**Frontend Advanced**

**Module II – Exercises**

**JQUERY Challenges**

**Exercise - Styling**

Use jQuery to select the body element

Use the .css() function to style the body. Give it a red background and blue text.

**Exercise - Changing the DOM**

Use jQuery to select the body element

Use the .html() function to change the body text. Replace the DOM with a new one.

We can completely rewrite our page.

**Exercise - Append**

Use jQuery to select the body element

Use the .append() function to append a new hyperlink.

Use the .attr() function to set an href attribute on the hyperlink. Open it in your DOM and check it's clickable.

We can add elements to our DOM.

**Exercise - Make it red**

Create a web page with half a dozen paragraphs on it.

Add a click handler. When you click a paragraph it goes red.

**Exercise - Keyup**

Create a text input field. Listen for the keyup event. Alert the value of the text box whenever keyup occurs.

Get the value of an input field using:

$('input').val();

Finally, append the value of the text box into a div.

**Exercise - Rollover**

Review the list of events on: http://api.jquery.com/category/events/

Create code that listens for a hover event. When the user rolls over a div, place the word "over" in that div.

When the user mouses out of the div, place the word "out" in the div.

Count the total number of rollovers and output that in a div.

**Exercise - A big mess**

You can generate a jQuery object that contains everything on the page inside the body element like so:

$('body \*');

Write a jQuery function that iterates over everything on the page using each, makes them all position:absolute, and assigns random top and left values. You need to use **each** here to give them all different random values.

**Exercise – Hide my mail**

Assume you want to put email addresses on the Internet, but you want to keep them hidden from automated spam bots.

Rather than write your email as a link like this:

<a href="mailto:hello@example.com">hello@example.com</a>

You instead choose to obfuscate it like this:

<span class="email">hello at example dot com</span>

Use your knowledge of JQuery to rewrite the span as a clickable mailto hyperlink.

*Extra -* Put multiple different emails addresses on the page. Use the each method to ensure the code works for every email address on the page.

Note that you can use "string".replace('val', 'replacement') to replace values in a string.

**AJAX Exercise**

**Part 1 - Make the HTML**

First create a little html file containing a little bit of text and upload it to the server. Verify that you can indeed see the file by viewing it in a web browser.

**Part 2 - Do the AJAX**

Now write a Javascript file that hits that URL (note, the url will need to be local to the file, so no http:// at the start):

var url = "myfile.html"

$.ajax(url);

Use the browser console to verify that the file is being downloaded.

**Part 3 - Listen for the result**

Next add a lister. We're going to listen for 'done', so…

$.ajax(url).done(function(data) {

alert(data);

});

This function is called when the ajax request completes successfully.

**Part 4 - Do something with the result**

Finally we want to do something with the result. Lets just append it to the body.

$.ajax(url).done(function(data) {

$('body').append(data);

});

Great. We have called the ajax, listened for the result, and finally added the result to our page. Note that we don't need to just add the result to the page, we could do something else with it instead. For example, we could put it into a particular div, make it pop up, or just use it for something else.

**Exercise - Flickr**

You can Pull a Flickr image feed from the following URL:

http://api.flickr.com/services/feeds/photos\_public.gne?tags=cat&tagmode=any&format=json&jsoncallback=?

You can pull this feed using a call like this:

$.getJSON(url).success(function() {

// write the feed to the DOM here...

});

Pull the feed, then for each image, render the image and draw it to the DOM.

**Exercise - HIJAX**

Overriding the function of hyperlinks is a useful thing to be able to do. The hyperlink will work normally in the absence of JavaScript but will AJAX in the content if it is able. This technique is sometimes rather cleverly called HIJAX.

Add a list of hyperlinks to a page. When a hyperlink is clicked, detect that event, and call a function that retrieves the content via AJAX and inserts it into an element on the page.

Further Work

Add a data attribute to some of your links data-remote="true" now add the event only to hyperlinks with this data-attribute, so only links with data-remote set will use AJAX.

Add a spinner. link is clicked, display a little spinner next to the link to show something is happening. the complete event occurs, remove the spinner.

Add caching. Add a data-section-id attribute to your hyperlinks. When you click a link, first check for the presence of a div with that id. If it is present, reveal it. If not, create it, ajax the content into it, hide any other divs that may be on the page, and reveal the div that you just created.