JQUERY – LAB

INTRODUCTION

jQuery is Javascript. It is a Javascript framework that makes working with Javascript much quicker and easier for developers. jQuery has been adopted by everyone from Adobe to Apple through to Microsoft and Google.

GET THE LIBRARY

The first thing you need to do is reference the jQuery library. You can do this two ways. Download the files from http://jquery.com/ or make a reference to one of the many CDNs for the library.

Browse to http://docs.jquery.com/Downloading_jQuery.

Here you will find the latest release of jQuery. It can be downloaded in minified or uncompressed format. Uncompressed is more easily editable if you choose to edit the library (probably unlikely). The minified version is reduced in size by removing formatting and shortening variables names to give as small a file as possible.

We'll use the latest 3.x version of jQuery. The 1.x versions have support 'old' versions of IE ie prior to IE9.

- Download a copy the minified version and place it in *scripts* folder of the work files.
- open the file *index.html* from the labs zip file.
- In the `<head>` of the document add:

```
<script src="scripts/jquery-3.1.1.min.js"></script>
```

- Save and test your file

Alternatively you could use the CDN version. The benefit is that it may well already be cached in a user's browser thus speeding up your page delivery.

```
<script src="
http://ajax.googleapis.com/ajax/libs/jquery/3.1.1/jquery.min.js">
</script>
```

DOCUMENT READY

When using jQuery you need to ensure that the jQuery has been downloaded and is ready to work. When your code is placed in the `<head>` you need an event to trigger you code. This is so that references to the DOM are understood by the Javascript. Javascript has an `onLoad` event often associated with the HTML `<body>` tag whereas with jQuery we used `$(document).ready`. The code is as follow:

```
<script>
     $(document).ready(function(){
       // fire the code
     });
</script>
```

Add the following code to your *index.html* example to check that jQuery is loaded and ready to go.

```
<script>
     $(document).ready(function(){
       alert('Hello from jQuery');
     });
</script>
```

CSS SELECTORS

The neat thing about jQuery is the way it users CSS selectors to target content and then if you want to change that content in some way or other.

This can be done by using `$(selectorName)`. Change the code in the *index.html* example as follows:

```
<script>
$(document).ready(function(){
      //alert('Hello from jQuery');
      $(".lowlight").addClass("highlight");
});
</script>
```

In the above example, the jQuery targets any content of class ‘lowlight. If you have a look at the HTML you will see there is indeed a paragraph in this document with this class.

```
<p class="lowlight"> …. </p>
```

The jQuery targets this paragraph and then uses its `addClass` method to add a CSS class `highlight` which you will find declared in *styles/main.css*.

What happens here is that the class `highlight` is also added to content of class `lowlight`. The HTML is thus changed by jQuery to become:

```
<p class="lowlight highlight"> …. </p>
```

You can view this by using Dreamweaver's Live Code view.

SELECTING JUST THE FIRST() ELEMENT

In the above example the jQuery will actually look for all elements of class intro. In this sense it is very like 'vanilla' Javascript `querySelectorAll()` method. If you only require the first element then use the `first()` method ie:

```
$("p").first().addClass("highlight");
```

REMOVING CLASSES WITH REMOVECLASS()

As well as the `addClass()` method there is `removeClass()` method. Try:

```
$(".lowlight").addClass("highlight");
$(".lowlight").removeClass("highlight");
```

JQUERY EVENTS

Beyond the `ready` event jQuery has lots of events that can be responded to. As in the previous example you can use CSS selectors to attach an event to an object.

Amend your code as follows:

```
<script>
$(document).ready(function() {
	$("#showMore").click(function(){
		console.info("I was clicked")
	});
});
</script>
```

When the user clicks on `<p id="showMore">` the event will be triggered.

