**COMM644  
Assignment 5**

**Practice with Forms and the DOM Event Model (10 points)**

In this part of the assignment you will practice working with form objects and the DOM Event Model:

1. Add a button to a web page. Use the old school HTML attribute method to attach a click event handler to the button. That handler should call a function that displays the message “I have been clicked” out into an alert box.
2. Add a button to a web page and uniquely ID the button. Write a script that finds the button by its ID. Then use the DOM Element method to associate a function with the onclick event of the button. Again, display the message “I have been clicked” out into an alert box.
3. Add a button to a web page and uniquely ID the button. Write a script that attaches an event listener to the button and listens for the click event. When it’s heard, a user-defined function should be called and the message “I have been clicked” should be displayed out into an alert box.
4. Modify the code within step 3 so that the code that displays an alert box is written directly within a callback function as opposed to a user-defined function.
5. Modify the code in step 4 so that you can write your script within the <head> tag of the page rather than underneath the button in the markup.
6. Add a link to your web page. Add the text “Visit Google” to the link, set the ID to “redirect” and set the href attribute to http://www.google.com. Now, write a script that alerts the user that they clicked on the link but prevents the browser from also redirecting to the Google site.
7. Add a text box and a button to a web page. Add an event listener to the button. When the click event is heard, the button should be disabled, and the text that the user input into the text box should be displayed within an alert box.
8. Create a new page called newpage.html. You will now have 2 web pages including practice.html and newpage.html. Add a button to practice.html. When the button is clicked, newpage.html should appear as a popup window at 300 pixels wide by 300 pixels high. As soon as newpage.html loads however, resize the window to 500 pixels wide by 300 pixels high. You will need to handle the load event on newpage.html and use a BOM method on newpage.html to accomplish this task.
9. Add two buttons to a web page. Add the text Start to one button and Stop to the other button. Using the BOM .setInterval() method, figure out a way to move display the text “Learning JavaScript is fun!” into the console window every 2 seconds but only when the Start button is clicked. If the user clicks on the Stop button, the console should stop logging the text immediately.
10. Add a drop down list to a web page. Populate the drop down list with 4 items. Add a button next to the drop down list and set the text to “Select”. When the user clicks the button, the item selected in the drop down list should appear within an alert box.

**The Calculator 2.0 (20 points)**

In this part of the assignment you will practice working with the DOM, DOM Events, Functions, and more to create a better version of the calculator that we built in a previous class. To complete this part, follow the steps outlined below:

1. Notice the markup that I’ve included for you in the web page. There is a text box that will display the value of each button click and 16 different buttons each of which represents a major key on a basic calculator. Begin by adding an event listener to the window object that listens for the load event and when heard, calls a user-defined function called init().
2. Create the user-defined function called init(). This is the function where you will handle all of the event listeners for every button in calculator button.
3. Create an anonymous helper function that gets an element from the DOM and returns that element as an object stored in “$”.
4. Within the init() function, attach an event listener to each of the 16 buttons in the web page. You will listen for the click event and when heard, a function called enter() should be called for each of the buttons except for the equal sign.   
   TIP: Since you are attaching an event listener to an object (each button in the calculator), you will need to pass the value of the selected object into the enter() function. You can do this using the syntax $(“one”).value for instance.
5. For the equal button, call a function called calculate(). You will not pass any arguments into this function.
6. Create a user-defined function called enter() that accepts val as an argument.
7. Within the enter() function get the result text box by its ID and set its value plus equal to the val parameter being passed in.
8. Create a user-defined function called calculate() that doesn’t accept any arguments.
9. Within the calculate() function get the result text box by its ID and set its value equal to the calculation currently stored in the result text box. HINT: Use the built-in eval() function to perform the heavy lifting for you.

Figure out a way to simplify your code so that you don’t have to manually attach an event handler to each and every button. You should be able to programmatically attach a listener to every button on the page and attach a handler that listens for the click without having to write so much code.

HINT: Remember the concept of “event bubbling”. Perhaps it makes more sense to listen for an event on a parent element rather than each and every button….

**The Employee Management System v2.0 (20 points)**

If you can recall, in lab 09 we built an Employee Management application that allowed the user to show all employees, add an employee, and delete an employee. The user had to enter the commands manually within a JavaScript prompt. In this part of the assignment you will rebuild that lab using forms, form objects, and you will script all of the code using DOM scripting techniques. When building the application, keep the following points in mind:

* As was the case with lab 09, you will use an array (multi-dimensional) to persist the employee data for this application.
* Include 5 employees to start with so that the user can see employees in the system as soon as the application loads.
* When the application loads, when a new employee is added, or when an existing employee is deleted, employees should appear neatly structured within a table. A column should be made available for name, title, extension, and a fourth column for the delete button (see example below).
* The application should include 3 text boxes and 1 button across the top of the page that allows the user to add a new name, title, and extension for the employee being added to the system.
* Use JavaScript to check and make sure that the user enter a value within each of the 3 text boxes. Do not use the Required attribute here…use JavaScript! If the user fails to enter a value in the text box and clicks the Add button, display a red error message next to that text box.
* Display the count of employees in the system within the header. Ex: “Showing 3 Employees”
* Next to each employee in the table, a delete button should be visible that when clicked, deletes that employee from the system.
* You do not need to use Web Storage for this assignment.

Although I am flexible on the design, it should resemble the following layout:

