**Lab 4**

**This lab assumes you have done all your previous labs and now able to use your previous experience!**

**Build the following simple Javascript app. It should have exactly the same functionality!**

**Task 1)**

<http://www.back2college.com/gpa.htm>

Also, please validate all data that user enters i.e credit hours cannot be greater than 4 or less than 3 (Following UCP Convention).

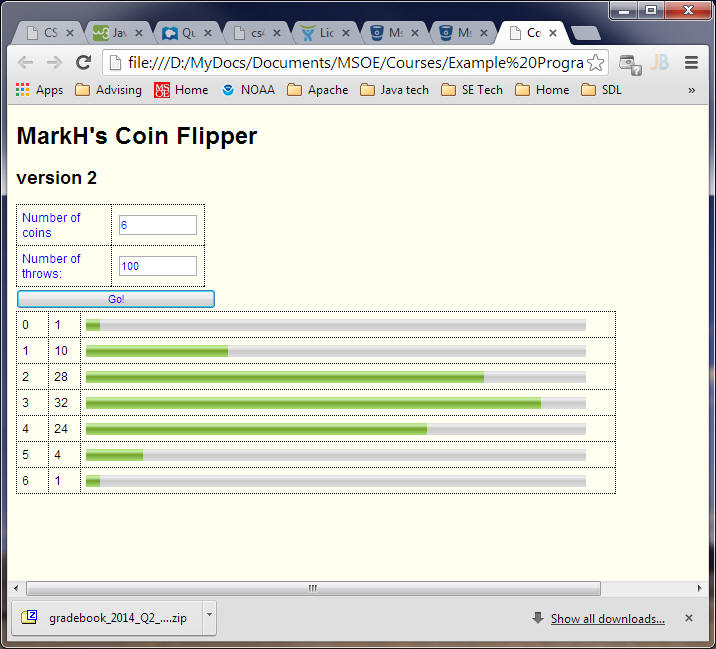
**Please Read: by default have fields for 5 courses only – Add a button that says “Add another Course” which appends a div to your courses section.**

See Next Page for Task 2

**Task 2 – Coin Flipper**

**Input and output**

In this lab task, you will create a CoinFlipper application to use HTML input elements to get user-specified values for both the number of coins and number of throws (repetitions). You'll also change the way the program displays the output by scripting the DOM to dynamically create HTML content, so that it looks something like this:



**What is Coin Flipper?**

**Coin Flipper is a simple application that throws a specific coins and for each coins it assigns a random number between 1 and 100 and displays their result in a histogram. Note that if there are more than 1 throws then add up all the rand numbers generated on each throw and after last throw take mod by 100.**

Note that this example uses tables with various elements nested within the table's cells (cells are dotted to make them visible here, but the outlines are normally not shown). Using tables to assist with HTML "GUI" element layout is a common approach, although there are other techniques available. If you want to, you may use another technique to achieve the same general appearance.

For input, you are **required** to use text boxes, purposely so you will be forced to handle errors on input. Normally, if you can avoid input errors by using a better paradigm (such as a drop-down menu of valid choices), you should take such an approach. For the purpose of this lab, you need to make sure that the user specifies from 1 and 10 coins, and from 1 and 100,000 throws.

You are required to display error messages regarding user input adjacent to the input fields - do not use the alert() function to present errors.

Your application must use an HTML button element to initiate the coin-flipping algorithm (details below). When that completes, your application's printHistogram() function should display output similar to that of the previous lab, except that it must be displayed on the same web page via JavaScript-generated dynamic HTML. There are various approaches to creating the histogram display; one way is to use the <progress> or <meter> elements (new to HTML5) to display the equivalent of the asterisk-based histograms of the previous lab. **Note: printing asterisks is not allowed!** For this lab, the histograms must be graphical in appearance.

**CSS Styling**

Use CSS style rule to achieve the same general appearance as shown above, although you may choose your own colors, fonts, font sizes, table borders. However, you must adhere to the following few requirements: Style the histogram bars to take up about 90% of the overall width of the output portion of the display (the default width for the <meter> and <progress> elements is rather small). Similarly, make the error message area wide enough to present those messages on only 1 or 2 lines. You may put your CSS rules in the .html file or in an external .css file.

**Event Handling**

For this assignment, place the majority of your JavaScript code in the CoinFlipper.js file. In your CoinFlipper.html file, put only a minimal amount of code in the <head> section, like so:

<!-- CoinFlipper functions in external file -->

<script src="CoinFlipper.js"></script>

<-- embedded JavaScript -->

<script>

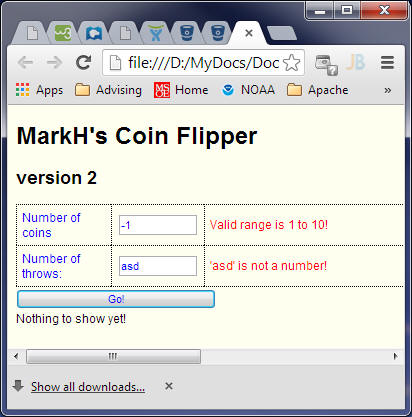
window.onload = initCoinFlipper; // event handler - executes CoinFlipper initialization code when page finishes loading

</script>

Within the initCoinFlipper() function, you should set up an event handler for the button so that it executes the rest of the code when the "Go" button is pressed.

**Error Handling**

When the "Go" button is pressed, you'll need to retrieve the values in the text fields and check them for valid values. You will need to show (via DOM scripting) error messages alongside the invalid input (and hide messages for valid input). This is the typical approach used in checkout forms in Amazon and other e-commerce sites. A reason-specific error message should be generated that **distinguishes** between a) value outside of allowed range, b) not a valid number (e.g 8.7), c) nothing entered, d) non-numeric input (e.g. "asd").



For validation of user input, you may find the following code useful as a **starting** point, but you will have to add additional code to complete the error checking:

/\* Checks to see if the specified argument contains an integer.

Returns false if simple checks indicate an invalid value or a value that is not a number.

\*/

function isInteger(value) {

var bool = isNaN(value); // true if not a number

bool = bool || (value.indexOf('.') != -1); // true if not a number like 3.2

bool = bool || (value.indexOf(",") != -1); // true if not a number like 3,2 (european decimal)

return !bool;

}

If your application detects invalid input, it should display error messages, and should **not** display any histograms.