So we could extend this as follows:

```
<script>
$(document).ready(function(){
	$("#showMore").click(function(){
		$("#moreContent").addClass("highlight");
	});
 });
</script>
```

Try changing the event handler to `mouseover` ie

```
$("#showMore ").mouseover (function(){
//etc
```

There is also a `mouseout` event so you could extend the example as follows:

```
$("#showMore").mouseover(function(){
		$("#moreContent").addClass("highlight");
 });
$("#showMore").mouseout(function(){
		$("#moreContent").removeClass("highlight");
 });
```

THE ON() METHOD

The above examples used shorthand events. Alternatively use the jQuery `on()` method which in the case of a click event would appear as follows:

```
$("#showMore").on('click', function(){
	$("#moreContent").addClass("highlight");
});
```

THE JQUERY SHOW() METHOD

Showing and Hiding content is simple to achieve with jQuery. The CSS property `display` when set to `none` will hide content.

Notice in the HTML there is an ID selector 'moreContent'.

```
<p id="moreContent">Sed ut perspiciatis unde omnis iste natus error sit
voluptatem accusantium doloremque laudantium, totam rem aperiam, eaque ipsa
sunt explic …
```

Edit the stylesheet *styles/main.css* associated with the *index.html* file by adding a rule as follows:

```
#moreContent{
	display:none;
}
```

This will hide the content of the paragraph of ID ‘moreContent’ when the page loads.

Above this paragraph in the HTML is a `<p>` of ID ‘showMore’. Add the following jQuery to attach a click event to this content:

```
$("#showMore").on('click',function(){
		$("#moreContent").show();
});
```

The `show()` method will display content that has been hidden. It is roughly equivalent to setting the CSS on the matched HTML element to `display:block`.

Syntax

```
.show( [ duration ], [ easing ], [ callback ]
```

The jQuery show method has three option parameters. The duration of the animation in milliseconds, the easing effect applied to the animation and a function to be called once the animation has finished. Amend your code as follows:

```
$("#showMore").on('click',function(){
            $("#moreContent").show('fast', 'linear');
});
```

THE JQUERY HIDE() METHOD AND SWITCHING SHOW/HIDE WITH TOGGLE()

The `hide()` method does the reverse of the `show()` method. It again takes three optional parameters. This would be useful for hiding the content. With our current code all we could do would be add another bit of HTML content to do call the `hide()` method. However, there is an easier way because the jQuery `toggle()` event will 'toggle' an element between `show` and `hide`.

```
$("#showMore").on('click', function(){
            $("#moreContent").toggle("fast");
});
```

Again we want to feedback to our user to indicate what clicking on `#showMore` will do. Conveniently the `toggle()` method has a call back function that we can call once the 'toggle' is completed.

Change your code as follows:

```
$("#showMore").on('click', function(){
            $("#moreContent").toggle("fast", function(){
                  if($(this).is(':hidden')){
                        $("#showMore").text("Show Details");
                  }else{
                        $("#showMore").text("Hide Details")
                  }
            });
});
```

The above uses the jQuery `:hidden` selector to detect whether the element has a `display` property set to a value of `none`. This was feed to the `is()` method that can be used to check elements against given parameters.

NAVIGATING THE DOM - PARENT / CHILDREN / PREV / NEXT

The `parent()` method can be used to the target the parent of an object - that is the HTML element that encloses the selected object. Therefore given the HTML structure of:

```
<ul>
       <li><a href="#">Sheffield Wind Farm</a>…</li>
       <li><a href="#">Bradford Water Mill</a>…</li>
       <li><a href="#">Yorkshire Solar Centre</a>…</li>
</ul>
```

The `<ul>` is the parent of the `<li>`. The code below would target the parent of the <li> clicked that is the `<ul>`, so the class 'highlight' is added to the `<ul>`.

```
$("li").click(function(){
      $(this).parent().addClass('highlight');
});
```

The jQuery `this` keyword refers to the current object and this makes the code more flexible.

