# *Web Programming II (420-C20-HR)*

# *Lab 12 – jQuery 1*

Date assigned: Monday, April 25, 2016

Date due: **Monday, April 25, 2016, 11:00 a.m.**

**Learning Objectives**

Upon successful completion of this lab exercise, the student will be able to:

* Do some beginning work with jQuery

Lab Set-Up

1. All formatting and styling must be done using HTML5 and CSS3. All CSS formatting characteristics must be kept in a separate file in the assets folder of your website.
2. For all the web pages you create, ensure that you give each page a title and provide an appropriate header and some formatting. Don’t spend a lot of time, but spend some time making the page look nice with background, font and other CSS formatting.
3. All JavaScript functions MUST be kept in a separate file and attached to the html file using the appropriate method.
4. For this lab you can use whatever environment you wish to use, Dreamweaver, Expression Web, Visual Studio or other.
5. Copy the files from Moodle to a folder called H:\420-C20\Labs \YourUserName\_C20\_L12. Use your tool of choice and create a new site called C20\_L12. Store the site in the folder H:\420-C20\Labs \YourUserName\_C20\_L12. All the files you create for this lab must be stored as part of this new site.
6. Add the following script tag BEFORE the script tag containing your own JavaScript in each html file you create.  
   <script src="https://code.jquery.com/jquery-2.2.3.min.js"></script>  
   You can copy and paste the line…

To do:

**Part A – Guessing Game**

1. In this section you will create a guess the number game in JavaScript and use some jQuery to change the look of things. The game waits 10 seconds between guesses and then displays a warning message asking the user to continue.
2. Create a file called parta.html. Inside add appropriate heading labels (such as “Guessing Game” or something) and a form with a single text field with an appropriate label and a button with the value “Guess”. After the form add three divs one with the ids: msg, guesses and theCount respectively (or one div with three paragraphs with those ids).
3. Link jQuery using the script tag above and a JavaScript file called parta.js that you create as follows:
   1. First, add the timeout by adding a call to a function called timeIsRunningOut() after the file is loaded. Use the jQuery document ready event to call the function.
   2. Create a JavaScript function called timeIsRunningOut() which assigns the **global** variable called theTimer to the setTimeout function. The setTimeout function calls a function called Continue() every 10 seconds (10000 milliseconds)
   3. Create a JavaScript function called continue() which prompts the user with a confirm box with the message “You haven't entered a number in 10 seconds. Do you want to continue?” If the user selects OK (the confirm function returns true) then called the function timeIsRunningOut to reset the timer. If the user selects Cancel from the confirm box, do not set the timer.
   4. Save the file and test it in the browser to make sure it times out every 10 seconds until you press cancel.
   5. Create global variable called myNumber at the top of the JavaScript section and use Math.random (and Math.floor or Math.ceil) to create a random number between 1 and 100.
   6. Add a click event handler to the button to call the function isItRight(). This does NOT need to be done with jQuery but can be if you want to figure out how to do so.
   7. Create a JavaScript function called isItRight() which does the following
      1. Stops the 10 second timer set in the timeIsRunningOut function.
      2. Gets the value of the guess field from the form and compares it to the myNumber variable. Once again, this can be done without jQuery; however, if you wish to use jQuery use the jQuery function .val() to get the value of a form selector.
      3. If the guess equals the number, use jQuery to set the value of the msg div to “That’s right!!” (Use the .text function)
      4. Else, if the guess is less than the number set the value of the msg div to “Too Low”, set the focus to the guess field and call the timeIsRunningOut function to reset the timer.
      5. Else (the guess must be greater than the number) set the value of the msg div to “Too High”, set the focus to the guess field and call the timeIsRunningOut function to reset the timer.
      6. Use the JQuery append function to append a comma (,) plus the value of the current guess to the div guesses. Add logic using the jQuery text function to only add the comma
4. Save the file and test it in the browser. You should have a working guessing game where the computer selects a number from 1 to 100 and the user has to guess the number.
5. Add a validation function to ensure that the value entered by the user is an integer (a number) greater than zero and less than 100. Provide an error message in a separate div above the form if the entered value is not valid.
6. Add a counter which counts how many guesses it takes the person and display that in the div/paragraph theCount. Use the jQuery .text or .html method to update theCount after each guess. Do not count errors or invalid entries. (Bonus points if you count the invalid entries separately and display the count of those in a separate div/paragraph using jQuery).
7. Use the jQuery addClass and removeClass methods and/or the jQuery css method to change the style (colour, size, font, etc) of the result paragraph depending on whether the guess is correct, too high or too low.

