SOFE2800 – Prelab 2

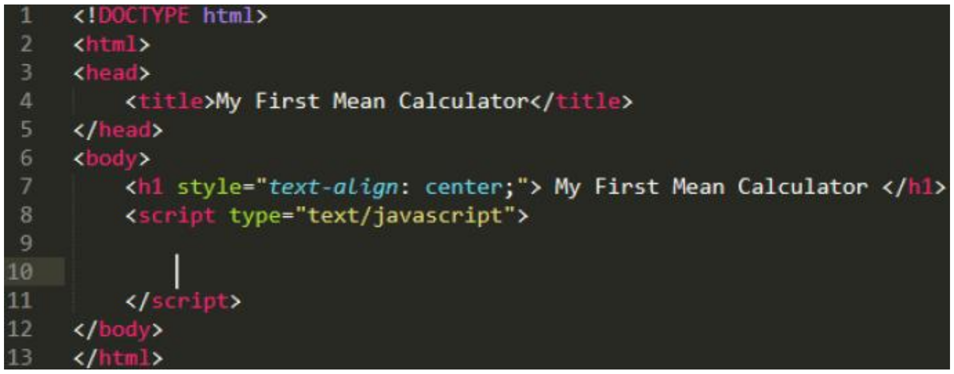
JavaScript Debugging

Before starting this lab, it's helpful to be familiar with **debugging**—a method for identifying and fixing errors (bugs) in your code. Modern browsers include built-in debugging tools. In **Google Chrome**, you can press **F12** and select Console to debug JavaScript. Two common debugging methods are:

1. **console.log()** – Prints JavaScript values in the console so you can verify that variables and code behave as expected.
2. **Breakpoints** – Pause code execution at a specific line to inspect variables and program flow. You can resume execution with the play button.

Debugging Activity

1. Open your text editor, create a new file, and save it as **debug.html** in your workspace.
2. Set up a basic HTML template. In the <body>, create a centered header: **"My First Mean Calculator"**.
3. Add a `<script>` tag with **type="text/javascript"** for your JavaScript code.

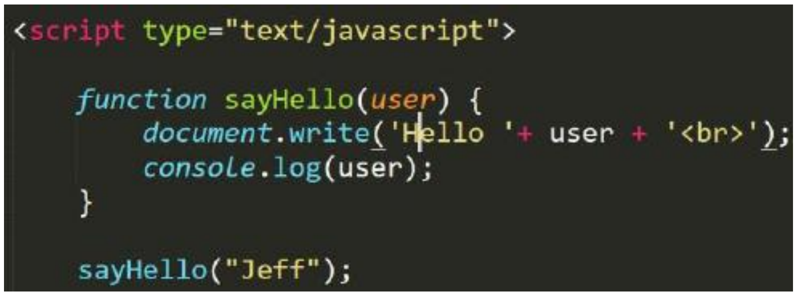


```html
<!DOCTYPE html>
<html>
<head>
    <title>My First Mean Calculator</title>
</head>
<body>
    <h1 style="text-align: center;"> My First Mean Calculator </h1>
    <script type="text/javascript">

    </script>
</body>
</html>
```

4. Create your first function, **sayHello(user)**, which prints a greeting to the user. Call the function

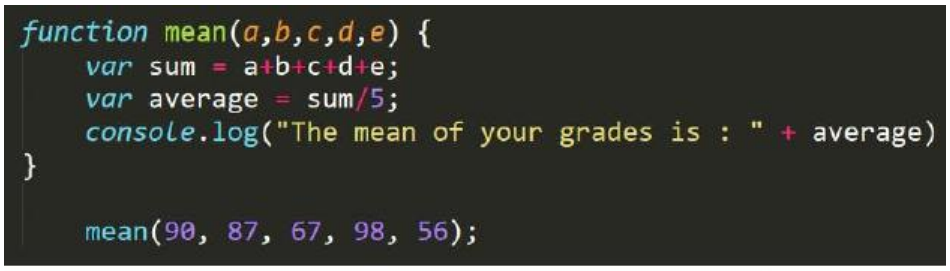
with your name as a parameter.

5. Inside **sayHello**, use **console.log(user)** to check the value of the user variable. Open Chrome, press **F12**, select **Console**, refresh the page, and your name should appear.

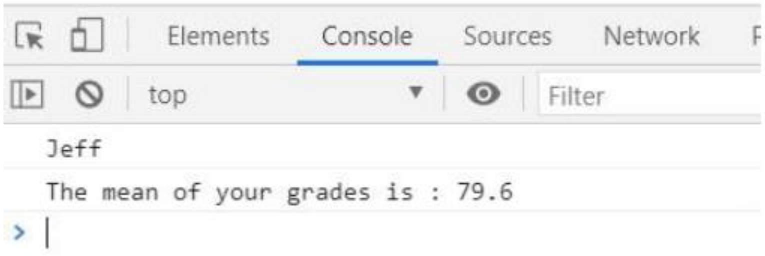


```
<script type="text/javascript">

    function sayHello(user) {
        document.write('Hello '+ user + '<br>');
        console.log(user);
    }

    sayHello("Jeff");
```

6. Create a **mean(a, b, c, d, e)** function:
   - Declare a variable **sum** to store the sum of all five parameters.
   - Declare a variable **average** to calculate the mean.
   - Display the average on the page.
   - Use **console.log()** to verify the values of sum and each parameter.



