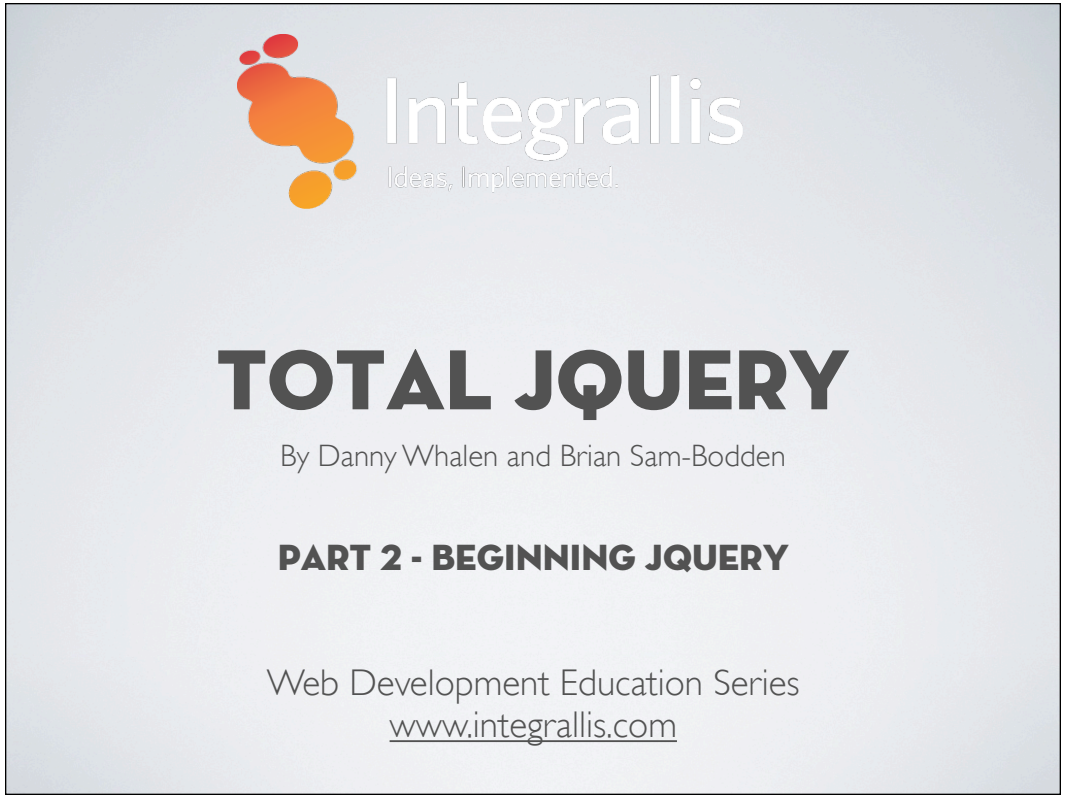


A publication of Integrallis Software, LLC.
www.integrallis.com/en/training
info@integrallis.com

OVERVIEW & OBJECTIVES

- Part 2 will cover:
  - An introduction to the DOM
  - Introducing jQuery: DOM manipulation simplified
  - Selecting, Manipulating and Adding Elements
  - Basic Animation
  - Attaching Events, Callbacks and Unobtrusive JavaScript
  - Simple Form Validation
  - Using jQuery plugins: Validation Plugin
  - Encapsulating Behaviors: Creating your own Plugins
  - Styling with jQuery

THE DOM

DOCUMENT OBJECT MODEL

DOM

DOCUMENT OBJECT MODEL

- The DOM is the browsers' internal representation of a document as nodes and attributes
- The is compose of a DOM tree is nested list of nodes, attributes and child nodes that is unique to the particular browser implementation
- The DOM is also an API to access and modify the elements of a document
- The DOM API is accessed using JavaScript, since it is so convoluted and the implementations are so inconsistent, a lot of the gripes about JavaScript are actually about the DOM

For a comprehensive coverage of how a modern browser works see http://www.html5rocks.com/en/tutorials/internals/howbrowserswork/

DOM

DOCUMENT OBJECT MODEL

"JavaScript is also coupled with The DOM, a horrendous API."

Douglas Crockford ,Yahoo!

DOM

DOCUMENT OBJECT MODEL

"Nearly every DOM method is broken in some way, in some browser"

John Resig, JQuery Creator

DOM

DOCUMENT OBJECT MODEL

- In a Web Browser environment the Browser **Window object is the global object** and the global execution context
- Two objects back the principal entities in a Browser:
  - The **Document** object represents an **HTML document**
  - The **Window** object represents the **browser window** (or frame) that serves as the display for a document
- The **Window** is the **entry point** to the DOM interaction; it **defines** a **document property** that points to the Document object

DOM

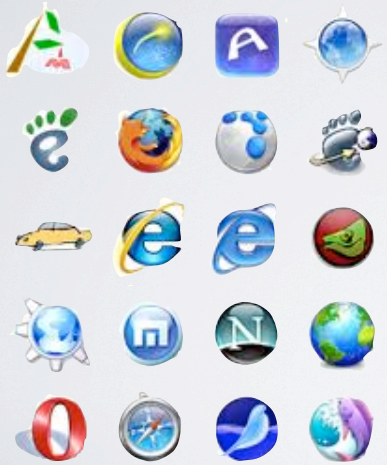
DOCUMENT OBJECT MODEL

- DOM API provides a set of basic element retrieval methods:
  - **getElementById**: The most used DOM method. Implementations in older browsers sometimes return elements with a name that matches the requested id
  - **getElementsByTagName**: Badly broken in IE < 8
  - **getElementsByClassName**
  - **querySelectorAll**: Newer method for selecting using CSS. Some consistency issues from browser to browser

DOM

DOCUMENT OBJECT MODEL

- Writing cross-browser code is extremely difficult:



	Win 2000	Win XP	Win Vista	Mac 10.4	Mac 10.5
Firefox 3.†	A-grade	A-grade	A-grade	A-grade	A-grade
Firefox 2.†		A-grade			A-grade
IE 7.0		A-grade	A-grade		
IE 6.0	A-grade	A-grade			
Opera 9.5†		A-grade			A-grade
Safari 3.1†				A-grade	A-grade

Yahoo! Browser Grading Matrix

DOM

DOCUMENT OBJECT MODEL

- In summary the problems faced by JavaScript developers stem from the DOM API inconsistencies amongst the different browsers:
  - Missing features / Extra features
  - Browser Bugs: Too many to count!
  - External code: Inconsistencies in the markup
  - Bad JavaScript: Global variables stepping over code, monkey patching of core objects or DOM objects


jQuery
write less, do more.