**Part B – jQuery More**

1. Open the file partb.html from the folder you copied at the beginning of the lab. The file contains a series of simple headers and paragraphs along with three images. You will use jQuery to modify these. DO NOT CHANGE ANY CSS FOR THESE FILES.
2. Add a script tag to link to jQuery before the final body tag and a link to a JavaScript file called partb.js immediately after it.
3. You are going to add a series of buttons to the page, below the HTML tags and above the <script> tags. The buttons do NOT have to be in a form. You will be adding event listeners for each button click, but instead of calling a separate function, use an inline function in the add event listener to perform the operations listed.
4. Add a button to the bottom of the page with the value Add Class. The click event, uses the jQuery addClass function to add the class ‘myclass’ to the first paragraph of the HTML document.
5. Add a button with the value Remove Class which removes the class ‘myclass’ from the first paragraph using the jQuery removeClass function.
6. A button with the value Add Title to add the title attribute with the text “No title provided” to all images if and only if the image does NOT already have a title attribute. (YES this CAN (and must) be done in one line of jQuery (think of “not” on the selector)).
7. A button with the value Add Indicator which adds the text “Heading: ” to the beginning of every heading (h1-h6) element.
8. A button with the value Add Line which selects all the even numbered paragraphs and adds a horizontal line (tag <hr/>) after each even paragraph.
9. Now let’s introduce some text effects… Add a button with the value Toggle Paragraphs which applies the slideToggle function in a “slow” manner to all the paragraphs (note, if you put the buttons in a paragraph they will disappear as well. You may want to put your buttons in their own div instead of paragraphs).

The slideToggle function which can take one parameter which is a duration either in milliseconds or using the strings “fast” or “slow”. The first time the function is called the selected elements are “slid” out and the second time, they are “slid” back in.

So, clicking the button above once will make all the paragraphs disappear and clicking it a second time will make them re-appear.

1. Add a button with the value Fade which selects the last paragraph (do not count, use the appropriate selector) and uses the jQuery fadeTo function to do a fast fade to .1 opacity.

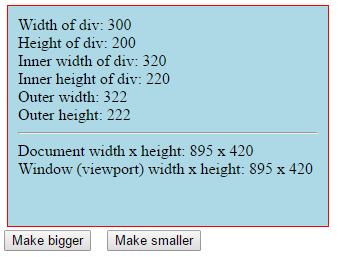
The fadeTo function takes two parameters, the first being the duration either in milliseconds or using the strings “fast” and “slow”. The second parameter is the opacity to which the selected element(s) are faded to.

**Part C – One More jQuery Piece**

1. Now a quick example of other jQuery functionality. jQuery lets you measure the height and width in various ways corresponding to the box model. In this next example you will create a div with a margin and padding and border and then measure its dimensions.
   1. Create a file called partc.html and link the jQuery library and a JavaScript file called partc.js immediately after it.
   2. In the HTML file, add a div with the id myDiv. Include a button with the value “Make bigger” and a button with the value “Make smaller”.
   3. Add an embedded style tag to the head of the document to format the id myDiv as follows: height 200px, width 300px, padding 10px, margin 3px, border 1px, solid and red, background colour lightBlue (you can choose other colours if you like).
   4. After the document is ready, add an event listener to the div myDiv which calls the function doMeasure. Then add the function doMeasure to:
2. Initialize a variable (called myTxt) to an empty string
3. Using the jQuery width() function, select the myDiv div and add to the myTxt variable the string “Width of div” and the jQuery value of the width and a line break (<br/>).

Note: Each of the following has the corresponding jQuery function.

1. Do the same to add the height() of the div to the myTxt var
2. Do the same with the innerWidth() and the innerHeight()
3. Do the same with the outerWidth() and the outerHeight()
4. Use the jQuery html function to replace the HTML in the div with the value of the myTxt variable.
   * 1. Save and run the page. When you click on the div, the values should be displayed. The values are: width 300, height 200, inner width: 320, inner height: 220, outer width: 322 and outer height: 222.
     2. Now add a horizontal rule to the txt variable (<hr>)
     3. Then add code to get the document height and width ($(document).height(), etc) and the window height and width and display that below the horizontal rule).
     4. Run the program again. When you click the div, something like the following should be displayed:



* + 1. Now add an event listener for the click event on the first button which will increase the width of the div by 20 and the height by 10 whenever it is clicked.

 does the width. To modify the height, chain (add to the same command using a period and then the new command) a similar height command to the given statement.

* + 1. Now add an event listener for the click event on the second button which will decrease the width of the div by 20 and the height by 10 whenever it is clicked.
    2. Run it again, when you click in the div, the dimensions are displayed. When you click the button the div resizes (you can then click the div again to check the new size).

**To submit**

When you have completed the lab exercise, show me your completed work and create a zip file of your YourUserNameC20L12 folder, containing all of files (html, css and js) and any subfolders,and copy the zip file to the Moodle page for the course.