Javascript-Arrays with forEach and map

Objective

In this exercise, you will learn how to use the `forEach()` and `map()` array methods in JavaScript. These methods are used for iterating over arrays and performing operations on their elements.

Understanding the Code

You are provided with a blank `index.js` file. Your task is to fill in the file by completing the following steps.

Here's what you need to do:

1. **Declare an Array**

Declare a variable `fruits` and initialize it with an array containing the elements: "apple", "banana", and "cherry".

2. **Using `forEach()` to Log Each Element**

You will use the `forEach()` method to log each fruit in the `fruits` array.

You will write the following code to iterate through the `fruits` array and log each fruit name to the console as console.log(fruit);

After running this code, you should see each fruit printed in the console.

3. **Using `map()` to Create a New Array with Modified Elements**

Now, use the `map()` method to create a new array named `uppercasedFruits` with the fruit names in uppercase.

After running this code, log the `uppercasedFruits` array using `console.log()`.

4. **Using `forEach()` to Sum the Lengths of Each Fruit's Name**

Next, use the `forEach()` method again to calculate the total length of all the fruit names.

- **Task**: Declare a variable with name `totalLength` using `var` keyword and for each fruit in the `fruits` array, add the length of its name to a `totalLength` variable.

After completing the iteration, log the total length of all fruit names using `console.log(totalLength);`.

5. **Using `map()` to Create a New Array with Transformed Fruit Information**

Finally, use the `map()` method to create a new array that contains objects with each fruit's name and its length named as `fruitInfo`.

- **map()**: This time, you will return an object with `name` and `length` properties for each fruit.

After running this code, log the `fruitInfo` array using `console.log(fruitInfo);`.

Mandatory Assessment Guidelines:

1. **All actions like build, compile, running application, running test cases will be through Command Terminal.**
2. **To open the command terminal the test takers, need to go to Application menu (Three horizontal lines at left top) -> Terminal ->New Terminal.**
3. **This editor Auto Saves the code.**
4. **If you want to exit(logout) and continue the coding later anytime (using Save & Exit option on Assessment Landing Page) then you need to use CTRL+Shift+B-command compulsorily on code IDE. This will push or save the updated contents in the internal git/repository. Else the code will not be available in the next login.**
5. **These are time bound assessments the timer would stop if you logout and while logging in back using the same credentials the timer would resume from the same time it was stopped from the previous logout.**
6. **This is a web-based application, to run the application on a browser, use the internal browser in the workspace. Click on the second last option on the left panel of IDE, you can find Browser Preview, where you can launch the application.**

   Note: The application will not run in the local browser

7. **You can follow series of command to setup Angular environment once you are in your project-name folder:**
    a. **npm install --no-bin-links --unsafe-perm -> Will install all dependencies -> takes 10 to 15 min.**
    b. **node src/index.js -> To compile and run the index.js file.**
    c. **node src/test/custom-grader.js -> to run all test cases. It is mandatory to run this command before submission of workspace -> takes 5 to 6 min.**
8. **Once you are done with development and ready with submission, you may navigate to the previous tab and submit the workspace. It is mandatory to click on "Submit Assessment" after you are done with code.**
9. **You need to use CTRL+Shift+B - command compulsorily on code IDE, before final submission as well. This will push or save the updated contents in the internal git/repository, and will be used to evaluate the code quality.**