Amending the code above we could traverse the DOM with `next()`. This looks for the next sibling of the current element.

```
$("li").click(function(){
      $(this).next().addClass('highlight');
});
```

To test, click on the middle link of the three. Changing the code to use `prev()` works back up the siblings.

PREVENTDEFAULT()

In the above examples when clicking on the `<li>` we are also clicking the `<a>`. We can refine the above using:

```
$('.sideMenu li a').on('click', function(ev){
            ev.preventDefault();
}
```

In the above the event is set an 'event' object given the name 'ev'. This is then run against the method `preventDefault()` which will stop the links from linking to the null link of `#`.

TOGGLE VISIBILITY

In `<div class="sidebar">` you will notice each `<li>` contains a `<div class="infoPanel">`. The class 'infoPanel' has the property `display:none`. We can toggle this property for each individual list item with the following:

```
$('.sideMenu li a').on('click', function(ev){
            ev.preventDefault();
            $(this).next('.infoPanel').toggle();
});
```

In this instance the `next()` method looks for next element in the DOM of class 'infoPanel'.

USING THE ATTR() METHOD TO CHANGE ATTRIBUTES

The following code can be applied to the image in the file. This uses `prev()` to target the image on mouseover/mouseout on the `<figcaption>` element. The `attr()` is used to change the `src` of the image to change the source of the image.

```
$("figcaption").mouseover(function(){
            $(this).prev('img').attr('src', 'images/SolarPanelsZoom.jpg');
 });
$("figcaption").mouseout(function(){
            $(this).prev('img').attr('src', 'images/SolarPanels.jpg');
 });
```

HASCLASS()

The jQuery `hasClass()` method is used to check whether an object has a named class attached to it.

Don't forget we can use all our Javascript knowledge as well such as conditional logic, so we could amend the code as follows:

```
$('.sideMenu li a').on('click', function(ev){
            ev.preventDefault();
            if($(this).hasClass('highlight')){
                  $(this).removeClass('highlight');
            }else{
                  $(this).addClass('highlight');
            }
      });
```

When the text in the `<li>` is clicked we now check to see if the 'highlight' class has been applied and it not apply it. If it does we remove it. If not we add it.

This kind of conditional logic is used a lot so jQuery simplify things with a `toggleClass()` method.

```
$('.sideMenu li').on('click', function(){
      $(this).toggleClass('highlight');
});
```

Adding a `next()` method, as the <a> is a sibling to `<div class="info">` we can use:

```
$('.sideMenu li a').on('click', function(ev){
      ev.preventDefault();
      $(this).toggleClass('highlight');
      $(this).next('.infoPanel').toggle();
});
```

CONTENT SLIDEUP() AND SLIDEDOWN() METHODS

The `slideUp()` method provides a shortcut to animate an element to `display:none`, sliding the content up in the process. The `slideDown()` does the reverse, whilst `slideToggle()` will toggle the slide dependent on whether the content is visible or not.

Syntax

```
.slideUp/slideDown/slideToggle( duration, opacity, [callback] )
```

- **Duration** – duration of animation in milliseconds or a string such as 'fast' (200) and 'slow' (600).
- **Opacity** - A number between 0 and 1 denoting the target opacity.
- **Callback** [Optional] - A function to call once the animation is complete.

Amend the previous example as:

```
$('.sideMenu li a').on('click', function(ev){
      ev.preventDefault();
      $(this).toggleClass('highlight');
      $(this).next('.infoPanel').slideToggle('fast');
});
```

CONTENT FADEIN() AND FADEOUT() METHODS

The `fadeTo()` method provides a shortcut to animating the CSS opacity property.

Syntax

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

- **Duration** – duration of animation in milliseconds or a string such as 'fast' (200) and 'slow' (600).
- **Opacity** - A number between 0 and 1 denoting the target opacity.
- **Callback** [Optional] - A function to call once the animation is complete.

The `fadeOut()` method provides a shortcut to animating the CSS opacity property to zero. In addition the opacity property has reached zero the CSS property 'display' of the targeted element is set to 'none'.

