** method of the event object to **cancel the submission** of the data to the server.

The jQuery event methods for forms

Event Method	Description
`focus(handler)`	The handler runs when the focus moves to the selected element.
`blur(handler)`	The handler runs when the focus leaves the selected element.
`change(handler)`	The handler runs when the value in the selected element is changed.
`select(handler)`	The handler runs when the user selects text in a text or textarea box.
`submit(handler)`	The handler runs when a submit button is clicked.
`focus()`	Moves the focus to the selected element and triggers the focus event.
`blur()`	Removes the focus from the selected element and triggers the blur event.
`change()`	Triggers the change event.
`select()`	Triggers the select event.
`submit()`	Triggers the submit event for a form.

jQuery event methods for forms examples

A handler that disables or enables radio buttons when a check box is checked or unchecked

```
$("#contact_me").change(function() {// the change event for the checkbox
    if ($("#contact_me").attr("checked")) {
        $(":radio").attr("disabled", false) // enables radio buttons
    } else {
        $(":radio").attr("disabled", true)  // disables radio buttons
    }
});
```

A handler that triggers the submit event after some data validation

```
$(document).ready(function() {
    $("#join_list").click(function() {// join list is a button not a submit button
        // data validation code
        $("#email_form").submit();
    });
    // end click
});
// end ready
```

The jQuery shorthand methods for working with Ajax

The jQuery shorthand methods for working with Ajax

- **jQuery** includes several shorthand methods that let you request and receive HTML, XML, or JSON data.
- All of the shorthand methods let you include data that will let the web server filter the results of the request so only the right results are returned. You can send this data as a **query string** or as a **map**.
- The only difference between the **$.get** and **$.post** methods is the method that is used for the request (GET or POST). These are the same methods that you specify when you set up an HTML form.
- The $.each method is an expanded form of the each method that can be used to process the items in the returned data.

The jQuery shorthand methods for working with Ajax (continued)

The shorthand methods for working with Ajax

Method	Description
`load(url[,data][,success])`	Load HTML data.
`$.get(url[,data][,success][,dataType])`	Load data with a GET request.
`$.post(url[,data][,success][,dataType])`	Load data with a POST request.
`$.getJSON(url[,data[,success])`	Load JSON data with a GET request.

The parameters for the shorthand methods

Parameter	Description
`url`	The string for the URL to which the request is sent.
`data`	A **map** or **string** that is sent to the server with the request, usually to filter the data that is returned.
`success`	A function that is executed if the request is successful.
`dataType`	A string that specifies the type of data (html, xml, json, script, or text). The default is XML.

The $.each method for processing the data that's returned

Method	Description
`$.each(collection, callback)`	The **collection** parameter is an object or array. The **callback** parameter is a function that's done for each item in the collection.

The jQuery shorthand methods (examples)

A load method

```
$("#solution").load("solutions.html");
```

A $.get method that includes data and calls a success function

```
$.get("getmanager.php", "name=agnes", showManager);
```

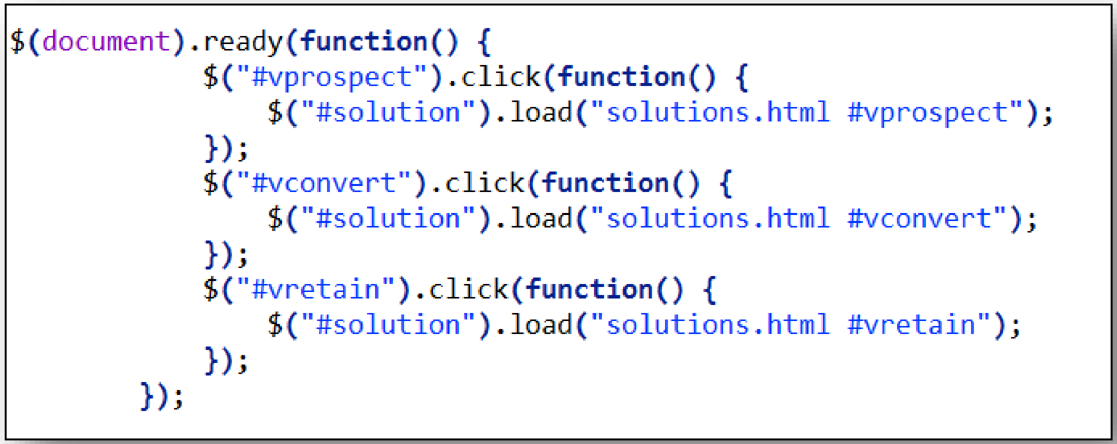
A $.getJSON method with an embedded function

```
$.getJSON("team.json", function(data){
    // the statements for the success function
}
```

How to use the load method to load HTML data

- The load function can only load content from files on the same server as the page making the call.
- During testing, you'll be able to load files from the file system in **Firefox**, **IE**, and **Safari**, but **Chrome** and **Opera** require all of the files to be on an actual web server.