JQUERY

WRITE LESS, DO MORE

- JQuery...
  - is a fast and concise JavaScript library that simplifies HTML document traversing and manipulation
  - provides an API designed with a focus on expressiveness and convenience
  - simplifies event handling and an event-driven architecture
  - works well across all important browsers!
  - makes JavaScript fun!

JQUERY

WRITE LESS, DO MORE

- **JQuery** advantages:
  - optimized for DOM manipulation
  - operates on multiple DOM elements with no special syntax
  - powerful CSS3 selector API (XPath is also supported)
  - fastest selector performance amongst JS libraries
  - modular plugin architecture
  - provides a single namespace and it plays well with others (doesn't abuse the global namespace)

jQuery
write less. do more.

stackoverflow Questions Tags Users Badges Unanswered Ask Question

Add a number to another number in JavaScript

0

hallo

I have got a number in my JavaScript variable! Now how do I add another number to it? Please

javascript

3 Answers

oldest newest votes

22

You should definitely use jQuery. It's really great and does all things

link | edit | flag

answered **11 minutes ago**
I-3jQuery
1,234 2 13

I agree, jQuery is really the best, it solves all kinds of browser problems and is good, as well – ||sumc0da 8 mins ago

+1 jquery is best quality code ever, if you don't use your a idiot – Werry_Togan 4 mins ago

add comment

4

I think there's a jQuery plugin for that. Google for jQuery basic arithmetic plugin.

link | edit | flag

answered **5 minutes ago**
Timothy Goatse
4,321 1 12

yeah, jQuery is definately the way to go – fishnipples 5 mins ago

I used the jQuery diet plugin and lost 10kg in a week – jfatty 4 mins ago

add comment

-2

To add numbers together you should use the + operator, for example:

```
var a= 1;
var b= a+2;
alert(b);  // 3
```

link | edit | delete | flag

answered **50 seconds ago**
bobince
some

-1 not enough jQuery – ||sumc0da 30 secs ago

you suck – Timothy Goatse 3 secs ago

tagged
javascript × 18553

asked
a while ago

viewed
some times

latest activity
just now

Wanted: Yet another ASP.NET developer. See this and other great job listings at jobs.stackoverflow.com.

Related

What is the best number?

How can I use JavaScript to parse some HTML using regex?

JavaScript: why is my text content getting mangled when I clone nodes? Obviously I must be doing something wrong as jQuery is perfect

Stupid JavaScript floating point numbers are broken

How can I extract number from HTML using a regex without ZALGO singing the song that ends the world?

Is there a jQuery plugin for making an HTML page appear in the browser?

Where are my legs?

JQUERY

WRITE LESS, DO MORE

- Let's start with a simple example; removing multiple elements using raw JavaScript (the hard way)
- The **hide_divs** function retrieves all elements by tag name where the tag name is **'div'** using the **getElementsByTagName** method
- Next, the function iterates over the elements setting the **style.display** to **'none'** effectively hiding the divs

```
function hide_divs() {
    var divs = document.getElementsByTagName('div');

    for (var i = 0; i < divs.length; i++) {
        divs[i].style.display = 'none';
    }
}
```

JQUERY

WRITE LESS, DO MORE

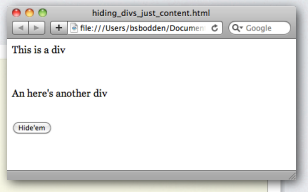
- Let's start with a simple skeleton HTML page that will serve to host our example:

```
<html>
    <head>
        <style type="text/css">
        </style>
    </head>
    <body>
        <script>
        </script>
    </body>
</html>
```

JQUERY

WRITE LESS, DO MORE

- We add a couple of **div**'s wrapping a paragraph of text
- We add a button with an **onclick** event set to trigger the **hide_divs** function



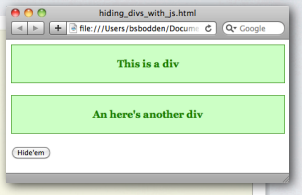

```
<html>
    <head>
        <style type="text/css">
        </style>
    </head>
    <body>
        <div class="boxybox"><p>This is a div</p></div><br />
        <div class="boxybox"><p>An here's another div</p></div><br />
        <input type="button" onclick="hide_divs()" value="Hide'em" />
        <script>
        </script>
    </body>
</html>
```

JQUERY

WRITE LESS, DO MORE

- We'll also add an inline style for the class **boxybox**:

```
<html>
    <head>
        <style type="text/css">
        .boxybox {
            border:solid 1px #349534;
            background:#C9FFCA;
            color:#008000;
            font-weight:bold;
            padding:4px;
            text-align:center;
        }
        </style>
    </head>
    <body>
        <div class="boxybox"><p>This is a div</p></div><br />
        <div class="boxybox"><p>An here's another div</p></div><br />
        <input type="button" onclick="hide_divs()" value="Hide'em" />
        <script>
        </script>
    </body>
</html>
```



JQUERY

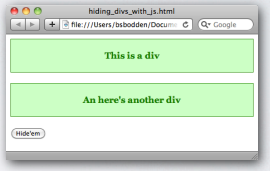
WRITE LESS, DO MORE

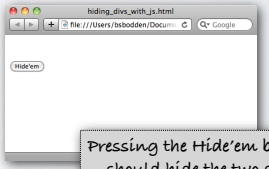
Follow Along

- To complete the example add the **hide_divs** function to the page:

```
<html>
    <head>
        <style type="text/css">
        .boxybox {
            border:solid 1px #349534;
            background:#C9FFCA;
            color:#008000;
            font-weight:bold;
            padding:4px;
            text-align:center;
        }
        </style>
    </head>
    <body>
        <div class="boxybox"><p>This is a div</p></div><br />
        <div class="boxybox"><p>An here's another div</p></div><br />
        <input type="button" onclick="hide_divs()" value="Hide'em" />

        <script>
        function hide_divs() {
            var divs = document.getElementsByTagName('div');

            for (var i = 0; i < divs.length; i++) {
                divs[i].style.display = 'none';
            }
        }
        </script>
    </body>
</html>
```

part_2/examples/hiding_with_js.html





Pressing the Hide'em button should hide the two divs

JQUERY

WRITE LESS, DO MORE

- Let's perform the same task with jQuery
- To get started we first need to include in our HTML pages the appropriate `.js` files
- jQuery can be downloaded from http://docs.jquery.com/Downloading_jQuery
- jQuery can be also be linked to your pages using one of the Content Delivery Networks (CDN) that host jQuery including `code.jquery.com`, Google and Microsoft
- Most CDNs provide both regular and minified versions of the `.js` files

JQUERY

WRITE LESS, DO MORE

- For our examples we will use the Google CDN provided versions of the jQuery libraries
- Include a **script** tag in the head of the HTML page with the **src** set to the Google hosted **jquery.js** file version **1.8.2**:

```
<script src="http://ajax.googleapis.com/ajax/libs/jquery/1.8.2/jquery.js"></script>
```

