

ASSIGNMENT 4

Version 21/WI

MAPPINGS:

The following course objectives and/or outcomes are measured in this assignment:

COURSE OBJECTIVES

- 4A: Develop test plans for the applications you develop.
- 4B: Use browser developer tools to debug applications by setting breakpoints, viewing the current data values, and stepping through the execution of statements.
- 4C: Trace the execution of an application with `console.log()` statements.
- 4D: Develop DOM scripting applications that work with forms and controls, including applications that add nodes to the DOM.

COURSE OUTCOMES

- 1. Use JavaScript to create interactive web applications.
- 2. Use JavaScript to interact with application programming interfaces (APIs).
- 3. Use browser developer tools to troubleshoot web page/application issues.
- 4. Write clean, consistent code.

Continues . . .



NOTES:

Implementing the functionality of the `performConversion()` function in Assignment 4 will require you to employ problem solving skills more heavily this week. Think back to the tools you used in INFO 1003. You may have to develop a flowchart or write out pseudocode to work through the implementation. While working on this assignment, **think simple**. Break the problem down into simpler, smaller problems and solve the smaller problems, building towards a larger solution. For example, solve for converting the functionality of *Fahrenheit to Celsius* first; don't try to solve everything at once. Once you have solved for one part of the problem, move to implementing functionality for *Celsius to Fahrenheit*, and so forth.

The assignment utilizes concepts covered in Chapters 1 – Chapter 6 of the course textbook. You should not need to employ tools or concepts from beyond Chapter 6 to solve this problem.

If you reach an impasse, think: “*What do I already know?*”

A sample solution for Assignment 4 may be viewed here:

<https://grosas.faculty.mccinfo.net/courses/info2124/v21fa/assignment04/>

You will be submitting two files this week. Both files should be attached to a single assignment submission. A link has been provided to a video demonstrating how to accomplish this task if you are unfamiliar with this process. The link is in the *Submission* instructions at the end of the assignment.

Continues...



GRADING:

Task	Points	Criteria
Part 1		
1.1 Setup		
3	5	Each line of the standard opening comment is worth 1 point. Subtract 1 point per missing item.
4	5	Pass/Fail
5	5	Pass/Fail
1.2 Define <code>toggleDisplay()</code>		
1	2	Pass/Fail
2	2	Pass/Fail
3	2	Pass/Fail
4	2	Pass/Fail
1.3 Implement <code>toggleDisplay()</code>		
	4	2 points per radio button; within the context of the 2 points per button adjust for missing features or errors.
1.4 Define and Implement <code>performConversion()</code>		
1	5	5 points for task, deduct points based on errors (instructor discretion).
2	5	5 points for task, deduct points based on errors (instructor discretion); however, spelling errors in the error in the error message, or the implementation of a error text that differs from the exemplar should <i>not</i> be penalized
3.a	2	Pass/Fail
3.b	4	Pass/Fail: 2 for each radio button accounted for.
3.c	15	The code should determine which radio button is selected, then invoke the appropriate conversion based on the selected radio button to obtain the correct result. The code should also include additional helper functions to calculate the conversion of feet/meters and meters/feet.
Part 2		
2.b	6	3 points for each test. You may deduct from the 3 points if information is missing, incomplete, or incorrect.
3.c	4	1 point for each complete line of documentation in the test plan (notes does not need to contain anything).
3.e	4	2 points for each conversion that displays two digits after decimal point (pass/fail for each conversion).
Total	72	

Penalties

Deduct 50% from entire assignment for the use of the **var** keyword in variable declarations.

Deduct 30% from the entire assignment if the solution does not load and/or if errors appear in the console that are not generated by explicit methods of the console object (in other words, errors that have not been troubleshot and resolved by the student prior to submission).

Deduct 60% from the entire assignment if JavaScript is inline/embedded instead of external.

Deduct a maximum of 7.5% for code that does not comply with the course *Style Guide* and/or which is messy/unorganized, uncommented, or missing semicolons.