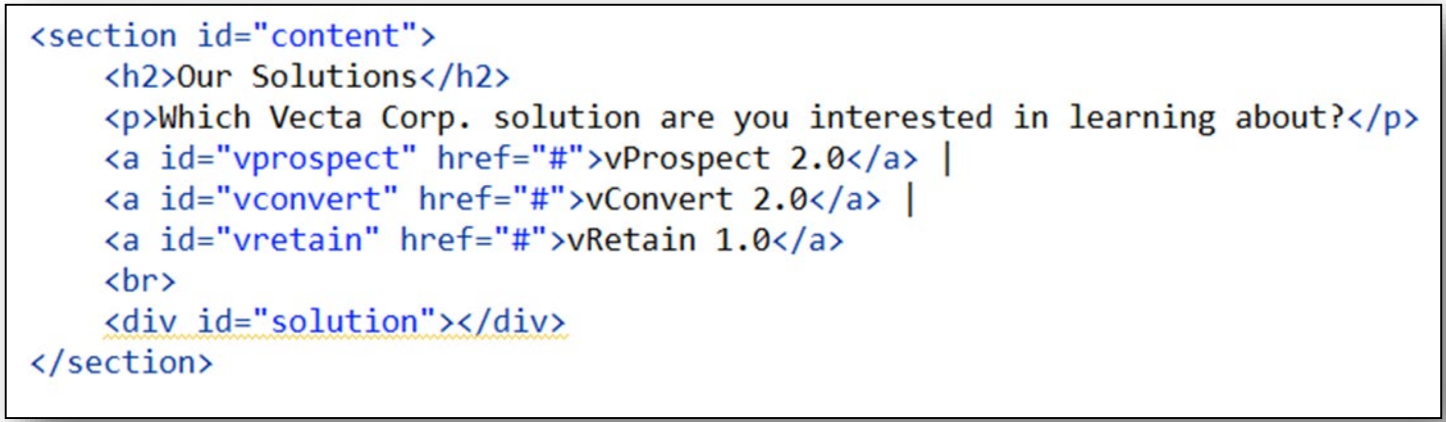
The JQuery that loads the data when a link Is clicked



```
$(document).ready(function() {
            $("#vprospect").click(function() {
                $("#solution").load("solutions.html #vprospect");
            });
            $("#vconvert").click(function() {
                $("#solution").load("solutions.html #vconvert");
            });
            $("#vretain").click(function() {
                $("#solution").load("solutions.html #vretain");
            });
        });
```

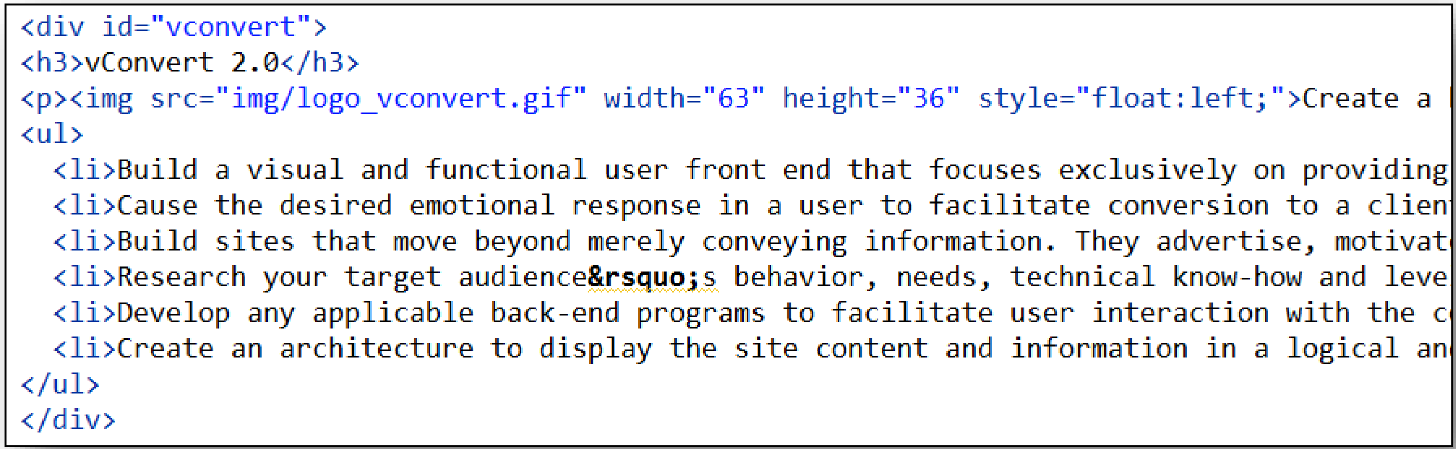
How to use the load method to load HTML data (Cont'd)