- For the "follow along" examples use the sample HTML file located in:

part_2/examples/hiding_with_jquery.html

JQUERY

WRITE LESS, DO MORE

- jQuery follows a simple philosophy:
  1. Find one or more DOM elements
  2. Do something to those elements
- The entry point for the JQuery API is the jQuery function: `jQuery()`
- To find page elements we can pass a CSS selector expression to the jQuery function:

```
jQuery('div');
```

JQUERY

WRITE LESS, DO MORE

- The call to the `jQuery` function passing a CSS selector returns a jQuery collection (an array of DOM elements):

```
jQuery('div');
```

- You can call methods on the returned collection like:

```
jQuery('div').size();
```

JQUERY

WRITE LESS, DO MORE

- jQuery comes with a wide array of methods for traversing the DOM tree:

```
jQuery('div').next();
jQuery('div').prev();
jQuery('div').prev('p');
jQuery('div').parent();
jQuery('div').parents();
```

JQUERY

WRITE LESS, DO MORE

- Elements of a jQuery collection can be accessed like an array:

```
jQuery('div')[0];
jQuery('div')[1];
```

- You can iterate with the `each` iterator method passing an anonymous function:

```
jQuery('div').each(function(index, div) {
    alert(index + ': ' + div.id);
});
```

JQUERY

WRITE LESS, DO MORE

Follow Along

- Let's rewrite the `hide_divs` function using jQuery
- First we'll collect all the `div` elements
- Then we'll hide them using the `hide` method

```
function hide_divs() {
    jQuery('div').hide();
}
```

- The jQuery API allows you to daisy chain method calls providing a fluid API

JQUERY

WRITE LESS, DO MORE

Follow Along

- The `jQuery` function is the entry point to all of the functionality of the jQuery API, it is effectively the library's namespace
- jQuery provides a shorter alias to the `jQuery` function: the dollar sign (`$`):

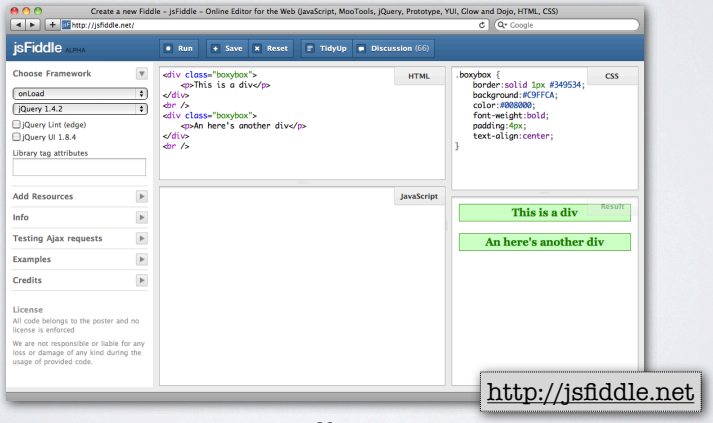
part_2/examples/hiding_with_jquery.html

```
function hide_divs() {
    $('div').hide();
}
```

JQUERY

WRITE LESS, DO MORE

- An alternative way to experiment with jQuery is **jsFiddle**; a JavaScript Web "playground" where developers can dynamically test the effects of their JavaScript using a multitude of libraries

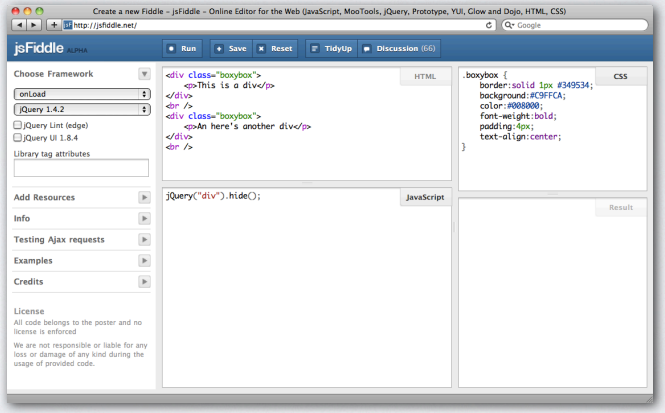


http://jsfiddle.net

JQUERY

WRITE LESS, DO MORE

- We can replicate the previous example using **jsFiddle** by selecting the jQuery 1.8.2 framework, adding our JavaScript and pressing the pressing the "Run" button:



SELECTORS

HARNESSING THE POWER OF CSS3

JQUERY SELECTORS

HARNESSING THE POWER OF CSS3

- CSS (Cascading Style Sheets) is a language for describing the rendering of HTML and XML documents
- CSS uses selectors for finding elements in a document and applying style properties to those elements
- jQuery supports CSS3 selectors in its underlying implementation; the Sizzle CSS selector engine provides the fastest way to select DOM elements
- The better you understand and apply CSS the better you will be as a jQuery developer

The CSS3 standard is actively under development by the W3C, see http://www.w3.org/Style/CSS/current-work

JQUERY SELECTORS

HARNESSING THE POWER OF CSS3

- CSS selectors are expressions that "select" elements on an HTML page
- If you've look at a CSS stylesheet before you have seen the familiar structure:

```
body {
    font-family: "Lucida Grande", Lucida, Verdana, sans-serif;
    overflow: hidden;
    height: 100%;
    max-height: 100%;
}
```

- The selector consists of anything left of the curly braces
- In the case above "`body`" selects the body element of the HTML document and applies the declaration block `{}` to it

JQUERY SELECTORS

HARNESSING THE POWER OF CSS3

- There are many different kind of selectors:
  - *Type:* Match by name, e.g. `body`, `div`, `p`, `em`, `h1`
  - *Class:* Match by class attribute, e.g. `.big`
  - *ID:* Match by id attribute, e.g. `#nav`
  - *Descendants:* Select descendants, e.g. `ul em`
  - *Child:* Select direct descendants, e.g. `div > em`
  - *Adjacent Sibling:* Following sibling, e.g. `h2 + h3`

IDs can only be applied once per page, while classes can be used as many times on a page as needed

JQUERY SELECTORS

HARNESSING THE POWER OF CSS3

- CSS selectors types (cont.):
  - *Attribute:* Match by attribute value or presence, e.g.

    img[src="go.png"], img[title],
    img[title~= "cancel"], img[title|= "cancel"],
    img[title^= "cancel"]

  - *Pseudo-classes:* Based on (dynamic) properties, e.g. :focus, :hover, :visited, :active, :first-child
  - *Pseudo-elements:* Select items not directly available in the document tree, e.g. p:first-line, p:first-letter, :before, :after

