

Lesson 05 Demo 03

Implementing Maps and Classes

Objective: To implement maps and classes in JavaScript, focusing on enhancing data handling and object-oriented programming practices to optimize functionality within a web development framework

Tools required: Visual Studio Code

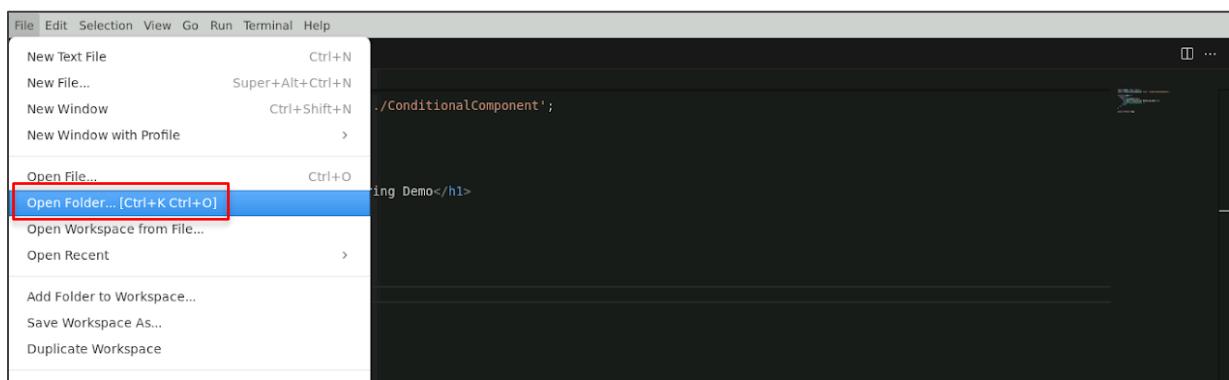
Prerequisites: None

Steps to be followed:

1. Create a JavaScript program using maps and classes
2. Test and verify their functionality

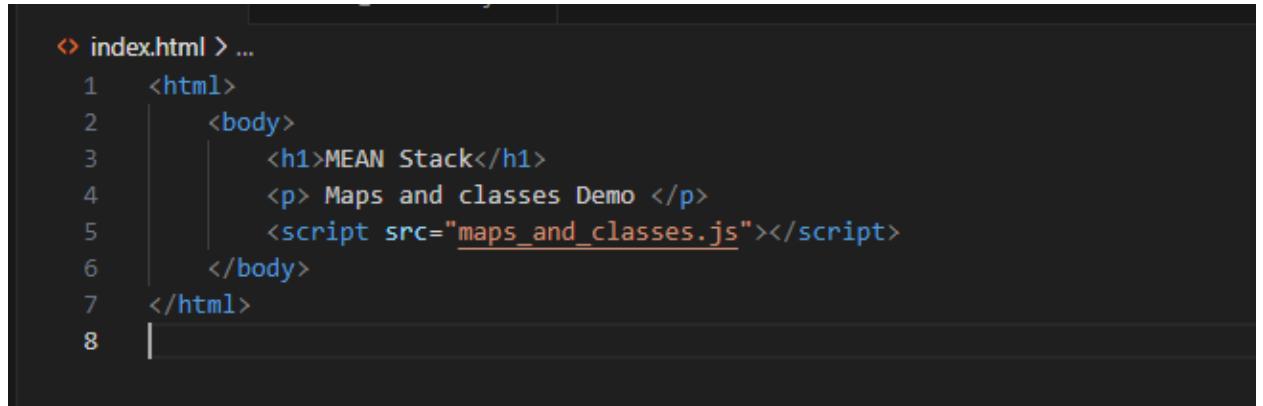
Step 1: Create a JavaScript program using maps and classes

- 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 following code in **index.html**:

```
<html>
  <body>
    <h1>MEAN Stack</h1>
    <p> Maps and classes Demo </p>
    <script src="maps_and_classes.js"></script>
  </body>
</html>
```



```
④ index.html > ...
1  <html>
2  |   <body>
3  |   |   <h1>MEAN Stack</h1>
4  |   |   <p> Maps and classes Demo </p>
5  |   |   <script src="maps_and_classes.js"></script>
6  |   </body>
7  </html>
8  |
```

