COSC 360
Lab 6 – jQuery

This is an individual assignment. This assignment is marked out of 20 points.

Due Date: March 14, 2021.

Information:

Using the provided HTML, JS and CSS files and images, complete the following tasks using jQuery. jQuery is a powerful JavaScript framework that started as a concise set of selector mechanisms and continues to add new features such as animation and parsing capability. In this lab, you will be extending JavaScript using jQuery, manipulating the DOM and handling events using jQuery and make asynchronous requests in JavaScript using jQuery.

You will need to inspect the HTML code of determines the correct `class` and `id` values for different elements in this lab.

Instructions:

Attempt to reproduce the page presented in Figure 1 as possible, noting the marking rubric.

Part I: DOM Manipulation with jQuery

1. View the provided HTML5 code and images. Do not change the directory structure. You will be able to view the page without a webserver. Examine the `lab6-1.html` file to understand how specific elements have been identified. Figure 1 shows the completed page for part I of the lab for reference.

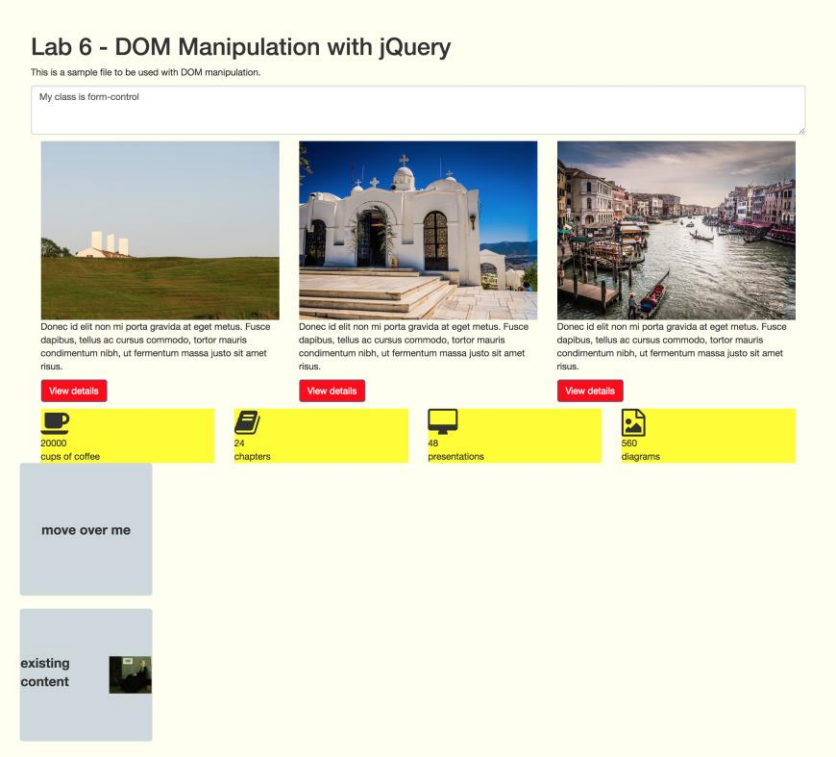

Figure 1 - Completed part I page

2. In order to use jQuery, you will need to include the jQuery JavaScript library. The library can be included from a content delivery network (CDN) or included from a local source. The most prudent strategy is to attempt to pull the library from the CDN, but resort to a locally hosted version in the event the CDN is unavailable.

   Add code to load `jquery-3.1.1.min.js` from the Content Delivery Network in the `<head>` section of your page.

3. As the CDN may not always be available, add a local fail-safe version of the script that can be loaded locally in the `<head>` of the page.
4. Add the appropriate tags so that `lab6-1.html` will utilize `lab6-1.js` (located in the `js` folder). Add all your JavaScript and jQuery code in this file (`lab6-1.js`) for part I. As this script will modify the DOM, ensure it is included in the correct location in your html file.
5. Utilizing jQuery, update the page title (identified by the id `pageTitle`) to **be Lab 6 – DOM Manipulation with jQuery**.
6. Utilizing jQuery, update the `textarea` with the id of `msgArea` with a message that prints out the class name for the `msgArea` as shown in the Figure 2 (hint: use `attr` to access the attributes of the element – just don't hard code it).