~= matches value in a space separated list
|= matches value a dash separated list
$= end with a certain value
^= starts with a certain value
*= contains a certain value

JQUERY SELECTORS

FINDING ELEMENTS - JAVASCRIPT CONSOLE

- We'll start our exploration of CSS selectors with a sample HTML file **sample_page.html**

part_2/examples/sample_page.html

JQUERY SELECTORS

FINDING ELEMENTS - JAVASCRIPT CONSOLE

- The sample HTML structure is shown below:

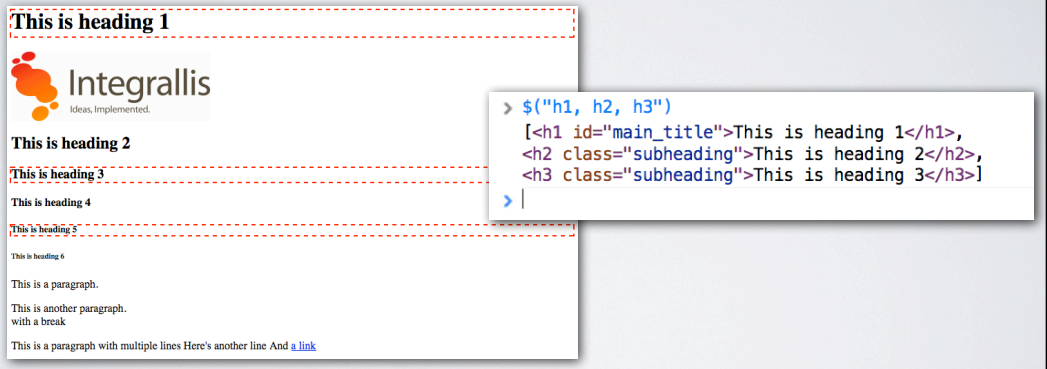
```html
<html>
    <head>
        <script src="http://ajax.googleapis.com/ajax/libs/jquery/1.8.2/jquery.js"></script>
    </head>
    <body>
        <h1 id="main_title">This is heading 1</h1>
        <p id="logo">
            <img src="http://integrallis.com/theme/images/logo.png" />
        </p>

        <h2 class="subheading">This is heading 2</h2>
        <h3 class="subheading">This is heading 3</h3>
        <h4 class="subheading">This is heading 4</h4>
        <h5 class="subheading">This is heading 5</h5>
        <h6 class="subheading">This is heading 6</h6>

        <p>This is a paragraph.</p>
        <p class="multiline">This is another paragraph.<br/> with a break</p>
        <p>
            This is a paragraph with multiple lines
            Here's another line
            And <a href="http://www.integrallis.com">a link</a>
        </p>

    </body>
</html>
```

part_2/examples/sample_page.html

JQUERY SELECTORS

FINDING ELEMENTS - JAVASCRIPT CONSOLE

- CSS selectors can be comma separated to match multiple elements, here we select 3 elements by type:

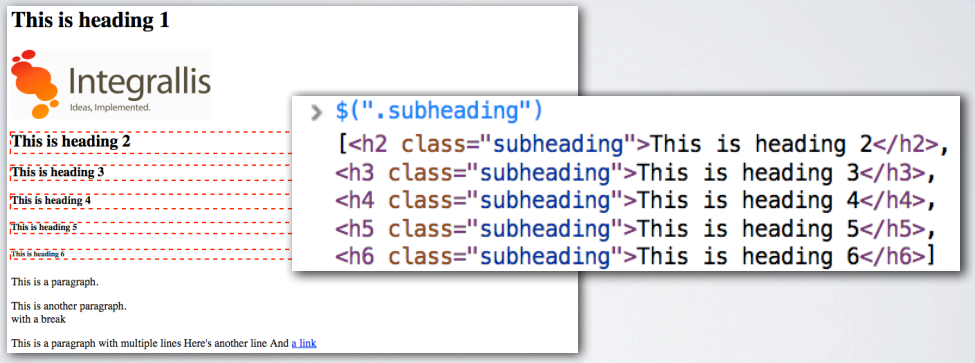



```
h1, h3, h5
```

JQUERY SELECTORS

FINDING ELEMENTS - JAVASCRIPT CONSOLE

- Selecting all elements with the **subheading** class:




```
> $(".subheading")
  [<h2 class="subheading">This is heading 2</h2>,
  <h3 class="subheading">This is heading 3</h3>,
  <h4 class="subheading">This is heading 4</h4>,
  <h5 class="subheading">This is heading 5</h5>,
  <h6 class="subheading">This is heading 6</h6>]
```

.subheading

JQUERY SELECTORS

FINDING ELEMENTS - JAVASCRIPT CONSOLE

- Selecting an element by the id logo:


```
> $("#logo")
  [
  ▼<p id="logo">
     <img src="http://integrallis.com/theme/images/
     logo.png">
   </p>
  ]
```

#logo

JQUERY SELECTORS

FINDING ELEMENTS - JAVASCRIPT CONSOLE

- With CSS we can select non-visible elements
- Below we use a descendant selector: get all `script` elements that are children of `head`

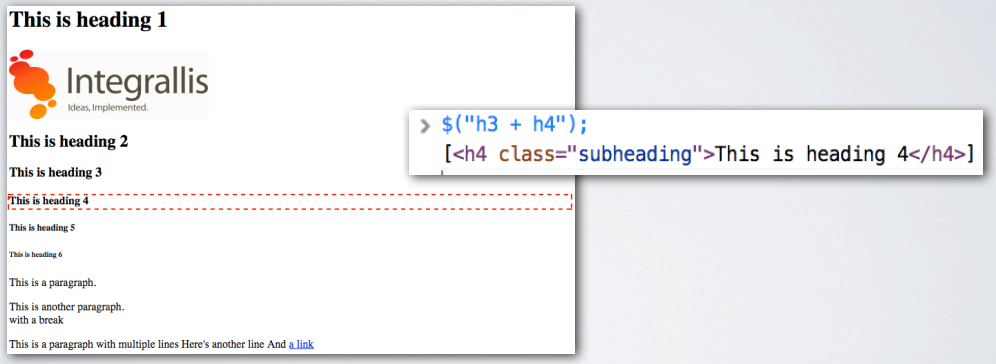
```
> $("head script")
  [<script src="http://ajax.googleapis.com/ajax/libs/jquery/1.8.2/jquery.js"></script>]
```

head script

JQUERY SELECTORS

FINDING ELEMENTS - JAVASCRIPT CONSOLE

- Selecting an adjacent sibling: find all `h4`s immediately following an `h3`




```
> $("h3 + h4");
  [<h4 class="subheading">This is heading 4</h4>]
```

`h3 + h4`

JQUERY SELECTORS

FINDING ELEMENTS - JAVASCRIPT CONSOLE

- Selecting any `a` element with an `href` attribute that starts with "`http://`" and is a child element of any `p` element

This is heading 6

This is a paragraph.