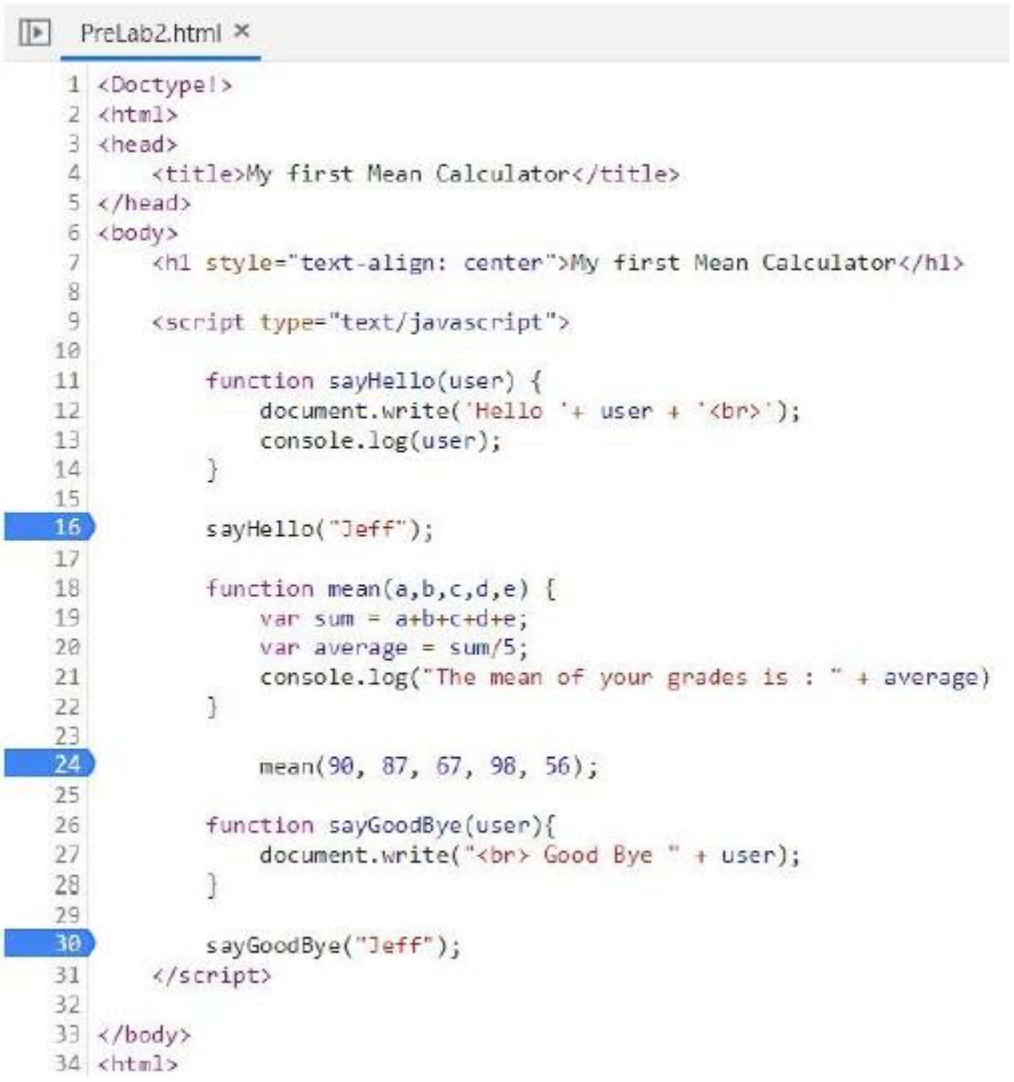
```
function mean(a,b,c,d,e) {
    var sum = a+b+c+d+e;
    var average = sum/5;
    console.log("The mean of your grades is : " + average)
}

    mean(90, 87, 67, 98, 56);
```



```
Jeff
The mean of your grades is : 79.6
>
```

7. Create a **sayGoodBye(user)** function that prints a goodbye message to a specific user. Call it with your name.

8. Use **breakpoints** to check your code execution:
   - In Chrome, open the **Sources** tab.
   - Set breakpoints on the three function calls by clicking the line numbers.
   - Refresh the page and trace the code to check variable values and detect any errors.