The HTML for the user Interface



```
<section id="content">
    <h2>Our Solutions</h2>
    <p>Which Vecta Corp. solution are you interested in learning about?</p>
    <a id="vprospect" href="#">vProspect 2.0</a> |
    <a id="vconvert" href="#">vConvert 2.0</a> |
    <a id="vretain" href="#">vRetain 1.0</a>
    <br>
    <div id="solution"></div>
</section>
```

The start of the second div element in the solutions.html file

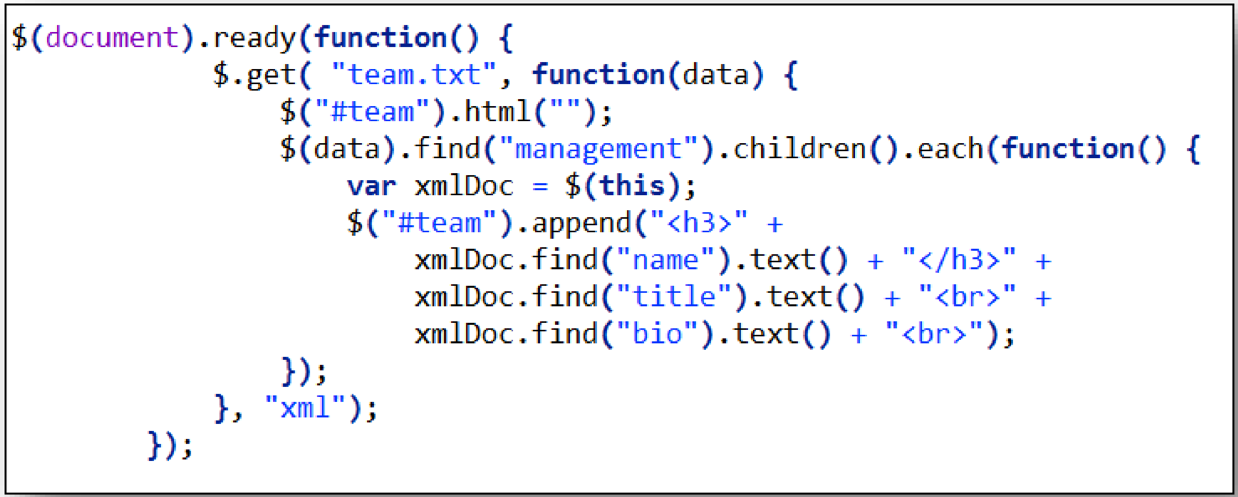


```
<div id="vconvert">
<h3>vConvert 2.0</h3>
<p><img src="img/logo_vconvert.gif" width="63" height="36" style="float:left;">Create a
<ul>
  <li>Build a visual and functional user front end that focuses exclusively on providing
  <li>Cause the desired emotional response in a user to facilitate conversion to a clien
  <li>Build sites that move beyond merely conveying information. They advertise, motivat
  <li>Research your target audience&rsquo;s behavior, needs, technical know-how and leve
  <li>Develop any applicable back-end programs to facilitate user interaction with the c
  <li>Create an architecture to display the site content and information in a logical an
</ul>
</div>
```

How to use the $.get or $.post method to load XML data

- ❖ The **$.get** and **$.post** methods work the same except for the method that’s used to send the data in the XHR request.
- ❖ You can use the jQuery **find** method to get the data in an **XML file**. Here, the first find method starts a chain that gets the children (team members) of the management data that's returned, and the other three find methods get the name, title, and bio fields for each team member.