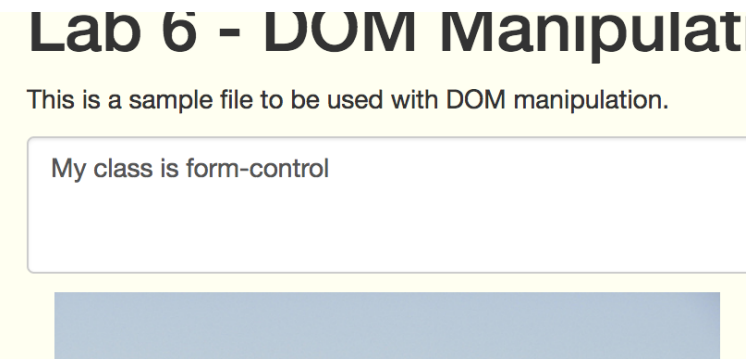

Figure 2 - Text Area update

7. Utilizing jQuery change the background colour of the '**View details**' buttons to be `red`. You do not need to loop through the collection of jQuery nodes as you might do with JavaScript; one of the powerful features of jQuery is that the jQuery function always returns a set of jQuery nodes and thus all functions operate the same whether there is one or many nodes.
8. Utilizing jQuery change the background colour of the `body` of the page to be `ivory`.
9. Utilizing jQuery, update items that are of the class `center-icon`, to include the class `selected` (which is this case will highlight them `yellow` as provided by the CSS).
10. Utilizing jQuery, add a click event handler to the `panel` class such that when you click on the panel, the `span` with the id of `message`, will update its contents to read **You clicked this panel** (Figure 3)**.** Test to ensure that this is working correctly.

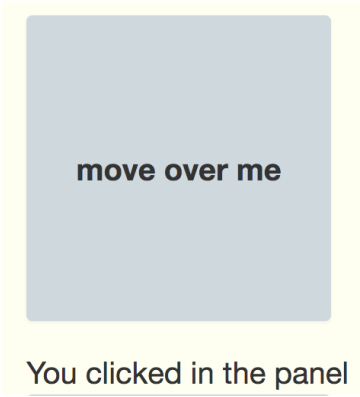

Figure 3 - Panel click message

11. As with JavaScript, you can use jQuery to bind several events together. Create the following events for the `panel`:
    a. When you move the mouse over the panel, update the `message` such that it prints out the x and y co-ordinates of the mouse in the page (Figure 4).

Figure 4 - Mouse x and y location

    b. When the mouse leaves the panel, update the `message` such that it prints out **The mouse has left** (Figure 5).

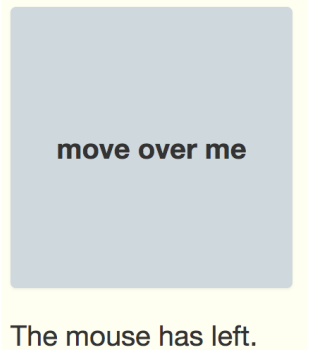

Figure 5 - Mouse has left message

12. Utilizing jQuery, create a new `img` using the image `images/art/thumbs/13030.jpg` and have the image display within the element with an `id` of `panel-2`.
13. Using jQuery, create a floating preview for the three main images on the page. When you mouse over the image, the code will dynamically add a `<div>` that contains a larger version of the image the mouse is over at the end of the document. You will need to create a new node (**with the `id` of preview which allows you to use the prebuilt styling**). In the new preview node, you will need to set

the `src` for the image to use the **medium image**. Here is some helper code that will build the node for you (assume that this is contained in an anonymous function):

```
// construct preview filename based on existing img
var alt = $(this).attr('alt');
var src = $(this).attr('src');
var newsrc = src.replace("small","medium");

// make dynamic element with larger preview image and caption
var preview = $('<div id="preview"></div>');
var image = $('<img src="' + newsrc + '">');
var caption = $('<p>' + alt + '</p>');
```

Once the node is constructed, you will need to place it in the correct position in the DOM such that it will be viewable as shown in Figure 6.

When the new node is to be displayed, **make the small image gray** (using the `gray` class) and then fade the preview in over 1000 milliseconds.

You will need to create a function that will remove the preview when the mouse leaves the image and remove the `gray` class from the original image. This function will need to be bound to images for a `mouseLeave` event. You can target the images using the selector `#stories img`.