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

```
var map1 = new Map();
map1.set("first name", "Robb");
map1.set("last name", "Stark");
map1.set("friend 1","Bran")
  .set("friend 2","Arya");
console.log(map1);
console.log("map1 has friend 3 ? " + map1.has("friend 3"));
console.log("get value for key = friend 3 - "+ map1.get("friend 3"));
console.log("delete element with key = friend 2 - " + map1.delete("friend 2"));
map1.clear();
console.log(map1);
class Employee
{
  constructor(id,name)
  {
```

```

this.id=id;
this.name=name;
}
detail()
{
document.writeln(this.id+" "+this.name+"<br>")
}
}

//passing object to a variable
var e1=new Employee(101,"Michael");
var e2=new Employee(102,"Bob");
e1.detail();
e2.detail();

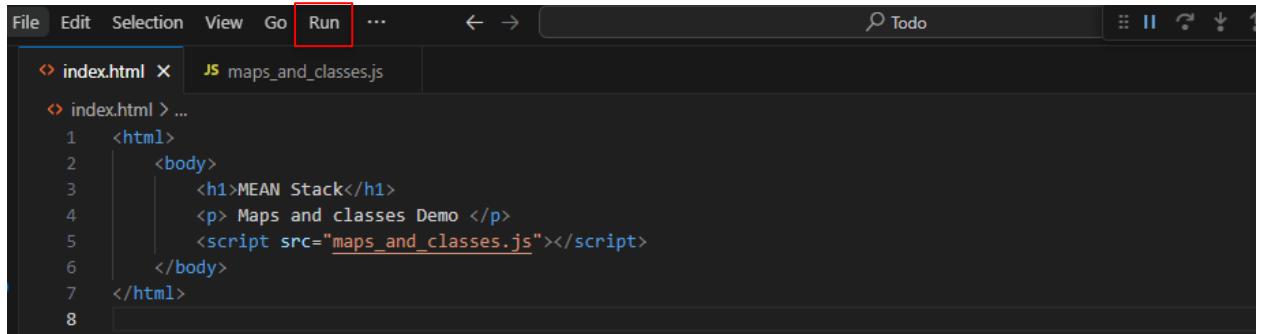
```

```

js maps_and_classes.js > ...
1  var map1 = new Map();
2  map1.set("first name", "Robb");
3  map1.set("last name", "Stark");
4  map1.set("friend 1","Bran")
5  | | .set("friend 2","Arya");
6  console.log(map1);
7  console.log("map1 has friend 3 ? " + map1.has("friend 3"));
8  console.log("get value for key = friend 3 - "+ map1.get("friend 3"));
9  console.log("delete element with key = friend 2 - " + map1.delete("friend 2"));
10 map1.clear();
11 console.log(map1);
12 class Employee
13 {
14   constructor(id,name)
15   {
16     this.id=id;
17     this.name=name;
18   }
19   detail()
20   {
21     document.writeln(this.id+" "+this.name+"<br>")
22   }
23 }
24 //passing object to a variable
25 var e1=new Employee(101,"Michael");
26 var e2=new Employee(102,"Bob");
27 e1.detail();
28 e2.detail();
```

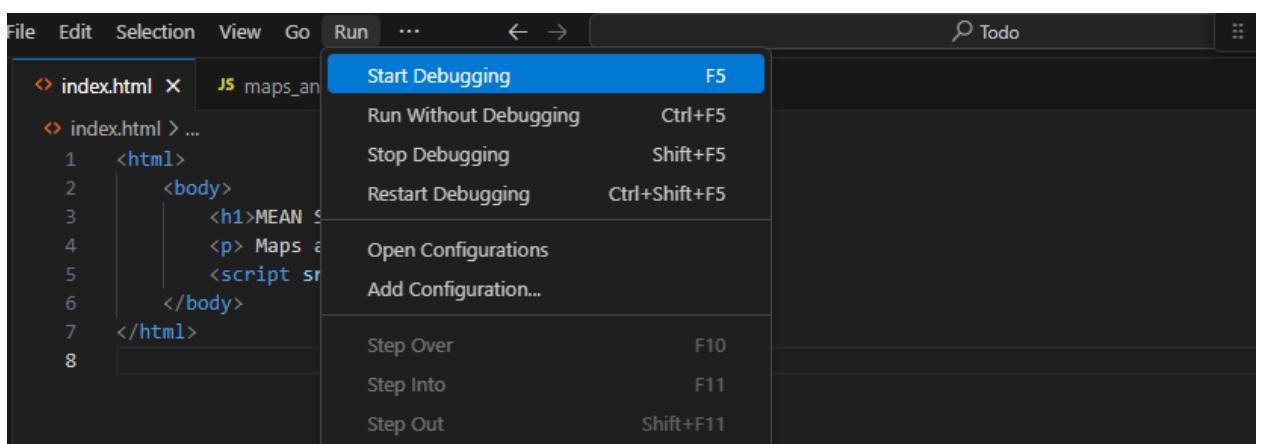
Step 2: Test and verify their functionality

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



The screenshot shows a code editor interface. At the top, there's a menu bar with File, Edit, Selection, View, Go, Run, and other options. The Run button is highlighted with a red box. Below the menu, there are two tabs: 'index.html X' and 'JS maps_and_classes.js'. The 'index.html' tab is active, displaying the following HTML code:

```
<html>
<body>
    <h1>MEAN Stack</h1>
    <p> Maps and classes Demo </p>
    <script src="maps_and_classes.js"></script>
</body>
</html>
```



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



By following the above steps, you have successfully created and tested a JavaScript program using maps and classes, effectively demonstrating their use in managing data and implementing object-oriented programming techniques within a web development framework.