Syntax

```
.fadeOut( [duration], [callback] )
```

- **Duration** [Optional] – duration of animation in milliseconds or a string such as 'fast' (200) and 'slow' (600). If omitted then a default value of 400 is used.
- **Callback** [Optional] - A function to call once the animation is complete.

Add the following content to the top of your *index.html* file immediately after the `<body>` tag.

```
<div id="welcomeBanner">
<div><img src="images/closebutton.png" width="45" height="45" alt="Close
Button" /></div>
<p>Thanks for visiting our site.  We hope you like the improvements.</p>
</div>
```

Then add the following to the jQuery.

```
$("#welcomeBanner img").click(function(){
            $("div#welcomeBanner").fadeOut();
 });
```

Notice the use of a CSS selector to assign the event to the HTML img element inside of the ID selector ‘welcomeBanner’.

FORM RELATED EVENTS

When a user clicks inside a HTML form field a javascript 'focus' event is triggered. When a user takes their cursor focus away from a HTML form element this triggers a 'blur' event.

We'll amend the 'sign up' form to add some form validation. Add the following error message to the HTML form.

```
<form action="go.html" novalidate>
    <label for="emailSignUp">Sign Up</label>
    <input type="email" placeholder="you@email.com" name="signUp"
id="emailSignUp"><input type="submit" value="Join Up">
    <p class="error">* Sorry email needed</p>
</form>
```

In the CSS there is a rule to hide the `<p class="error">`. The HTML form also has the attribute `novalidate` that will switch off any browser specific HTML5 validation as we don't want these to conflict with our custom validation.

For form elements the `val()` method is used to retrieve their value. We will therefore check to see if a value has been submitted.

When the user clicks inside the input form element and then outside without adding a value we can trigger an event.

```
$('input[type=email]').on('blur', function(){
      if($(this).val()){
            $('.error').hide();
      }else{
            $('.error').fadeIn(100);
      }
});
```

VALIDATION ON FORM SUBMISSION

The form validation can be called when the form is submitted with the `submit` event. To prevent the form from been submitted a `return` value of `true` or `false` is added. If the function returns `true` then the form is submitted.

```
$('form').on('submit', function(){
	if($('input[type=email]').val()){
		return true;
	}else{
		$('.error').fadeIn(100);
		return false
	}
});
```

VALIDATING EMAIL

To add proper email validation we need a little online help. A quick search online produced this algorithm which checks to see if a value is formatted as an email. The algorithm is using regular expressions. Regular Expressions are a way of matching text based on certain rules and are understood by Javascript.

```
var emailRegEx = /^[\w][\w\._\-]*@(?:[\w]+\.?)*[\w]\.[a-zA-Z]{2,4}$/;
```

Found at: http://www.alperguc.com/en/regular-expressions-regex/javascript.html

We can use this with the Javascript `search()` method. The `search()` method searches for a match between a regular expression and a string. If none is found it will return -1. Thus the following will check to see if a value is an invalid email address.

```
var emailRegEx = /^[\w][\w\._\-]*@(?:[\w]+\.?)*[\w]\.[a-zA-Z]{2,4}$/;
if(someEmail.search(emailRegEx)==-1){
	// Not valid
}else{
	// valid
}
```

Can you amend the code so that it does valid the user input as an email?

MOBILE MENU

Place the following HTML at the top of the page after the `<body>` tag.

```
<div id="navButton"><i class="fa fa-bars"></i></div>
```

There is already a CSS rule in the *styles/main.css* that hides this menu, but a rule in the *styles/mobile.css* that reveals it.

In the Javascript add the following to slide / toggle the menu.

```
$('#navButton i').on('click', function(){
      $('nav').slideToggle('fast');
});
```

Experiment with `fadeToggle()` and `toggle()` to see which effect you think works best. You could also investigate the jQuery `animate()` method to slide the menu in from the side.