ASSIGNMENT 9

Version 21/FA

MAPPINGS:

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

COURSE OBJECTIVES

- 9F: Use default parameters and rest parameters in your functions.
- 9G: Use closures, immediately invoked function expressions (IIFE), the module pattern, and ECMAScript modules in your applications.

COURSE OUTCOMES

- 1. Use JavaScript to create interactive web applications.
- 2. Write clean, consistent code.

OVERVIEW:

You have recently started working for BigRedWeb.com – a local web consulting firm. BigRedWeb is refactoring legacy code for a client.

You have been tasked with refactoring a clock and weather application for one of BigRedWeb's biggest clients. You will be separating code into libraries, implementing closures, and encapsulating code in an *Immediately Invoked Function Expression* (IIFE) as part of this task.

The client does not want any third-party libraries used in this task, so all work must be in **vanilla JavaScript**, no jQuery. To that end, you must be able to differentiate between jQuery-specific syntax and vanilla JavaScript syntax. If any jQuery-specific syntax is in the original code, you must rewrite the syntax to convert it to the vanilla JavaScript equivalent syntax.

Continues . . .

GRADING:

Task	Points	Criteria
Part 1		
1.1 Setup		
There are no points in this section of the assignment.		
1.2 Test the Application		
There are no points in this section of the assignment.		
1.3 Cleaning Up jQuery		
1	2	Pass/Fail
2	2	Pass/Fail
1.4 Move the Weather Class		
1	2	Pass/Fail
2	2	Pass/Fail
3	2	Pass/Fail
4	8	If the student leaves code from the Weather class in script.js, deduct 4 points. If the student does not name the JavaScript file lib.js, or if the file is in the wrong location, deduct a maximum of 4 points. If a combination of the errors is present, instructor's discretion up to a maximum of 8 points. If the task is not implemented, deduct 8 points.
5	3	Pass/Fail
1.5 IIFE Time		
2	2	Pass/Fail
3	2	Pass/Fail
4	8	Instructor's discretion – deduct points for any code not residing within IIFE as per the task description.
1.6 Closure Time		
2	2	Pass/Fail
3	2	Pass/Fail
4	3	Pass/Fail
5	8	Instructor's discretion – deduct points for any variable/constant that is not private. Deduct points for any method other than startClock() that is not private (startClock() should be the only public method).
6	2	Pass/Fail
Total	**50**	

Penalties
Deduct 50% from entire assignment for the use of the `var` keyword in variable declarations.
Deduct 30% from the entire assignment is 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 20% for code that does not comply with the course *Style Guide* and/or which is messy/unorganized, uncommented, or missing semicolons.

Continues…

TASK:

MODIFY THE APPLICATION

1.1 SETUP

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 **assignment09_starter.zip** from the *Module 9: Assignment* drop box in Canvas. The file is located beneath the heading *Assignment Resources*.
3. Extract **assignment09_starter.zip.** The file contains two files and two subfolders:
    a. a single HTML document named *index.html*.
    b. a subfolder named *css*, which contains a single file named *style.css*
    c. a subfolder named *js*, which contains two JavaScript files:
        i. *clock.js*
        ii. *script.js*
    d. a subfolder named *img*, which contains a single file named *ajax-loader.gif*

1.2 TEST THE APPLICATION

1. Open **index.html** in your browser. If prompted to allow the use of your location, click **Allow**. This is a safe script – the prompt is because the script can use your location to pull weather information.

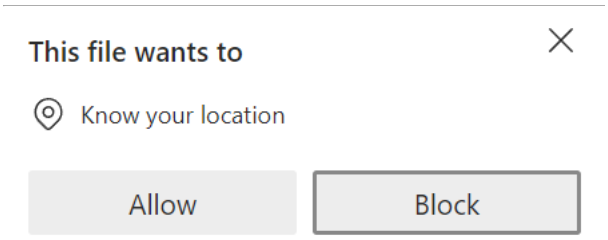

Figure 1

Continues . . .

2. Note the current time displayed on the left-side of the screen. Note the options to pull weather data on the left-hand side of the screen.

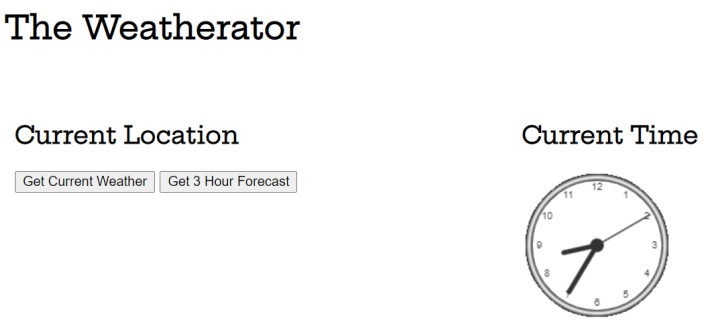

Figure 2

3. Click **Get Current Weather**. After a moment, the current weather should appear (Figure 3) – note the location may not be 100% accurate.

