JavaScript Functions & Default Arguments

Objective

In this exercise, you will learn how to define functions with arguments and use default parameters in JavaScript. Functions allow you to reuse code and make your programs more modular. Default parameters allow you to set a default value for a parameter if it is not provided.

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. **Define a Function with Arguments**

Define a function `greet` that accepts two parameters: `name` and `age`.

- **name**: The `name` parameter will represent the person's name.
- **age**: The `age` parameter will represent the person's age. Set a default value of `25` for the `age` parameter, which will be used if no age is provided.

The function should return a greeting message that includes the person's name and age as `Hello, my name is {name} and I am {age} years old.`

2. **Calling the Function with Arguments**

Call the `greet` function with both `name` and `age` arguments. For example, call the function like this:
- `console.log(greet('John', 30));`

This should print: `Hello, my name is John and I am 30 years old.`

3. **Calling the Function with One Argument**

Call the `greet` function again, but this time only pass the `name` argument. The default value for `age` should be used (which is `25`). For example:
- `console.log(greet('Jane'));`

This should print: `Hello, my name is Jane and I am 25 years old.`

4. **Define a Function to Calculate the Sum of Two Numbers**

Define a new function `sum` that accepts two parameters, `a` and `b`, and returns their sum.

This function should return the sum of `a` and `b`.

5. **Calling the Sum Function with Arguments**

Call the `sum` function with two numeric arguments, for example, `5` and `7`.

- Call the function like this:
- `console.log(sum(5, 7));`

This should print: `12`.

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 -> 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.