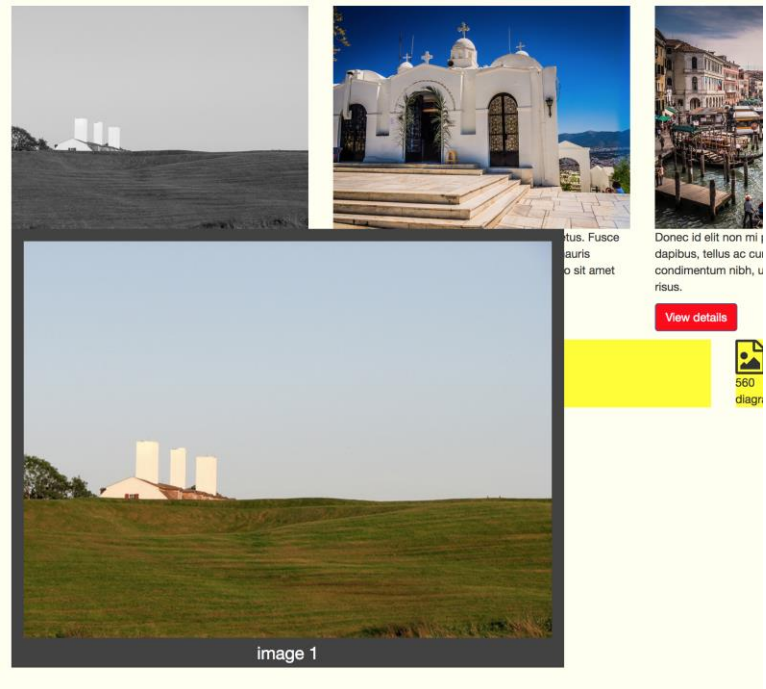

Figure 6 - Image preview

Part II: AJAX and jQuery

In this part of the lab, you will examine how to use jQuery with AJAX to asynchronously update content.

1. Open up the file `lab6-2.html` and examine the contents. This file uses two different techniques to create dynamic content, which in this case is to display the time.

   The page functions by using two JavaScript timers, both of which are set to interrupt every second. When the timer expires, JavaScript will call the associated callback function (`timer1()` and `timer2()` in this example). You can find the jQuery calls along with vanilla JavaScript to perform the same tasks.

   When `timer1()` is called, it will get the date from the system and updates the element with the current time. With `timer2()`, a different approach is used. Here, when the callback function is executed, the function contacts the server at the specified `url` and wait for the `data`. In this case the site returns a JSON object which the code proceeds to process, extracting the time key out, and updating the second time value. A JSON object is a set of data that is constructed with key-value pairs. With jQuery, when this data is returned, you can access the key-value pairs by using the . operator. For example, when the call is made to http://time.jsontest.com/, the following is returned:

```
{
   "time": "08:37:50 PM",
   "milliseconds_since_epoch": 1519072670918,
   "date": "02-19-2018"
}
```

   jQuery will store this as a JavaScript object, thus you can access the members (i.e., `time`) using the dot (`.`) operator followed by the key name (`data.time`).

   This technique can be used extensively to have a page pull and display new content without having to reload the page entirely (much to the happiness of the user). The callback function also contains the logic to deal with a situation where there may be issues with the connection.

   If you open up the console for your web browser, you will see the results of the async call being printed to the console (to get a better feel of what is going on).
2. Examine the JSON object returned by the url

   `http://www.randyconnolly.com/funwebdev/services/travel/cities.php`

   This produces a JSON object, with information about cities. Add a button to `lab6-2.html` and called it '**Get City Info**'. Using jQuery, create the code that will retrieve the information from this page asynchronously when you click the button. Add your jQuery and JavaScript to the existing script block. **Format the results to display in a table using correct headers for each column.**
3. Add additional functionality to the button to handle conditions where the data can't be retrieved (i.e., bad connection).

Evaluation Criteria:

	Part I
1 point	jQuery from CDN
1 point	JQuery from local

1 point	JS file included for lab6-1
1 point	Update of pageTitle with jQuery
1 point	Update of textarea with jQuery
1 point	Background color of View details set with jQuery
1 point	Body color set with jQuery
3 points	Panel updates with jQuery
1 point	Image display in panel-2 with jQuery
4 points	Floating preview images (image in position, fade-in, change in img colour and the mouseleave
	Part II
4 points	jQuery + AJAX to display cities in table async
1 point	Error handling for bad async connection