This is another paragraph.
with a break

This is a paragraph with multiple lines Here's another line And a link

```
> $('p a[href^="http://"]')
  [<a href="http://www.integrallis.com">a link</a>]
```

`p a[href^="http://"]`

JQUERY SELECTORS

CSS SELECTOR RESOURCES

- The complete reference for the selectors API can be found at http://api.jquery.com/category/selectors/
- A decent CSS selector cheat sheet can be found at http://net.tutsplus.com/tutorials/html-css-techniques/the-30-css-selectors-you-must-memorize/

ON DOCUMENT READY

WAITING FOR THE DOM TO BE READY

ON DOCUMENT READY

WAITING FOR DOM READINESS

- To successfully manipulate an HTML document we need to wait for the document to load
- Any DOM manipulations before the document is completely loaded can't be guaranteed to succeed
- Using raw JavaScript the implementation for this technique varies from browser to browser and it is easy to get it wrong

ON DOCUMENT READY

WAITING FOR DOM READINESS

- JQuery provides the ready method which takes a function to execute after the DOM is ready
- jQuery implementation works across all important browsers

```
$(document).ready(handler);
$().ready(handler); // not recommended
$(handler);

// will only bind if ready event hasn't fire
$(document).bind('ready', handler);
```

- The handler parameter is the function to be executed, typically an anonymous function

JavaScript load event does not get triggered until all assets loading is complete. In most cases, the script can be run as soon as the DOM is fully constructed. The function passed to .ready() is guaranteed to be executed after the DOM is ready

When using scripts that rely on the value of CSS style properties, reference stylesheets before referencing scripts

Code that relies on loaded assets should be placed in a handler for the load event instead

ON DOCUMENT READY

Follow Along

WAITING FOR DOM READINESS

- In a script tag we can declare the JavaScript that will be run when the document is ready:

```
<html>
    <head>
        <script src="http://ajax.googleapis.com/ajax/libs/jquery/1.8.2/jquery.js"></script>
        <script>
        $(document).ready(function(){
            alert("The document is ready!");
        });
        </script>
    </head>
    <body>
        <h1>JQuery is here!</h1>
    </body>
</html>
```

This basic skeleton will serve as the foundation to most JQuery example and labs

part_2/examples/on_ready.html

FIND AND MANIPULATE

SELECTING AND ANIMATING PAGE ELEMENTS

FIND AND MANIPULATE

Follow Along

SELECTING AND ANIMATING PAGE ELEMENTS

- Let's use jQuery to:
  - Select headings `h1`, `h3` and `h5`
  - Attach an `onclick` event to animate the elements

FIND AND MANIPULATE

Follow Along

SELECTING AND ANIMATING PAGE ELEMENTS

```
<style>
h1, h3, h5 {
  background-color: #bca;
  width: 300px;
  border: 1px solid green;
}
</style>
<script src=".../jquery.js"></script>
<script>
$(document).ready(function() {
    $("h1, h3, h5").click(function() {
        $(this).animate({
            width: "50%",
            opacity: 0.4,
            marginLeft: "0.6in",
            fontSize: "3em",
            borderWidth: "10px"
        }, 1500 );
    });
});
</script>
```

part_2/examples/select_and_animate.html

- An inline style is added to decorate the `h1`, `h3` and `h5` elements
- On document ready we select the elements and attach an on click event with the `click()` jQuery method
- We pass an anonymous function to the `click` method that invokes the jQuery `animate` method on each of the selected elements via the `this` object `$(this)`

The reference for the animate method can be found at http://api.jquery.com/animate/

FIND AND MANIPULATE

Follow Along

SELECTING AND ANIMATING PAGE ELEMENTS

- Clicking on the `h1`, `h3` or `h5` will display the animation:

ADDING ELEMENTS

ADDING ELEMENTS TO THE DOM

- jQuery makes it easy to add elements to the document
- The versatile jQuery function can take a text description of an element and insert it somewhere in the document based on a CSS selector

```
<script>
$(document).ready(function() {
    $("<p>Kilroy was here!</p>").insertAfter(".subheading");
});
</script>
```

part_2/examples/adding_elements.html

Most jQuery methods return a jQuery object facilitating method chaining as shown above. If a method returns a different collection you can call .end() to reverse to the previous collection

ADDING ELEMENTS

ADDING ELEMENTS TO THE DOM

- The resulting page with the added paragraphs is shown below:

part_2/examples/adding_elements.html

LAB 2.0

FIND AND MANIPULATE

- In Lab `2.0` reproduce the "Kilroy was here!" example using `part_2/examples/lab_2_0.html` to:
  - Add the "Kilroy was here!" paragraphs, wait 5 seconds then change the text to "Kilroy is about to go!", then fade them out in 5 seconds
  - You can use the following methods:
    - `text` (http://api.jquery.com/text/#text2)
    - `delay` method (http://api.jquery.com/delay/)
    - `fadeOut` method (http://api.jquery.com/fadeOut/)
    - `hide` method (http://api.jquery.com/hide/)
    - `show` method (http://api.jquery.com/show/)

EFFICIENT SELECTION

BEST PRACTICES WHEN SELECTING PAGE ELEMENTS

EFFICIENT SELECTION

BEST PRACTICES WHEN SELECTING PAGE ELEMENTS

- In a jQuery application selecting is the first thing you do and it could be the most CPU/time intensive activity of your scripts
- Therefore it is crucial that you efficiently select the elements to be manipulated
- Keep in mind the "query" part of jQuery is the selection of elements to be manipulated
- Just like with SQL you wouldn't "select *" every single time nor would you keep on issuing the exact same query with the same parameters repeatedly

QUERYING WITH CONTEXT

BEST PRACTICES WHEN SELECTING PAGE ELEMENTS

- Optimize selection by specifying a search context
- You can pass a second parameter to the jQuery function to narrow down the search over a specific context
- This is useful when you already have a reference to the context discriminator DOM element

```
<p class="c">C outside</p>
<p class="d">D</p>
<div id="inner">
    <p class="c">C inside</p>
    <p class="d">D</p>
    <p class="f">F</p>
    <p class="g">G</p>
</div>
<p class="e">E</p>
```

```
var $inner = $('#inner');

$inner_c_1 = $('p.c', $inner).first().clone();
$inner_c_2 = $('#inner p.c').first().clone();

$("#output_1").append($inner_c_1);
$("#output_2").append($inner_c_2);
```

part_2/examples/within_context.html

In the first selector $inner_c_1 we are narrowing the search to the $inner element which it is assumed to have been previously selected

The second selector $inner_c_2 uses a compound CSS selector to achieve the same result

So which one should you use?

