# Assignment: jQuery and HTML

This assignment has two parts. Both parts require you to write JavaScript and jQuery that changes the appearance an HTML web page.

# Part 1 Together as a class: Peekaboo Puppy

1. Retrieve assignment20\_puppy.zip provided by your instructor.
2. Extract the code into your mis2402workspace and open the index.html file in Visual Studio Code.
3. Preview index.html in Chrome. You will notice that the page has nicer margins than previous assignments. It also appears to have two columns.



Better margin

2 columns

1. The reason for this improvement is that the HTML is now inside a <div> with a class of “container”… and the container then holds div tags of class row and col-md-6.

All those are classes are defined by Bootstrap. Recall that Bootstrap is a library of pre-defined CSS classes. So, what we have, is a combination of those tags to create this effect:

 

You can get a good review of Bootstrap here: <https://www.w3schools.com/bootstrap4/>

1. Notice the click event handler for button 1. It adds two Bootstrap classes to the <div> identified by #sometag. Those classes are “alert” and “alert-primary”. The jQuery addClass method allows us to add one or more CSS classes, separated by spaces.

Also, in this click event handler we remove any previously existing classes. We do that with the jQuery removeClass method. The removeClass method, with no arguments, removes all the classes from the selected tag.

1. Amazingly, Bootstrap has a variety of classes used for styling tags. The Bootstrap class names follow a pattern – primary, secondary, info, warning, danger and success. So, an “alert” can be styled with “alert-primary”, “alert-secondary”, “alert-info”, “alert-warning”, “alert-danger” and “alert-success”. There are similar patterns for “text” (e.g. “text-success”) and buttons (e.g. “button-success”)
2. Create a click event handler for button 2. It should style the tag as “alert alert-secondary”.
3. Create click event handlers for the remaining buttons.
4. Create a click event handler for the image puppy1.
5. Create a click event handler for the image puppy2.
6. Use the show() and hide() methods discussed in class so that the images appear and disappear with each mouse click.

HINT: Here is an animated example that illustrates the effect you want to achieve:

<http://misdemo.temple.edu/gifs/peekaboo-gif.gif>

1. Upload your work. Be sure that you can find your work on the class server by typing in its URL in the browser. Test your work \*again\* on the class server.

For example:
http://misdemo.temple.edu/tux99999/assignment20\_puppy

# Part 2 On Your Own: Find the Owls, Again!

**THE PROBLEM:** Write a program allows a player to search for owls that are hidden in a 5 x 5 grid. When 5 owls are found, let the player know that they have won.

1. Retrieve assignment20\_owls2.zip provided by your instructor.
2. Extract the code into your mis2402workspace and open the index.html file in Visual Studio Code.
3. Inspect the code.
	1. In the HTML portion of the file, take note of how each individual table tag is identified with an id attribute.
	2. Notice that there is a JavaScript variable called gameover. This variable has an initial value of false. But, when the game is over, it should be set to true. Your code should use this variable to prevented the user from taking any further action.
	3. Notice the JavaScript function called getOwls. This function will return an array with 5 items in it. Preview index.html in Chrome and, using the Debug Console, manually run getOwls a few times. Do this so you can be comfortable with how this function works. *Do not modify the* getOwls *function.*



1. Get a feel for how the whole solution should work. The game is simple and intuitive. But you might want to watch this short video to understand what’s expected.
See: <https://youtu.be/LWVEMEK8ZUE> (this video has no audio)
2. Your assignment is to complete the checkTable function and the owlCounter function. Here are the details for the checkTable function.
	1. If the user enters anything other than an integer between 1 and 25 then the function should return “Bad data. Try again.”
	2. To determine if a user has made a good or bad guess, you will need to check each element of the array called owls. The easiest way to do this is with a loop.
	3. If the user enters a ***bad*** guess, then use jQuery to set **&#x274C;** as the inner html of the appropriate <td> tag. This is the HTML entity for a red X.

**FUN FACT:** For more cool / fun characters and emojis, see: [https://www.w3schools.com/charsets/ref\_emoji.asp](https://www.w3schools.com/charsets/ref_emoji.asp%20%20)

For a bad guess the function should return “Keep trying…”

* 1. If the user enters a ***good*** guess, then use jQuery to set **<img>** as the inner html of the appropriate td tag. The HTML img tag has already been styled to appear as a cute owl. The five characters **<img>** are all you need.

For a good guess, the function should return “Hoo! Hoo!” unless, of course, all 5 owls have been found. If that is the case then return “You Win!” and update the gameover variable to indicate that the game is over.

Once the game is over, write code to ensure that the user can’t keep on playing.

You should use the owlCounter function to determine if the game is over or not.

**HINT:** Get everything else working to your satisfaction before you try to deal with the “game over” condition.

1. Here are the details for the owlCounter function.
	1. This function will require a loop. Use the loop to inspect each <td> tag on the page. If the inner html of the <td> tag equals “<img>” then it counts as an owl.
	2. Return the number of owls found on the page.
	3. Taking this approach prevents an owl from being double counted. If I find an owl in box number seven. And then I enter seven again, I have still only found one owl. Counting the owl in box seven more than once would be double counting!
2. Test your work. Does the behavior of your solution match the YouTube video from step 5?
3. Upload your work. Be sure that you can find your work on the class server by typing in its URL in the browser. Test your work \*again\* on the class server.

For example:
http://misdemo.temple.edu/tux99999/assignment20\_owls2

How will this assignment be graded?

|  |  |
| --- | --- |
| Item | Point Value |
| All folders uploaded OK?(solutions must be ***exactly*** where requested!) | 10 |
| Part 1 works completely as expected. | 20 |
| Part 2  | -- |
| Report “Bad data. Try again.” If box number is blank or not a number. | 10 |
| Report “Bad data. Try again.” If box number is out of bounds. | 10 |
| Report “Keep playing…” as expected. | 10 |
| Report “You Win!” as expected. | 10 |
| Cells are populated with “X” as expected. | 10 |
| Cells are populated with the owl image as expected. | 10 |
| Code works. No irregularities in game play. This includes preventing game play when the game is over. | 10 |