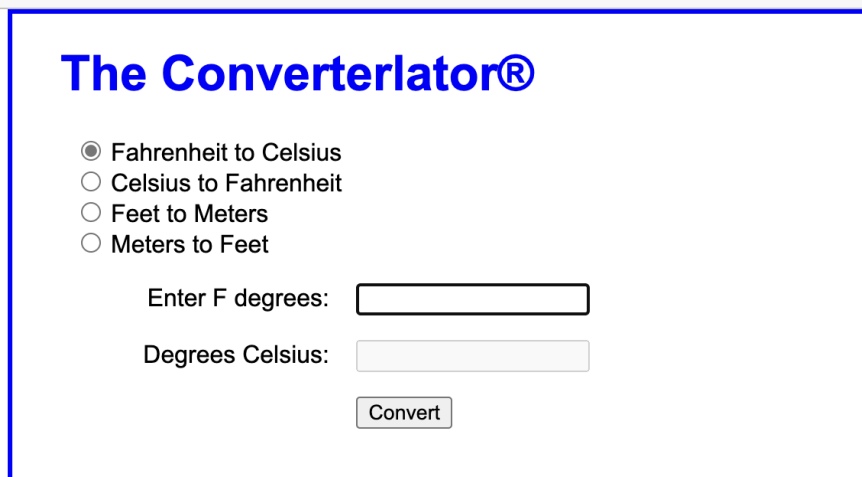
Continues...



TASK:

TEST THE APPLICATION

1. Review the course *JavaScript Style Guide* before starting this assignment. Part of the assignment will be graded on your adherence to the *Style Guide*.
2. Download **assignment04_starter.zip** from the *Module 4: Assignment* drop box in Canvas. The file is located beneath the heading *Assignment Resources*.
3. Extract **assignment04_starter.zip**. The file contains three files and one subfolder:
 - a. a single HTML document named *index.html*.
 - b. a single JavaScript file named *convert_temp.js*.
 - c. a single Stylesheet named *convert_temp.css*
4. Double-click **index.html** to launch it in your browser; examine the application's behavior.
 - a. The page will open, and the content shown in Figure 1 should appear.
 - b. Test the application by inputting values and clicking **Convert**. Note that nothing happens.
 - c. Open the browser's developer tools and activate the *Console* tab. Note the absence of errors.



The Converterlator®

☒ Fahrenheit to Celsius
☐ Celsius to Fahrenheit
☐ Feet to Meters
☐ Meters to Feet

Enter F degrees:

Degrees Celsius:

Figure 1: Starter version of index.html

Continues...

MODIFY THE APPLICATION

Part 1

1.1 SETUP

1. `convert_tempt.js` has already been linked to `index.html` for you.
2. Open `convert_tempt.js`.
3. Add the **Standard Opening Comment** to the top of the script (**5 points**).
4. Add "use strict" to the very top of the JavaScript file (above the *//DO NOT MODIFY THE CODE BELOW* comment).
(**5 points**)
5. As you develop your solution, use the browser tools to help you troubleshoot. Your solution must include at least one `console.log()` statement. (**5 points**)

1.2 DEFINE `TOGGLEDISPLAY()`

Find the block of code shown in Figure 2 (below) and write JavaScript to accomplish the following:

1. Change the text of `label_1` to the value stored in the variable `label1Text`. (**2 points**)
2. Change the text of `label_2` to the value stored in the variable `label2Text`. (**2 points**)
3. Clear the contents of `value_computed`. (**2 points**)
4. Then, select the degrees textbox. (**2 points**)

```
13  const toggleDisplay = (label1Text, label2Text) => {  
14  
15  }
```

Figure 2

Continues . . .



5. Once the changes have been made, you should see that selecting “Celsius to Fahrenheit” or selecting “Fahrenheit to Celsius” will change the text highlighted in Figure 3. . . .

