

## Lesson 05 Demo 02

### Working with IIFEs and Functions

**Objective:** To demonstrate the process of implementing Immediately Invoked Function Expressions (IIFEs) and functions in JavaScript for enhancing higher-order function usage, and efficient execution

**Tools required:** Visual Studio Code

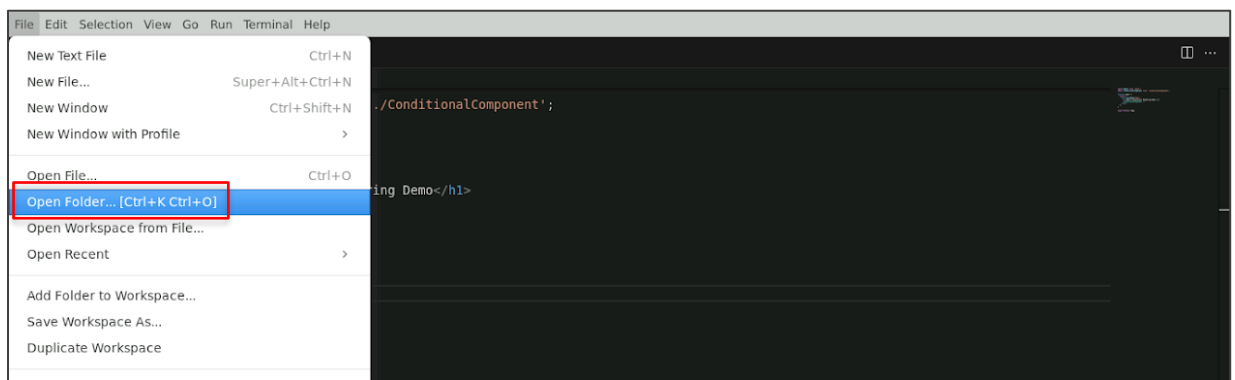
**Prerequisites:** None

Steps to be followed:

1. Write a JavaScript program using IIFEs and functions
2. Execute and verify the functionality of IIFEs and functions

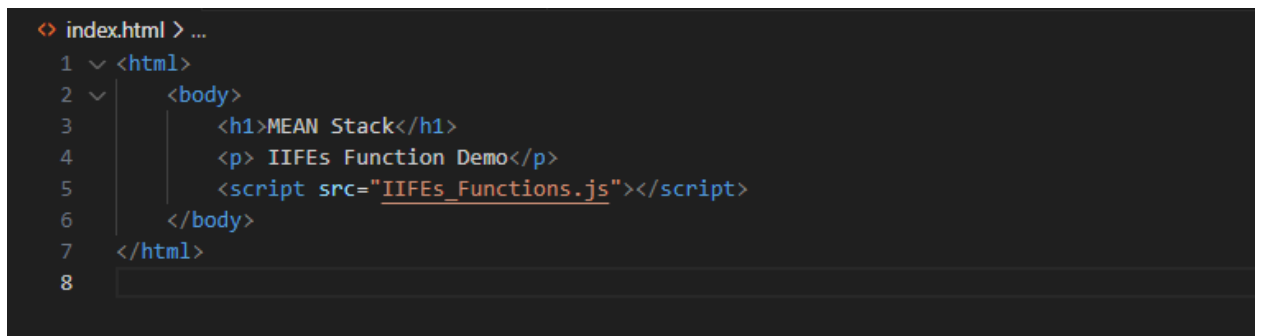
#### Step 1: Write a JavaScript program using IIFEs and functions

- 1.1 Navigate to Visual Studio Code, right-click on the **File** menu of the code editor, and select the **Open Folder...** option:



1.2 Write the given code in **index.html**:

```
<html>
  <body>
    <h1>MEAN Stack</h1>
    <p> IIFEs Function Demo</p>
    <script src="IIFEs_Functions.js"></script>
  </body>
</html>
```

A screenshot of a code editor with a dark background. The editor shows the HTML code for a file named 'index.html'. The code is as follows:

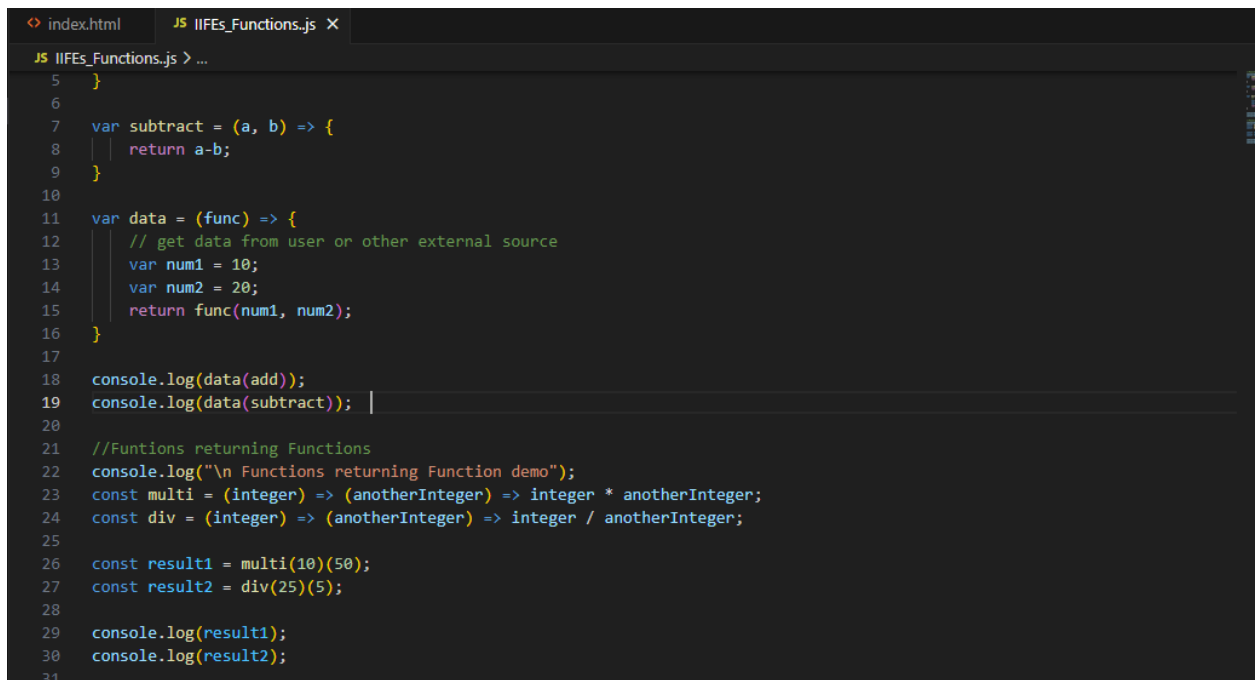
```
<html>
  <body>
    <h1>MEAN Stack</h1>
    <p> IIFEs Function Demo</p>
    <script src="IIFEs_Functions.js"></script>
  </body>
</html>
```

The code is displayed with syntax highlighting: HTML tags are in blue, text content is in white, and the script source file is in orange. Line numbers 1 through 8 are visible on the left side of the editor.

1.3 Click on the **src** folder of the project, select New File option, enter the filename as **IIFEs\_Functions.js**, and write the code shown below:

```
//Function as an argument
console.log("\n Functions as an argument demo");
var add = (a, b) => {
  return a+b;
}
var subtract = (a, b) => {
  return a-b;
}
var data = (func) => {
  // get data from user or other external source
  var num1 = 10;
  var num2 = 20;
  return func(num1, num2);
}
console.log(data(add));
console.log(data(subtract));
```

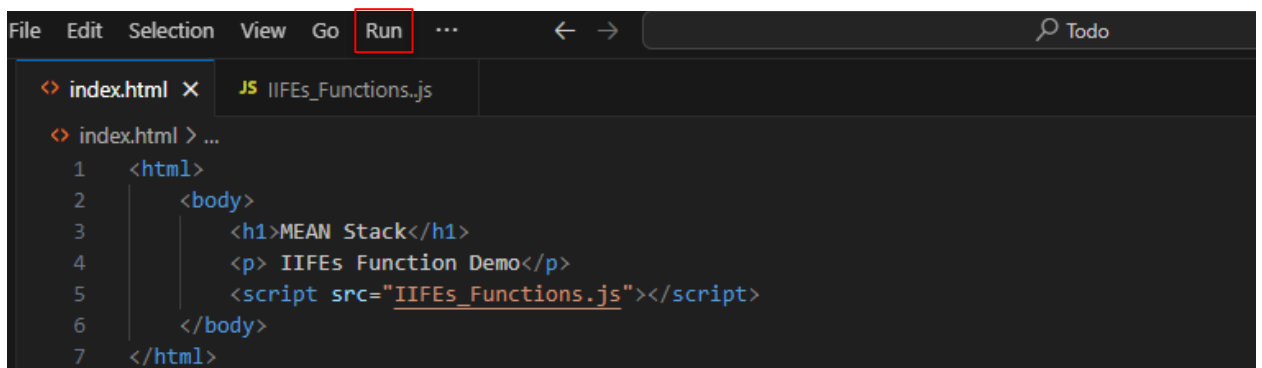
```
//Funtions returning Functions
console.log("\n Functions returning Function demo");
const multi = (integer) => (anotherInteger) => integer * anotherInteger;
const div = (integer) => (anotherInteger) => integer / anotherInteger;
const result1 = multi(10)(50);
const result2 = div(25)(5);
console.log(result1);
console.log(result2);
```