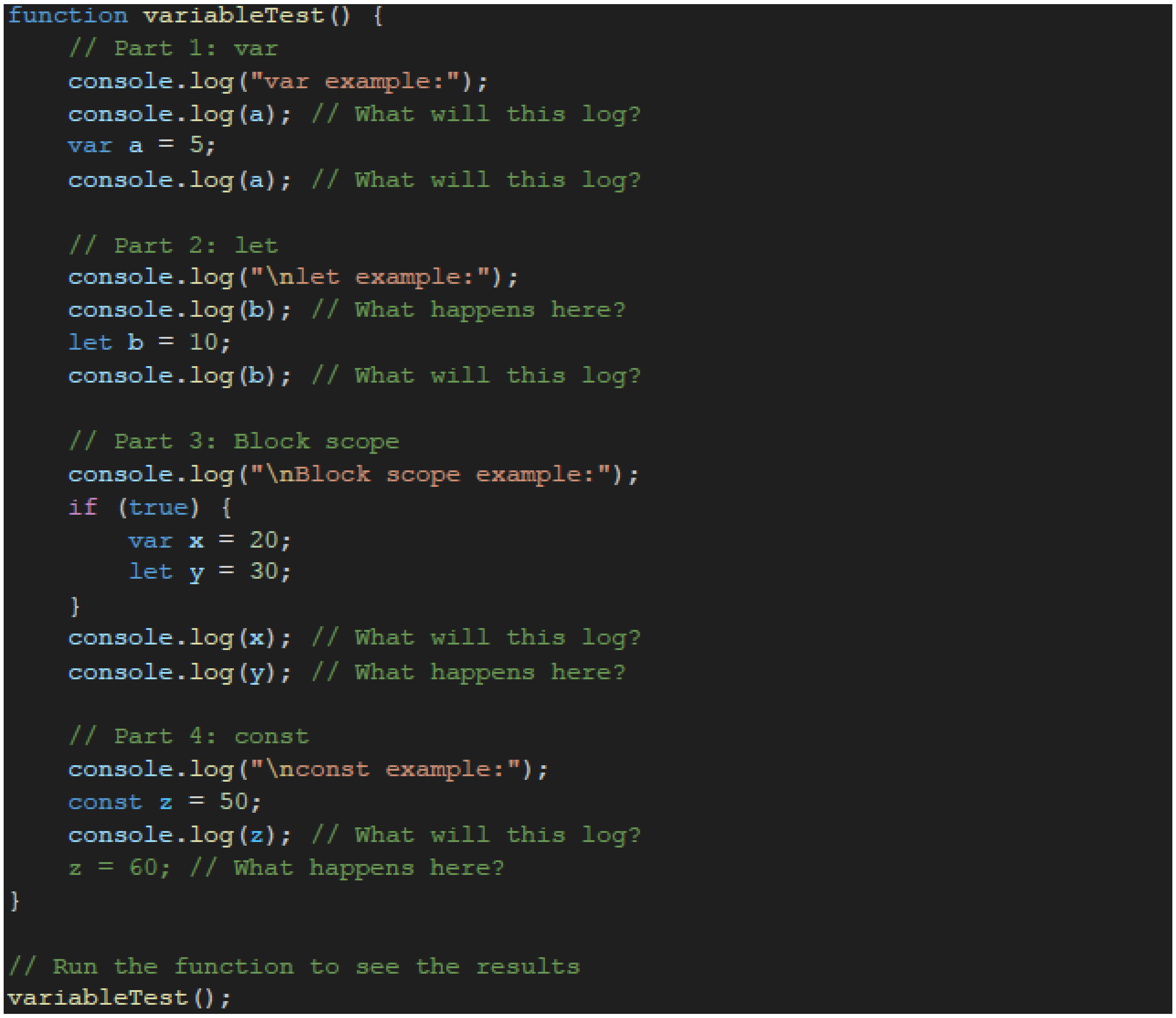


PreLab2.html ×

```
<Doctype!>
<html>
<head>
    <title>My first Mean Calculator</title>
</head>
<body>
    <h1 style="text-align: center">My first Mean Calculator</h1>

    <script type="text/javascript">

        function sayHello(user) {
            document.write('Hello '+ user + '<br>');
            console.log(user);
        }

        sayHello("Jeff");

        function mean(a,b,c,d,e) {
            var sum = a+b+c+d+e;
            var average = sum/5;
            console.log("The mean of your grades is : " + average)
        }

            mean(90, 87, 67, 98, 56);

        function sayGoodBye(user){
            document.write("<br> Good Bye " + user);
        }

        sayGoodBye("Jeff");
    </script>

</body>
<html>
```

Quick Exercise: *let*, *var*, and *const*

Run the following JavaScript script to compare the behavior of *let*, *var*, and *const* variables.



```
function variableTest() {
    // Part 1: var
    console.log("var example:");
    console.log(a); // What will this log?
    var a = 5;
    console.log(a); // What will this log?

    // Part 2: let
    console.log("\nlet example:");
    console.log(b); // What happens here?
    let b = 10;
    console.log(b); // What will this log?

    // Part 3: Block scope
    console.log("\nBlock scope example:");
    if (true) {
        var x = 20;
        let y = 30;
    }
    console.log(x); // What will this log?
    console.log(y); // What happens here?

    // Part 4: const
    console.log("\nconst example:");
    const z = 50;
    console.log(z); // What will this log?
    z = 60; // What happens here?
}

// Run the function to see the results
variableTest();
```

Key Observations:

- With **var**, pay attention to its hoisting behavior.
- With **let**, notice how it is limited to block scope.
- With **const**, remember that its value cannot be reassigned.