Requirements

Write a simple tip calculator that you might use at a restaurant. The emphasis for this this version of the app is functionality and code correctness. However, your UI for this assignment should have a plain, but purposeful layout.

Your UI consists of:

a. Total check amount – textbox, default to empty

b. Sales Tax Percent – textbox, default to 7%

c. Tip Amount Percent – textbox, default to 18%

d. Sales tax amount – read-only text *(your code must calculate this amount using the total check amount and sales tax percent)*

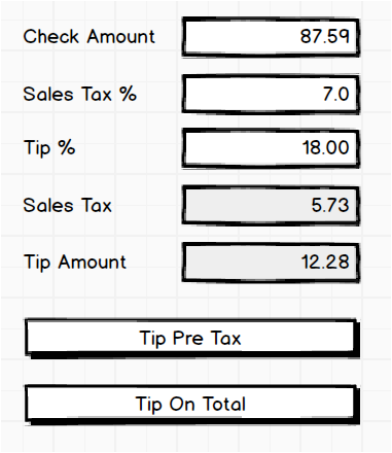
e. Tip amount – read-only text (your code must calculate this amount)

f. Two calculate buttons:

   i. One button to calculate the tip based on the total amount of the check

   ii. One button to calculate the tip based on the pre-sales tax amount of the check

g. A low-fidelity mockup for the UI is shown below:



1. Use absolute pixel placement to position the UI objects.
2. The user must be able to enter new values for both Sales Tax Percent, and Tip Amount Percent.
3. When either button is pressed, calculate and display the sales tax amount and tip amount.
4. Before calculations are performed, validate the input textbox values to ensure they are positive numbers with at most two decimal positions. If input is invalid, display an error message to the user, set the focus to the textbox in error, and skip the calculations.
5. To determine if a textbox contains a valid number or not, write your own *tryParse* function.
   a. This function should be passed two arguments:
      i. A string that will be checked to see if it can be converted to a positive or negative number.
      ii. An integer to indicate the maximum number of decimal positions allowed
   b. Your *tryParse* function should return either *true* or *false* indicating if the string contained a valid number or not.

   c. The logic in this function should loop through the string and check each character. The only valid characters are: the digit characters 0 thru 9; one decimal point character; and one hyphen (for negative numbers). If any other characters are found, the string should be considered invalid. If present, the hyphen character must be present in a sensible position. E.g. "123-456" should not be considered a valid negative number.

6. Develop your UI using an HTML page that uses CSS for positioning UI objects. Write all your processing code using JavaScript.
7. Your source code should contain two files – an HTML file containing the HTML and CSS UI definition and a JavaScript file containing all the processing logic.
8. You may not use any third-party code or regular expressions when completing this assignment.
9. Tip: The entire solution can be created in about 350 lines of code, including all UI, all logic, comments, and whitespace. If your lines of code greatly exceed this number, you are probably including more code than is necessary.

Submission

Submission instructions and due date will be announced at a later date.

Scoring

Please help us expedite the grading process by including your own comments in the table below if you didn't give yourself the Max Points or if you have any questions about the way you coded something. Failure to do so may result in a score of zero or at least require you to resubmit your work.

Max Points	Suggested Score	Actual Score	Requirement
5			Correct file submission, coding style and readability, etc.
10			General UI implementation
5			Input validation – messages, focus, type conversion
10			Input validation – tryParse function
10			Correct calculations
40	**0**	**0**	**Total**