The Converterlator®

Temperature

☒ Fahrenheit to Celsius
☐ Celsius to Fahrenheit

Distance

☐ Feet to Meters
☐ Meters to Feet

Enter **F** degrees:

Degrees **Celsius**:

The Converterlator®

Temperature

☐ Fahrenheit to Celsius
☒ Celsius to Fahrenheit

Distance

☐ Feet to Meters
☐ Meters to Feet

Enter **C** degrees:

Degrees **Fahrenheit**:

Figure 3

6. . . . and selecting “Feet to Meters” or selecting “Meters to Feet” will change the text highlighted in Figure 4

The Converterlator®

Temperature

☐ Fahrenheit to Celsius
☐ Celsius to Fahrenheit

Distance

☒ Feet to Meters
☐ Meters to Feet

Enter **feet**:

Meters:

The Converterlator®

Temperature

☐ Fahrenheit to Celsius
☐ Celsius to Fahrenheit

Distance

☐ Feet to Meters
☒ Meters to Feet

Enter **meters**:

Feet:

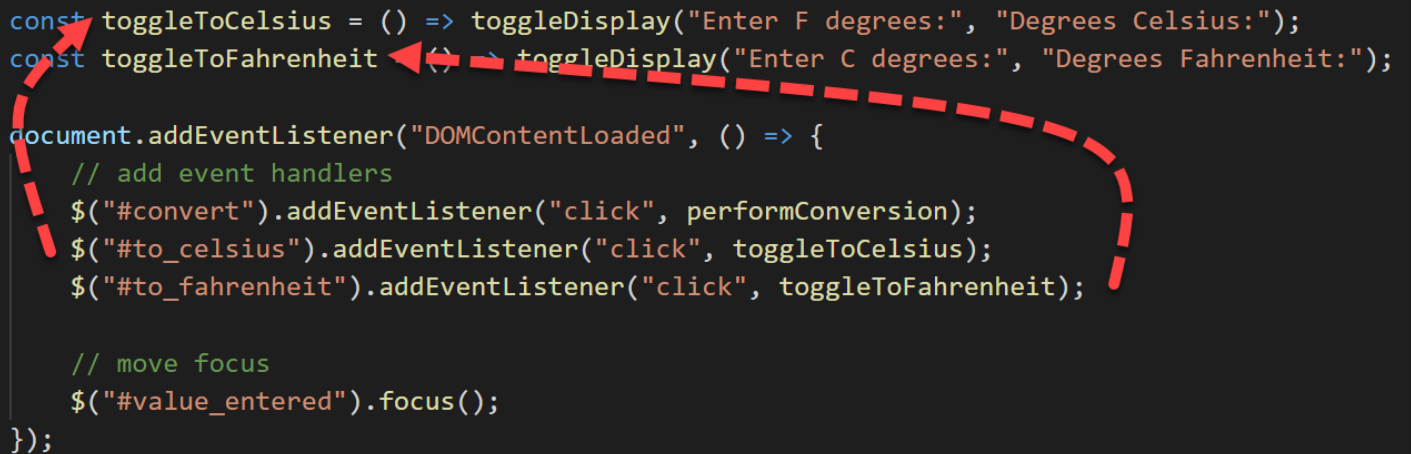
Figure 4

Continues . . .



1.3 IMPLEMENT TOGGLEDISPLAY ()

The code that controls the label change described in Task 1.2 is shown below, in Figure 5. As a courtesy, arrows have been added to show event handlers and their respective definition. Implement functionality to handle the click events for the “Feet to Meters” and “Meters to Feet” radio buttons, toggling the labels to display the appropriate text when each radio button is clicked. (4 points)



```
const toggleToCelsius = () => toggleDisplay("Enter F degrees:", "Degrees Celsius:");
const toggleToFahrenheit = () => toggleDisplay("Enter C degrees:", "Degrees Fahrenheit:");

document.addEventListener("DOMContentLoaded", () => {
  // add event handlers
  $("#convert").addEventListener("click", performConversion);
  $("#to_celsius").addEventListener("click", toggleToCelsius);
  $("#to_fahrenheit").addEventListener("click", toggleToFahrenheit);

  // move focus
  $("#value_entered").focus();
});
```

Figure 5

Continues . . .



1.4 DEFINE AND IMPLEMENT `performConversion()`

Find the code stub for the `performConversion` event handler (Figure 6).

```
const performConversion = () => {  
  
};
```

Figure 6

In the body of the `performConversion` event handler, write JavaScript to accomplish the following:

1. Obtain the value input by the user. Make sure you preserve numeric precision. **(5 points)**
2. If the value input by the user is **not** a number, display the appropriate error message in the `<div>` with the id value "message." Then, clear the input and select the `#value_entered` textbox. An example of this behavior and the error message you should display is shown in Figure 7. **(5 points)**

The Converterlator®

You must enter a valid number for degrees.