#1 if you already (and only if you already) have the context DOM element selected ($inner)
#2 if always going to be faster against the whole document since it takes advantage of querySelecterAll to do the selection in a single pass

Never do this => $inner_c_3 = $('p.c', $('#inner'));

CACHING QUERIES

BEST PRACTICES WHEN SELECTING PAGE ELEMENTS

- Imagine that we have a document with a set of divs containing of paragraphs with start reviews
- The divs have ids for the days of week, and a class for weekday or weekend
- There is existing jQuery code select all positive and negative reviews aggregated by weekend and weekday

```
<div id="monday" class="weekday">
    <p class="negative">*</p>
    <p class="positive">****</p>
    <p class="negative">**</p>
</div>
<div id="tuesday" class="weekday">
    <p class="negative">*</p>
    <p class="positive">***</p>
    <p class="negative">*</p>
    <p class="positive">*****</p>
</div>
```

```
$weekend_positives = $(".weekend p.positive");
$weekend_negatives = $(".weekend p.negative");
$weekday_positives = $(".weekday p.positive");
$weekday_negatives = $(".weekday p.negative");
```

part_2/examples/cache_queries.html

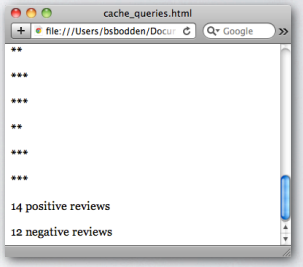
CACHING QUERIES

BEST PRACTICES WHEN SELECTING PAGE ELEMENTS

- Imagine now that we want to add a summary section with a count of all positive and all negative reviews
- Instead of re-querying the DOM we could simply combine the results of the previous queries to achieve our goal

```
$all_positives = $weekday_positives.add($weekend_positives);
$all_negatives = $weekday_negatives.add($weekend_negatives);

$summary = $("#summary");
$summary.append("<p>" + $all_positives.size() + " positive reviews</p>");
$summary.append("<p>" + $all_negatives.size() + " negative reviews</p>");
```

part_2/examples/cache_queries.html



EFFICIENT SELECTION

BEST PRACTICES WHEN SELECTING PAGE ELEMENTS

- Some other techniques for better selection:
  - *Use IDs instead of classes* (hopefully your IDs are unique) and the implementation takes advantage of the DOM's fast and native getElementByID() method
  - *Don't be afraid of restructuring your markup:* Sometimes poor decisions in your markup structure can lead to costly lookups when querying the document

UNOBTRUSIVE JAVASCRIPT

SEPARATING BEHAVIOR FROM STRUCTURE

UNOBTRUSIVE JS

SEPARATING BEHAVIOR FROM STRUCTURE

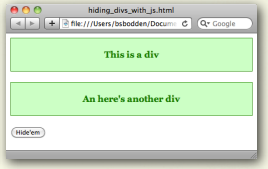
- Separating script behavior from page presentation is one way to keep your web applications clean but with raw JavaScript it is not always possible to do this in a clean way
- JQuery embraces separation on behavior from structure, a pattern known as ***Unobtrusive JavaScript***
- Operations on the DOM are externalized in your JavaScript files as well as the application of those operations to selected elements

UNOBTRUSIVE JS

SEPARATING BEHAVIOR FROM STRUCTURE

- Previously we assigned a behavior to a button in an obtrusive way. We mixed structure and behavior!

```
<html>
    <head>
        <style type="text/css">
        .boxybox {
            border:solid 1px #349534;
            background:#C9FFCA;
            color:#008000;
            font-weight:bold;
            padding:4px;
            text-align:center;
        }
        </style>
        <script src="http://ajax.googleapis.com/ajax/libs/jquery/1.8.2/jquery.min.js"></script>
    </head>
    <body>
        <div class="boxybox"><p>This is a div</p></div><br />
        <div class="boxybox"><p>An here's another div</p></div><br />
        <input type="button" onclick="hide_divs()" value="Hide'em" />

        <script>
        function hide_divs() {
            $('div').hide();
        }
        </script>
    </body>
</html>
```



part_2/examples/hiding_with_jquery.html

UNOBTRUSIVE JS

SEPARATING BEHAVIOR FROM STRUCTURE

- With jQuery and CSS selectors we can attach event to elements in an unobtrusive way!

```
<input type="button" onclick="hide_divs()" value="Hide'em" />

<script>
function hide_divs() {
    $('div').hide();
}
</script>
```

Obtrusive!

```
<input type="button" value="Hide'em" />

<script>
$(document).ready(function(){
    $('input:button[value^="Hide"]').click(function(){
        $("div").hide();
    });
});
</script>
```

Unobtrusive!

part_2/examples/unobtrusive.html

CALLBACKS

REGISTERING EVENT LISTENERS

CALLBACKS

REGISTERING EVENT LISTENERS

- Many jQuery methods accept a callback function as a parameter
- Callbacks specify code that runs in response to an event
- For example, most jQuery effect/animation methods can take a function that will be executed when the animation finishes

```
<input type="button" value="Hide'em" />

<script>
$(document).ready(function(){
    $('input:button[value^="Hide"]').click(function(){
        $("div").hide(5000, function(){
            alert("And now we're done!");
        });
    });
});
</script>
```

part_2/examples/callbacks.html

SIMPLE FORM VALIDATION

CLIENT-SIDE VALIDATION WITH JQUERY

FORM VALIDATON

CLIENT-SIDE VALIDATION WITH JQUERY

- Client-side form validation can be used to assist users and prevent unnecessary server round trips
- Client-side validation should support server-side validation, not replace it
- JQuery provides CSS selectors specific to forms such as the `:input` filter which selects all input elements (select boxes, text areas or buttons)
- Actions can be taken based on the value of a form field, which can be accessed with the `val()` function

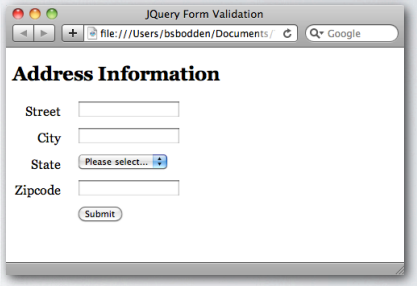
FORM VALIDATON

FOLLOW ALONG

CLIENT-SIDE VALIDATION WITH JQUERY

- Let's perform some simple client-side validation for the address form shown below:

```
<form action="">
    <div>
        <label for="street">Street</label>
        <input name="street" id="street" type="text">
    </div>
    <div>
        <label for="city">City</label>
        <input name="city" id="city" type="text">
    </div>
    <div>
        <label for="state">State</label>
        <select id="state" name="state">
            <option value="" selected>Please select...</option>
            <option value="CA">California</option>
        </select>
    </div>
    <div>
        <label for="zip">Zipcode</label>
        <input name="zipcode" id="zipcode" type="text">
    </div>
    <div>
        <input type="submit" value="Submit" />
    </div>
</form>
```



part_2/examples/address_form.html

FORM VALIDATON

CLIENT-SIDE VALIDATION WITH JQUERY

- The following jQuery code snippet adds an event handler to the **:submit** button
- Then selects all **:text** boxes and use an anonymous function to validate them changing their style with the **css** command if the **length** of their **val()** is zero