The JQuery for loading the XML data

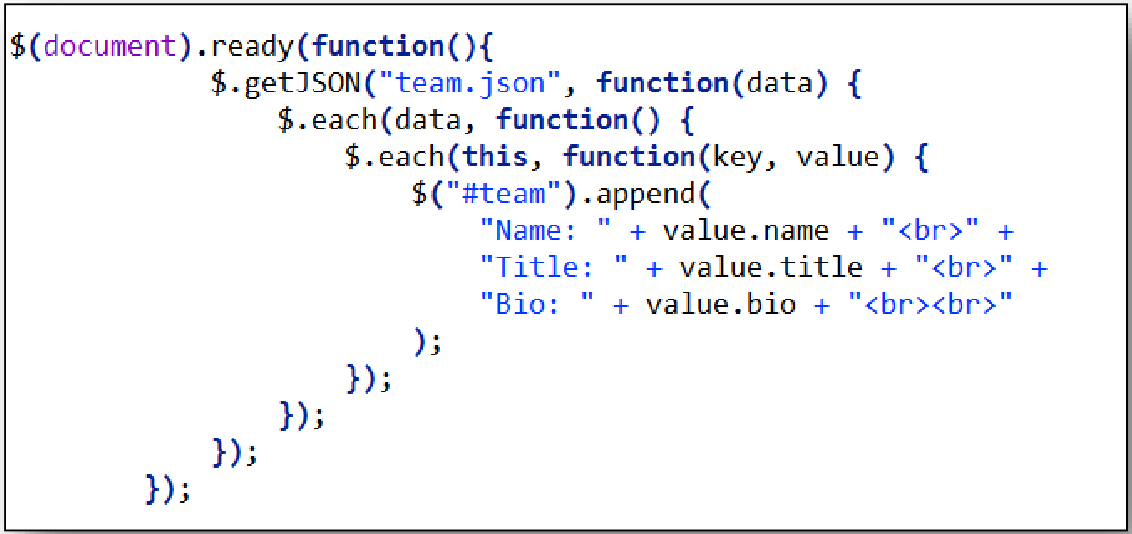


```
$(document).ready(function() {
      $.get( "team.txt", function(data) {
        $("#team").html("");
        $(data).find("management").children().each(function() {
          var xmlDoc = $(this);
          $("#team").append("<h3>" +
            xmlDoc.find("name").text() + "</h3>" +
            xmlDoc.find("title").text() + "<br>" +
            xmlDoc.find("bio").text() + "<br>");
        });
      }, "xml");
    });
```

How to use the $.getJSON method to load JSON data

- To process the returned **JSON data**, you can use nested **$.each** methods. The function in the first method will process each collection in the returned data (in this case, a single collection of team members).
- The function in the second **$.each** method will process each item (team member) in the collection. It will have two parameters that represent the **key** and **value** of each item. Then, you can use object notation to get the fields in the returned data.

The JQuery for loading the JSON data



```
$(document).ready(function(){
            $.getJSON("team.json", function(data) {
                $.each(data, function() {
                    $.each(this, function(key, value) {
                        $("#team").append(
                            "Name: " + value.name + "<br>" +
                            "Title: " + value.title + "<br>" +
                            "Bio: " + value.bio + "<br><br>"
                        );
                    });
                });
            });
        });
```

How to use the $.ajax method for working with Ajax

- ❖ The **$.ajax** method provides options that give you more control over the way the Ajax request works, such as providing a function for handling errors.
- ❖ The ***jqXHR object*** is jQuery's superset of the standard **XMLHttpRequest object** that provides the properties of that object.
- ❖ ***JSONP***, or **"JSON with padding"**, is a complement to JSON data that lets you request data from a server in a different domain, which is typically prohibited by web browsers due to their cross-domain security policies.

The syntax of the $.ajax method

```
$.ajax({ options })
```

Some of the options for the $.ajax method

Options	Description
`url`	The string for the URL to which the request is sent.
`beforesend(jqXHR, settings)`	A function that is executed before the request is sent. It can pass two parameters: the jqXHR object and a map of the settings for this object.
`cache`	A Boolean value that determines if the browser can cache the response.
`complete(jqXHR, status)`	A function that is executed when the request finishes. It can receive two parameters: the jqXHR object and a string that represents the status of the request.
`data`	A map or string that is sent to the server with the request, usually to filter the data that is returned.
`dataType`	A string that specifies the type of data (html, xml, json, script, or text).
`error(jqXHR, status, error)`	A function that is executed if the request fails. It can receive three parameters: the jqXHR object, a string that represents the type of error, and an exception object that receives the text portion of the HTTP status.
`jsonp`	A string containing the name of the JSONP parameter to be passed to the server. Defaults to "callback".
`password`	A string that contains a password that will be used to respond to an HTTP authentication challenge.
`success(data, status, jqXHR)`	A function that is executed if the request is successful. It can receive three parameters: the data that is returned, a string that describes the status, and the jqXHR object.
`timeout`	The number of milliseconds after which the request will time out in failure.
`type`	A string the specifies the GET or POST method.
`username`	A string that contains a user name that will be used to respond to an HTTP authentication challenge.

Summary

- jQuery enables developers to implement common tasks with minimal code
- Must link .js file containing library to HTML document
- Every jQuery statement starts with `$`
- Select elements in jQuery with CSS selector in quotes
- Can modify a selection
  - Append DOM traversal method to selection code

Summary (cont'd.)

- jQuery API includes methods for performing actions on elements
- Many methods allow you to look up or set value
- Event listener consists of
  - `$`
  - selector
  - method
- jQuery includes visual animation methods