**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 index.html and newpage.html. Add a button to index.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 II (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 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().

3. 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. Since you are attaching an event listener to an object, you can pass in this.value as a parameter to each of the enter() function calls.

4. For the equal button, call a function called calculate(). You will not pass any arguments into this function.

5. Create a user-defined function called enter() that accepts val as an argument.

6. 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.

7. Create a user-defined function called calculate() that doesn’t accept any arguments.

8. 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….

**JavaScript Image Rollovers (20 points)**

In this part of the assignment you will practice working with the DOM, DOM Events, Functions, and more to create an application that features a navigation bar with rollover images. To complete this part, follow the steps outlined below:

1. Notice the 5 images in the web page. These are the 5 images that you will be creating rollovers for. Notice that for each image, an “over” state exists for it in the same directory. Begin by creating a unique ID for each image so that you have image1, image2, image3, image4, and image5.

2. Now, within the script tag of your web page, create a helper function called getImage(). This function will accept one parameter. Within the function, return the result of getting an element by its ID, passing in the function’s argument as a parameter into the getElementById() method.

3. Create a new constructor function called Rollover that accepts two arguments including imageId and newImageURL.

4. Create a new object variable called target and store this within it.

5. Create 3 properties associated with target including newImageURL, image, and oldImageURL.

6. Store the value of the newImageURL parameter within the newImageURL property.

7. Store the return value of the getImage() function, passing in the imageID argument as a parameter within the image property.

8. Store this.image.src (the URL or source of the image you’re working with) within the oldImageURL property.

9. Create an event handler for onmouseover and one for onmouseout for each target image, or:

target.image.onmouseover = function() {

}

target.image.onmouseout = function() {

}

10. In the mouseover, set the src property of the target.image to the newImageURL property.

11. In the mouseout, set the src property of the target.image to the oldImageURL property.

12. Create an event handler to handle the loading of the web page (window).

13. Within the event handler function, create 5 new instances of the Rollover object, passing in the image’s ID and the rollover image path for each image as parameters within the constructor.