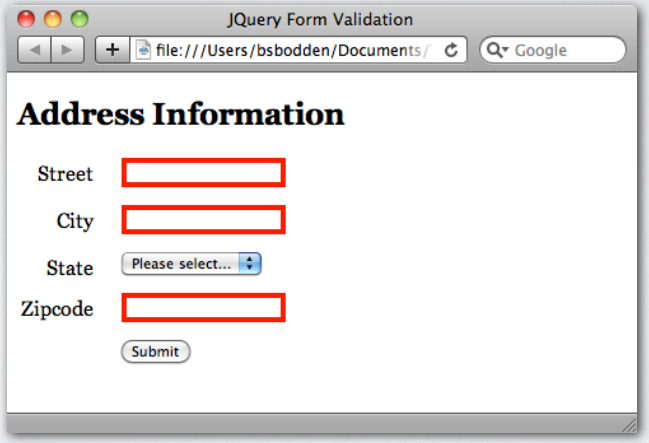
```
// basic form validation
$(document).ready(function(){
  $(':submit').click(function(e) {
    $(':text').each(function() {
      if($(this).val().length == 0) {
        $(this).css('border', '4px solid red');
      }
    });
    e.preventDefault();
  });
});
```

part_2/examples/address_form.html

FORM VALIDATON

CLIENT-SIDE VALIDATION WITH JQUERY

- Submitting the form should result in all empty text fields being highlighted in red:



LAB 2.1

FIND, MANIPULATE AND ANIMATE

- In Lab 2.1 modify the address_form.html page to:
  - enable the form visibility to be toggled by clicking on a button, label or graphic
  - animate the showing and hiding of the form
  - reset the input field CSS on submit if the field is valid
  - Use methods from jQuery's Effects API (http://api.jquery.com/category/effects/)
  - Bonus feature: Use Uniform to style your form (http://pixelmatrixdesign.com/uniform/)

PLUGINS

AUGMENTING JQUERY'S FUNCTIONALITY

PLUGINS

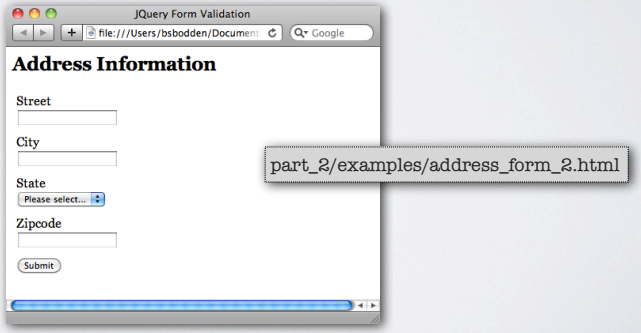
AUGMENTING JQUERY'S FUNCTIONALITY

- jQuery can be extended with plugins and there is a rich ecosystem of plugins available as open source
- jQuery's clean plugin architecture makes it easy to abstract and encapsulate functionality into a plugin
- The official jQuery plugins collection (http://plugins.jquery.com/) is no longer around, but you can still get to it here: http://archive.plugins.jquery.com/
- jQuery also has an official set of user interface components hosted at http://jqueryui.com

FORM VALIDATION

VALIDATION JQUERY PLUGIN

- A popular form validation jQuery plugin is Jorn Zaefferer's validation plugin available at http://docs.jquery.com/Plugins/Validation
- Download the HTML page shown below which we'll validate using the form validation plugin



FORM VALIDATION

VALIDATION JQUERY PLUGIN

- The plugin .js file is available from the Microsoft CDN at http://ajax.aspnetcdn.com/ajax/jquery.validate/1.9/jquery.validate.js
- The plugin works by applying a set of validation rules to a form element

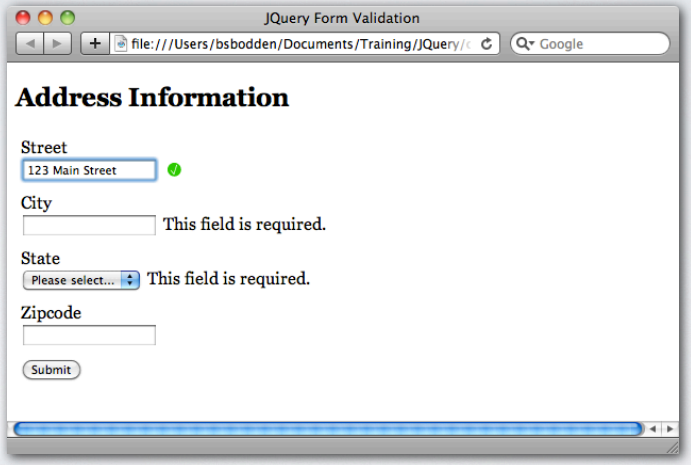
```
// validation plugin
$('#address_form').validate({
  rules: {
    street: {
      required: true,
    },
    city: {
      required: true,
    },
    state: {
      required: true,
    },
    zipcode: {
      required: true,
    }
  },
  success: function(label) {
    label.text('OK!').addClass('valid');
  }
});
```

part_2/examples/address_form_2.html

FORM VALIDATION

VALIDATION JQUERY PLUGIN

- The validation plugin makes form validation a breeze:



LAB 2.2

ADVANCED FORM VALIDATION

- For Lab 2.2 create an HTML page with a contact form containing:
  - First Name (required)
  - Last Name (required)
  - E-mail (required, must be valid)
  - Website URL (optional, must be valid)
  - Radio Button Group (required, contact me yes/no)
    - Use jQuery to show an information message in a div explaining the user's choice

