CST Lab 6.1 - Upgrade the Email List application

In this exercise, you’ll upgrade the code in the Email List application so the span elements that display the error messages don’t require id attributes.



1. Open the HTML and JavaScript files in this folder:

**\email\_list\**

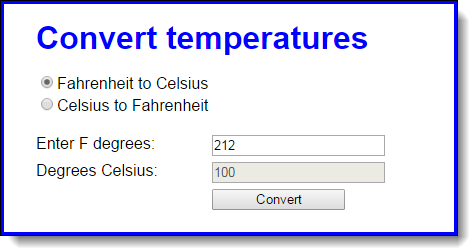
Then, run the application to see that the validation works as shown above.

In the HTML file, delete the id attributes for the span elements.

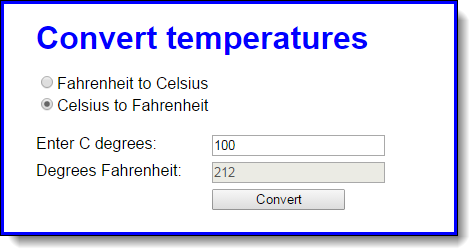
In the JavaScript file, use the properties of the Node interface to display the error messages without using the id attributes.

# CST Lab 6.2 - Develop the Temperature Converter

In this exercise, you’ll use radio buttons to determine whether the conversion is from Fahrenheit to Celsius or vice versa. You’ll also modify the DOM so the labels change when a radio button is clicked. When the application starts, it will look like this:



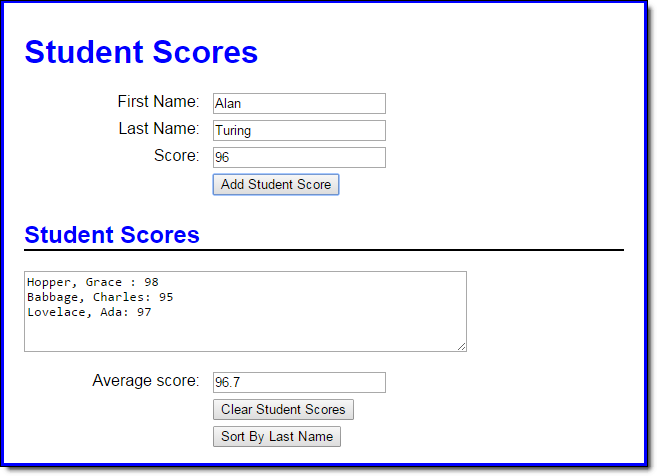
When the user clicks on the second radio button, the labels will change so the interface will look like this:



1. Download the HTML, CSS and JavaScript files from the convertTemp folder on Moodle
2. Note that the JavaScript file has some starting JavaScript code, including the $ function, a clearTextBoxes function, and an onload event handler that attaches three event handlers named convertTemp, toCelsius, and toFahrenheit.
3. Code the toFahrenheit function that is executed when the user clicks on the second radio button. It should change the text in the labels for the text boxes so they read as in the second interface above. It should also call the clearTextBoxes function to clear the text boxes.
4. Code the toCelsius function that is executed when the user clicks on the first radio button. It should change the text in the labels for the text boxes so they read as in the first interface above. It should also call the clearTextBoxes function to clear the text boxes.
5. Code the convertTemp function without any data validation. It should calculate the temperature based on which button is checked. To convert Fahrenheit to Celsius, first subtract 32 from the Fahrenheit temperature, and then multiply that result by 5/9. To convert Celsius to Fahrenheit, first multiply Celsius by 9/5, and then add 32. The result in either case should be rounded to zero decimal places.
6. Add data validation to the convertTemp function. The only test is whether the entry is a valid number. If it isn’t, this message should be displayed in a dialog box: “You must enter a valid number for degrees.”
7. Add any finishing touches to the application like moving the focus to the first text box whenever that’s appropriate.

# 6.3 - Develop the Student Scores application

In this exercise, you’ll develop an application that tracks student’s scores, computes the average of the entered scores, and sorts the entered students by last name. The interface looks like this:



1. Open the HTML, CSS and JavaScript files in the studentScores folder on Moodle
2. In the JavaScript file, note that six functions are supplied. The $ function. The start of a displayScores function. The start of an addScore function that ends by clearing the add form and setting the focus on its first field. The start of a clearScores function that ends by clearing the display area and setting the focus on the first name field. The start of a sortScores function. And an onload event handler that attaches the addScore, clearScores, and sortScores functions to the click events of the appropriate buttons and sets the focus on the first name field.
3. To start, code two global arrays, one to hold the score values and the other to hold the strings that display the students’ names and scores.
4. In the displayScores function, add the code that calculates the average score of all the scores in the first array, and stores it in the text box below the text area.

Then, add the code that gets the students’ names and scores in the second array and displays it in the text area.

Note: Arrays in javascript have a built-in function to ‘join’ all the elements into a string. The method takes a parameter which determines how the elements are to be separated. The default separator is a comma (,).

e.g. var arrayString = scoresString.join("\n");

This will join all the elements of the array into a string with each element separated by a new-line.

1. In the addScore function, save the score in the first array and to save the name and score string (as shown in the text box) in the second array. Then, call the displayScores function to redisplay the updated data.
2. In the clearScores function, add code that clears both global arrays.
3. In the sortScores function, add code that sorts the students by last name and then re-displays the score information.

Note: Use the built-in javascript array method sort().

# Print out the DOM tree

Write a program to parse through all the nodes in a document starting from a given node, and print out (document.write) the node name and value of all descendants

Hint:

* + Write a recursive function
  + The function will take one node as input
  + The function will loop through each child and print each child’s name (nodeName) and value (nodeValue)
  + The function will check if each child has children, and if so, it will recursively call the function for that child.

**Partial Program Output:**