Temperature

☒ Fahrenheit to Celsius
☐ Celsius to Fahrenheit

Distance

☐ Feet to Meters
☐ Meters to Feet

Enter F degrees:

Degrees Celsius:

Figure 7

Continues . . .

3. If the value input by the user **is** a number:
- Clear any previous error message. **(2 points)**
 - Determine which radio button is selected (*checked*): **(4 points)**
 - Compute and display appropriate value based on which radio button is checked by invoking the appropriate function. Then select the `#value_entered` textbox. An example of this behavior is shown in Figure 8. **(15 points)**
 - The functions for converting Celsius to Fahrenheit and Fahrenheit to Celsius have already been defined for you in the *helper functions* section of the code. You simply need to invoke them in the appropriate place within the body of *performConversion*. You will also need to write helper functions to convert feet to meters and meters to feet and invoke those functions at the appropriate place. You can use the following formulas when writing those functions:
 - $\text{meters} = \text{feet} / 3.2808$
 - $\text{feet} = \text{meters} \times 3.2808$
 - The output of **all** conversion functions should display 2 digits after the decimal point **(4 points)**.

The Converterlator®

Temperature

☐ Fahrenheit to Celsius
☐ Celsius to Fahrenheit

Distance

☐ Feet to Meters
☒ Meters to Feet

Enter meters:

Feet:

Figure 8

NOTE: You can determine which radio button is checked by reading an individual radio button's `checked` property. The `checked` property will be a Boolean value.

Continues...



Part 2

1. Download **INFO 2124 Software Test Template.docx** from the *Module 4: Assignment* drop box in Canvas. The file is located beneath the heading *Assignment Resources*.
2. Open **INFO 2124 Software Test Template.docx** and review its contents.
 - a. The test plan has two tests already defined in it. Browse to <https://grosas.faculty.mccinfo.net/courses/info2124/v21fa/assignment04/> and perform both tests. Both tests should pass. **Do not** fill out the test results for Steps 1 and 2 of Part 2.
3. Use **INFO 2124 Software Test Template.docx** to write a simple test plan by adding to the existing document.
 - a. Fill out the top table with the project name, the test writer (you), the tester (you), and the date of the test.
 - b. Review the assignment requirements. Write at least **two (2) new** tests: **(6 points)**
 - i. Do not rewrite or modify the existing tests.
 - ii. Each new test should include test data and the expected result.
 - iii. Review the assignment requirements in Part 1 to help you write the two new tests.
 - c. When complete, you should have four (4) tests. Run all four tests against **your** solution for Part 1 of this assignment (the two you were given, and the two you wrote). Record the results and indicate if each test passed or failed. **(4 points)**
 - i. The testing process can be a little tricky but if you write a test where you expect to receive an error and if you run the test and receive an error as expected, your test has passed. Pass/Fail is in terms of whether your test met expected outcomes (pass) or did not meet expected outcomes (fail).
 - d. The following fields should be filled out for each test in your test plan: *Step, Test Steps, Test Data, Expected Result, Actual Result, Status (Pass/Fail)*. You do not have to put anything into the *Notes* field unless you want to.
 - e. You may write more than two new tests; however, there is no additional credit for going above and beyond.

Continues...



SUBMISSION

When complete, create a single **ZIP** file containing your solution for Part 1 of this assignment. The ZIP file should contain all files included in the original starter code or added as part of this assignment.

Attach and upload the ZIP file for Part 1 of this assignment and the Word document with your completed test plan from Part 2 of this assignment to Assignment 4 and submit. You will be attaching both files to a single assignment submission. If you have never done this before, please watch this short video: <https://use.vg/Djbd22Iw6POg> - if you have questions, contact your instructor for assistance.

NOTE: Canvas is configured to only accept ZIP files, DOC files, and DOCX files; it will not accept ZIPx, 7ZIP, pZip, RAR, etc. or files that are not in Microsoft Word format.

End Assignment.