CREATING PLUGINS

EXTENDING JQUERY

CREATING PLUGINS

EXTENDING JQUERY

- Below is a very simple jQuery plugin; the **helloWorld** plugin
- It extends the **jQuery.fn** object with a plugin configuration object that returns a function that returns function (closure) that adds an invisible paragraph and then fades it in

```
// define the plugin
(function($){
    $.fn.extend({
        helloWorld: function() {
            return this.each(function() {
                $(this).html('<p style="display: none" id="hello">Hello World!</p>');
                $("#hello").fadeIn('slow');
            });
        }
    });
})(jQuery);
```

part_2/examples/hello_world_plugin.html

CREATING PLUGINS

EXTENDING JQUERY

- The jQuery plugin is used like any other jQuery method:

```
// use the plugin
$(document).ready(function(){
    $("#target").helloWorld();
});
```

```
<body>
    <h2>Hello World of jQuery Plugins</h2>
    <div id="target"></div>
</body>
```

part_2/examples/hello_world_plugin.html

CREATING PLUGINS

EXTENDING JQUERY

1	Use an anonymous function to wrap your code an avoid conflicts (pass the jQuery function)
2	Use the extend method to attach the new method to jQuery
3	The plugin's name is the name of the function used as the single attribute of the object passed to extend
4	Iterate over the matched elements
5	Use jQuery to manipulate the matched elements

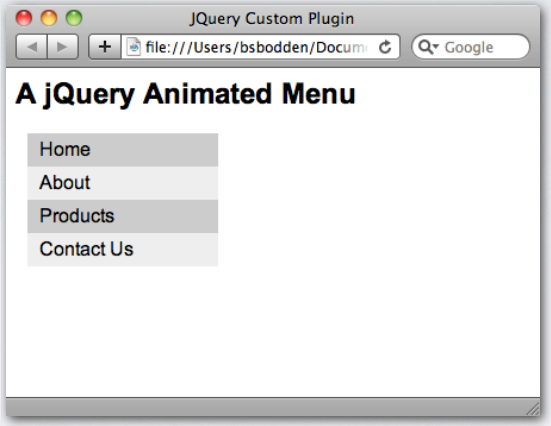
part_2/examples/animated_menu.html

```
(function($){
    $.fn.extend({
        animatedMenu: function(options) {
            var defaults = {
                animatePadding: 60,
                defaultPadding: 10,
                evenColor: '#ccc',
                oddColor: '#eee'
            };

            var options = $.extend(defaults, options);

            return this.each(function() {
                var o = options;
                var obj = $(this);
                var items = $("li", obj);

                $("li:even", obj).css('background-color', o.evenColor);
                $("li:odd", obj).css('background-color', o.oddColor);

                items.mouseover(
                    function() {
                        $(this).animate({paddingLeft: o.animatePadding}, 300);
                    }
                ).mouseout(
                    function() {
                        $(this).animate({paddingLeft: o.defaultPadding}, 300);
                    }
                );
            });
        }
    });
})(jQuery);
```

CREATING PLUGINS

EXTENDING JQUERY

- The animated menu plugin in action:



part_2/examples/animated_menu.html

LAB 2.3

CREATING PLUGINS

- In Lab **2.3** you are asked to created a custom jQuery plugin to:
  - Add an on click event to an element
  - Click on the element once will make the element grow (double in size, animated)
  - Click the element a second time and it will be restored to its original size
  - When you hover over the element it should change its opacity by 30%
  - Hint: Use `toggle` and `animate`

CSS WITH JQUERY

USING JQUERY TO MANIPULATE CSS

CSS WITH JQUERY

USING JQUERY TO MANIPULATE CSS

- In an ideal world we should strive to keep our styles away from our JavaScript! But there are times when in order to achieve clean page interactivity we must dynamically alter the styles of the page
- jQuery provides the css method to read or directly manipulate style of DOM elements

```
<p id="green">I will be green.</p>
<p id="blue">I will be blue.</p>
<button>Style This</button>
```

```
$("button").click(function(){
    $("p#green").css({"background-color":"green","font-size":"300%"});
    $("p#blue").css({"background-color":"blue","font-size":"300%"});
});
```

part_2/examples/jquery.css.html

CSS WITH JQUERY

USING JQUERY TO MANIPULATE CSS

- We use the css method to style the paragraphs:

part_2/examples/jquery.css.html

CSS WITH JQUERY

USING JQUERY TO MANIPULATE CSS

- Certain styling tasks are better handled by a combination of jQuery and CSS styles
- jQuery selectors and pseudo selectors can make it easier to determine which elements will be affected by a style and when
- The separation of concerns (and UJS) can be achieved by striving to keep styling in CSS files but applying those styles via jQuery
- Common styling techniques like transparency, mouse hovering changes, animations and dealing with tabular data are more easily and consistently achieved with jQuery

CSS WITH JQUERY

USING JQUERY TO MANIPULATE CSS

- A typical example of a styling operation that is well-suited for jQuery is to zebra-stripe a table
- The jQuery team has a nice "zebra table showdown" at http://blog.jquery.com/2006/10/18/zebra-table-showdown
- For the sake of comparison let's see what striping a table looks like with raw DOM manipulation versus a jQuery implementation

CSS WITH JQUERY

USING JQUERY TO MANIPULATE CSS

- To Zebra Stripe an HTML table we loop over the table rows and add the classes "even" and "odd" accordingly
- To achieve this using raw DOM manipulation we would have to do something like:

```
var tables = document.getElementsByTagName("table");
for ( var t = 0; t < tables.length; t++ ) {
    var rows = tables[t].getElementsByTagName("tr");
    for ( var i = 1; i < rows.length; i += 2 )
        if ( !/(^|\s)odd(\s|$)/.test( rows[i].className ) )
            rows[i].className += " odd";
}
```

- While with jQuery we can use the full power of CSS and CSS pseudo selectors:

```
$("tr:nth-child(odd)").addClass("odd");
```


- Example taken directly from http://blog.jquery.com/2006/10/18/zebra-table-showdown/
- The example above "includes a check to make sure that the 'odd' class doesn't already exist on that table row. This is taken care of by all modern libraries" including jQuery

CSS WITH JQUERY

USING JQUERY TO MANIPULATE CSS

- jQuery CSS related methods:

Method	Description
addClass	Adds one or more classes to selected elements
css	Sets or returns one or more style properties for selected elements
hasClass	Checks if any of the selected elements have a specified class
height	Sets or returns the height of selected elements
offset	Sets or returns the position (relative to the document) for selected elements
offsetParent	Returns the first parent element that is positioned
position	Returns the position (relative to the parent element) of the first selected element
removeClass	Removes one or more classes from selected elements
scrollLeft	Sets or returns the horizontal position of the scrollbar for the selected elements
scrollTop	Sets or returns the vertical position of the scrollbar for the selected elements
toggleClass	Toggles between adding/removing one or more classes from selected elements
width	Sets or returns the width of selected elements

LAB 2.4

CSS STYLING WITH JQUERY

- In Lab 2.4 use jQuery to style an HTML table:
  - Zebra stripe the table (rows with alternating colors)
  - On hover on a row change the color
  - On click make the font on the row bold
  - **Bonus:** Wrap the Zebra Striping functionality into a jQuery plugin