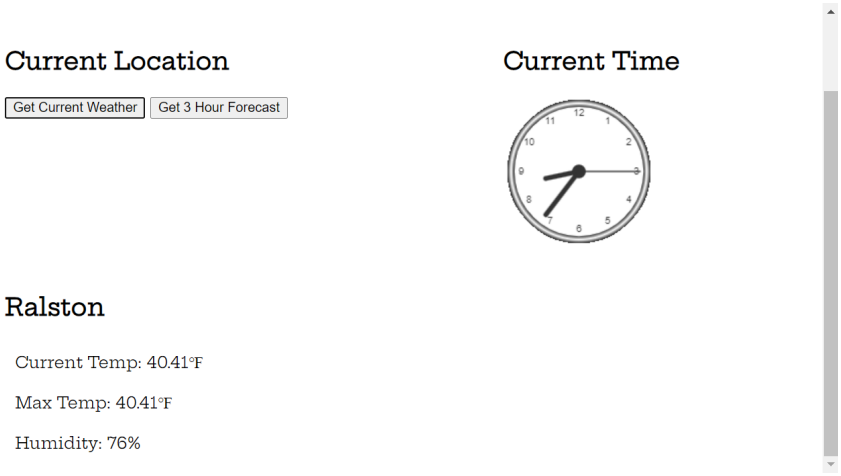

Figure 3

Continues . . .

4. Click **Get 3 Hour Forecast**. After a moment, the 3-hour forecast should appear beneath the clock (Figure 4). Once again, the location may not be 100% accurate.

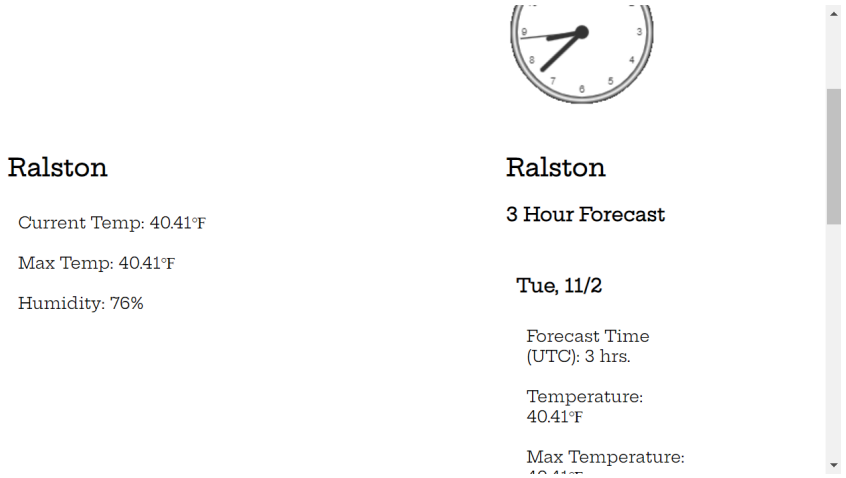

Figure 4

5. Close your browser.

1.3 Cleaning Up jQuery

1. Open **index.html** and remove the link to the jQuery library **(2 points)**.
2. Review **js/script.js** and **js/clock.js** – rewrite any jQuery-specific code so that only vanilla JavaScript is left **(2 points).**
3. Test: the application should function the same as it did when working through **Step 1.2**.

1.4 Move The Weather Class

1. In the **js** folder, create a new JavaScript file named **lib.js (2 points)**
2. Add the **Standard Opening Comment** to the top of the script **(2 points**).
3. Add "use strict" beneath the **Standard Opening Comment**. **(2 points)**.
4. **Move** the `Weather` class from **script.js** to **js/lib.js (8 points)**
5. Open **index.html** and link **lib.js** and to the **index.html** in the appropriate location **(3 points)**.
6. Test: the application should function the same as it did when working through **Step 1.2**.

Continues . . .

1.5 IIFE Time

1. Open **js/script.js**
2. Add the **Standard Opening Comment** to the top of the script **(2 points**).
3. Add "use strict" beneath the **Standard Opening Comment**. **(2 points)**.
4. Move all the code within the event handler that listens for all document content to be loaded into an *Immediately Invoked Function Expression* (IIFE). The only code that should reside outside of the *IIFE* should be the definition for the event handler that waits for all page content to load. **(8 points).**
5. Test: the application should function the same as it did when working through **Step 1.2**.

1.5 Closure Time[1]

1. Open **js/clock.js**
2. Add the **Standard Opening Comment** to the top of the script **(2 points**).
3. Add "use strict" beneath the **Standard Opening Comment**. **(2 points)**.
4. Define a closure named **`Clock`**: (3 **points)**
    a. All existing variables, constants, and functions/methods should be private *except* for `startClock()`, which will be the only public method. **(8 points)**.
5. Open **js/script.js**. Find the line shown in Figure 5 and modify it so that it reflects the code shown in Figure 6 (**2 points).**

```
startClock();
```

Figure 5

```
const c = Clock();
c.startClock();
```

Figure 6

6. Test: the application should function the same as it did when working through **Step 1.2**.

Continues . . .

[1] Just in case you missed 1999: https://www.youtube.com/watch?v=xGytDsqkQY8

1.6 CLEANUP

1. Test your application. If necessary, use the browser's developer tools to troubleshoot any errors you may have.
2. Comment and clean up your code:
    a. Make sure to document what functions and blocks of code do.
    b. Ensure consistent alignment, spacing, and carriage returns.
    c. Make sure to remove non-used code; comments should describe your code only.

SUBMISSION

When complete, create a single ZIP file containing your solution for 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 to Assignment 9 and submit.

NOTE: Canvas is configured to only accept ZIP files, DOC files, and DOCX files; it will not accept ZIPx, 7ZIP, pZip, RAR, etc.

End Assignment.