JQUERY – LAB

INTRODUCTION

jQuery is Javascript. It is a Javascript library that makes working with Javascript much quicker and easier for developers. jQuery is widely used in web community though eventually it is likely to be made redundant by better Javascript and CSS features.

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="js/jquery-3.6.0.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/3.6.0/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>
```

As with vanilla Javascript we can also store the code in an external file. Create a file of *js/main.js* and add the code from above (minus the script tags) and attach it to the index.html page with:

```
<script src="js/main.js"></script>
```

It is important to add this after the call to the jQuery library.

CSS SELECTORS

The neat thing about jQuery is the way it uses 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 *js/mains.js* example as follows:

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

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/mobile.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 the Inspector in Chrome.

SELECTING JUST THE FIRST() ELEMENT

In the above example the jQuery will actually look for all elements of class `lowlight`. 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 in *js/main.js* to include the follows:

```
      $("#showMore").click(function(){
            console.info("I was clicked")
      });
```

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

So we could extend this as follows:

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

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/mobile.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.

THE 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.

MOBILE MENU

Amend the HTML at the top of the page after the `<header>` tag to include a div with an image in it to act as a ‘burger’ menu.

```
<div id="logo">
          <div class="burger">
            <img src="images/whiteMenu.svg" alt="Menu" />
          </div>
          <h1><a href="#">Your Name</a></h1>
</div>
```

This is set to be hidden on the desktop view through the *desktop.css* rule:

```
.burger {
  display: none
}
```

There is a rule in the *styles/mobile.css* that reveals it as well as applies some styles.

```
.burger {
  display: block;
  margin: 15px 0 10px 5px;
  width: 5%;
}
```

We can attach an event to the `div.burger` to slide the nav up and down with the following code:

```
$(".burger").on("click", function () {
    $("nav").slideToggle("fast")
  });
```

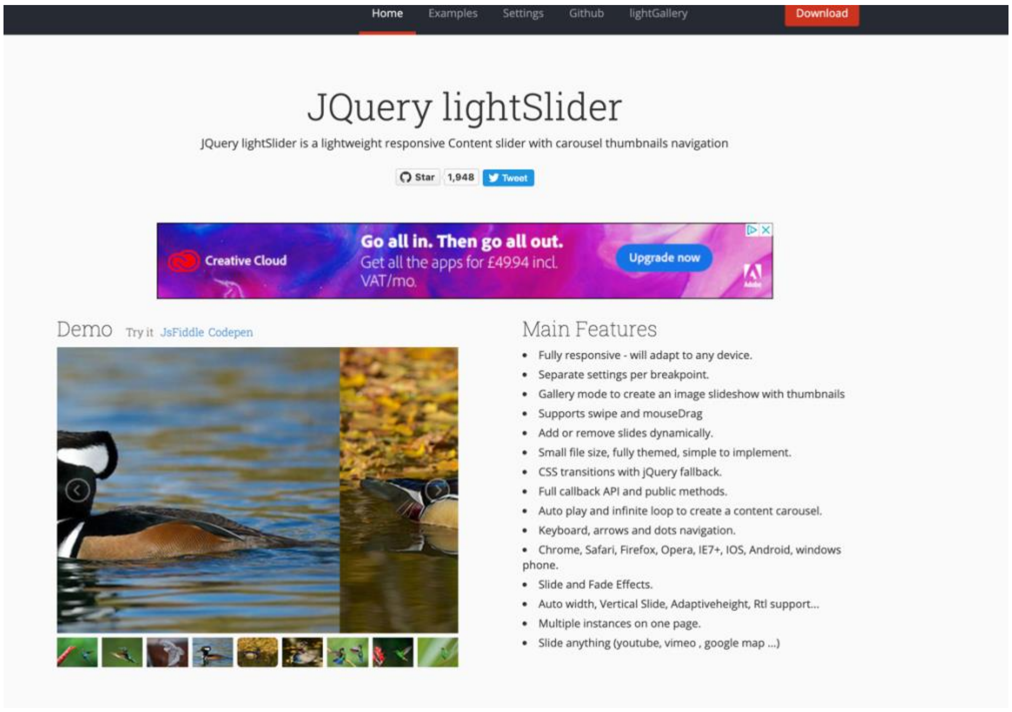
Experiment with `fadeToggle()` and `toggle()` to see which effect you think works best. You could also investigate the jQuery `animate()` method to slide the menu in from the side.

WORKING WITH PLUGINS

There is a vibrant community of jQuery developers who produce ‘plugins’ for use in web pages. These can perform all kinds of complex functionality. Image galleries and sliders are popular plugins.

In this example we’ll make use of the jQuery ‘lightslider’ plugin found at:

http://sachinchoolur.github.io/lightslider/index.html



JQuery lightSlider

JQuery lightSlider is a lightweight responsive Content slider with carousel thumbnails navigation

Main Features

- Fully responsive - will adapt to any device.
- Separate settings per breakpoint.
- Gallery mode to create an image slideshow with thumbnails
- Supports swipe and mouseDrag
- Add or remove slides dynamically.
- Small file size, fully themed, simple to implement.
- CSS transitions with jQuery fallback.
- Full callback API and public methods.
- Auto play and infinite loop to create a content carousel.
- Keyboard, arrows and dots navigation.
- Chrome, Safari, Firefox, Opera, IE7+, IOS, Android, windows phone.
- Slide and Fade Effects.
- Auto width, Vertical Slide, Adaptiveheight, Rtl support...
- Multiple instances on one page.
- Slide anything (youtube, vimeo , google map ...)

With any plugin you will need to read the documentation to understand how the plugin is used and what options it has. They will often require you to download CSS and JS files for use in your project. For this lab we’ve already downloaded the necessary CSS and JS files for the plugin to work in our web page.

Amend the `<head>` of the *index.html* page to load the CSS and JS files required by the plugin.

```
<link rel="stylesheet" type="text/css" href="styles/mobile.css" />
    <link
      rel="stylesheet"
      media="only screen and (min-width:720px)"
      href="styles/desktop.css"
    />
    <link rel="stylesheet" type="text/css" href="styles/lightslider.css" />
    <script src="js/jquery-3.6.0.min.js"></script>
    <script src="js/lightslider.js"></script>
    <script src="js/main.js"></script>
```

The *lightslider.css* and *lightslider.js* files are now loaded into the page.

Add the required HTML for an image slider.

Remove:

```
<p>
 <img src="images/dishOne.jpg" alt="Big Dish" width="900" height="200"
class="resize-img"/>
</p>
```

And replace it with:

```
<div class="sliderContainer">
      <div id="lightSlider">
             <div><img src="images/dishOne.jpg" alt="Dish One" /></div>
             <div><img src="images/dishTwo.jpg" alt="Dish Two" /></div>
             <div><img src="images/dishThree.jpg" alt="Dish Three" /></div>
             <div><img src="images/dishFour.jpg" alt="Dish Four" /></div>
             <div><img src="images/dishFive.jpg" alt="Dish Five" /></div>
     </div>
</div>
```

This will provide 5 images for the image slider to loop around.

Amend *js/main.js* with the following:

```
$("#lightSlider").lightSlider({
    item: 1,
    slideMargin: 0,
    loop: true,
  });
```

Notice how the code targets the `div#lightSlider` and applies the `lightSlider()` method to it. The settings uses are outlined in the Javascript object `{…}` syntax.

CHALLENGE

1. Review the lightSlider document to experiment with slider settings.
2. Investigate other slider plugins to see if they could be incorporated into your site.