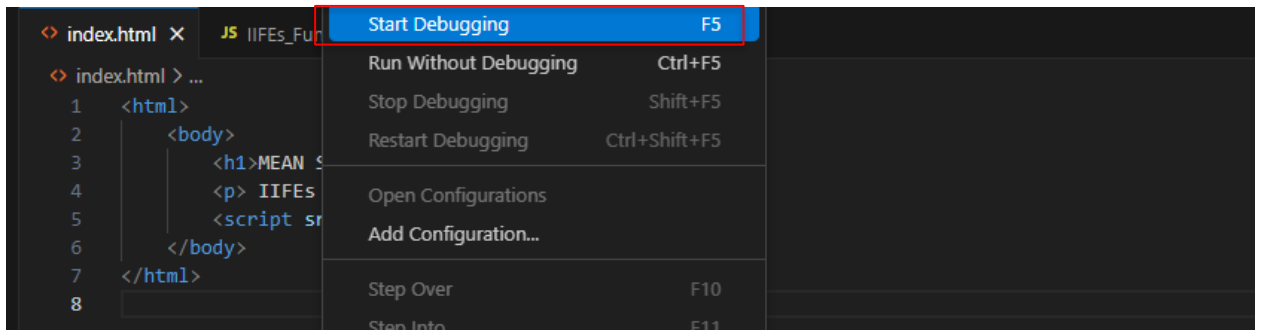
```
JS IIFEs_Functions.js > ...
5   }
6
7   var subtract = (a, b) => {
8     |   return a-b;
9   }
10
11  var data = (func) => {
12    |   // get data from user or other external source
13    |   var num1 = 10;
14    |   var num2 = 20;
15    |   return func(num1, num2);
16  }
17
18  console.log(data(add));
19  console.log(data(subtract)); |
20
21  //Funtions returning Functions
22  console.log("\n Functions returning Function demo");
23  const multi = (integer) => (anotherInteger) => integer * anotherInteger;
24  const div = (integer) => (anotherInteger) => integer / anotherInteger;
25
26  const result1 = multi(10)(50);
27  const result2 = div(25)(5);
28
29  console.log(result1);
30  console.log(result2);
31
```

## Step 2: Execute and verify the functionality of IIFEs and functions

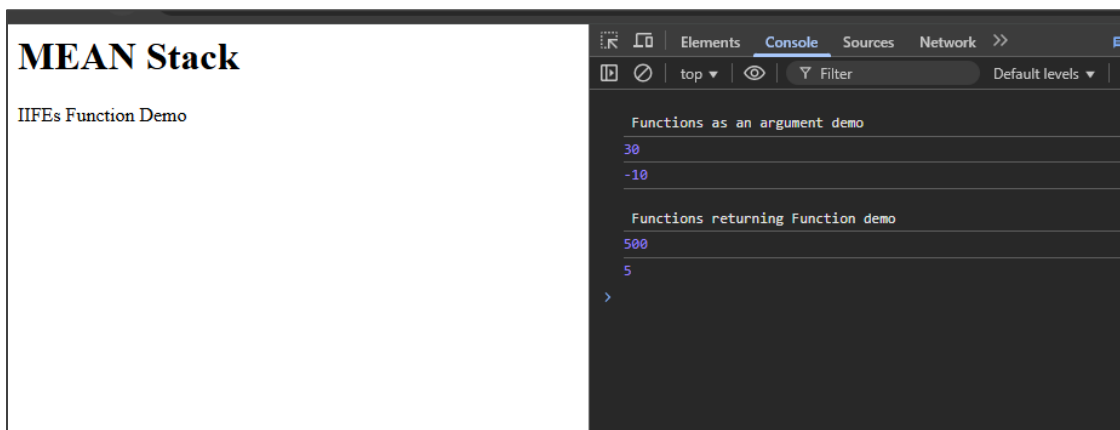
2.1 Click on **Run** and then on **Start Debugging** to execute the JavaScript file



```
File Edit Selection View Go Run ...
index.html x JS IIFEs_Functions.js
index.html > ...
1  <html>
2    <body>
3      <h1>MEAN Stack</h1>
4      <p> IIFEs Function Demo</p>
5      <script src="IIFEs_Functions.js"></script>
6    </body>
7  </html>
```



2.2 When the server starts running, right-click and select the **Inspect Element** option and click on the **Console** tab to see the output



By following the above steps, you have successfully implemented IIFEs and functions in JavaScript, demonstrating their role in modular programming, higher-order function execution, and functional programming techniques for cleaner and more reusable code.