

## Lesson 02 Demo 11

### Implementing Stacks Using Arrays

**Objective:** To implement a stack using arrays in JavaScript, covering key operations such as push, pop, peek, and display to enhance your data structure manipulation skills

**Tools required:** Visual Studio Code

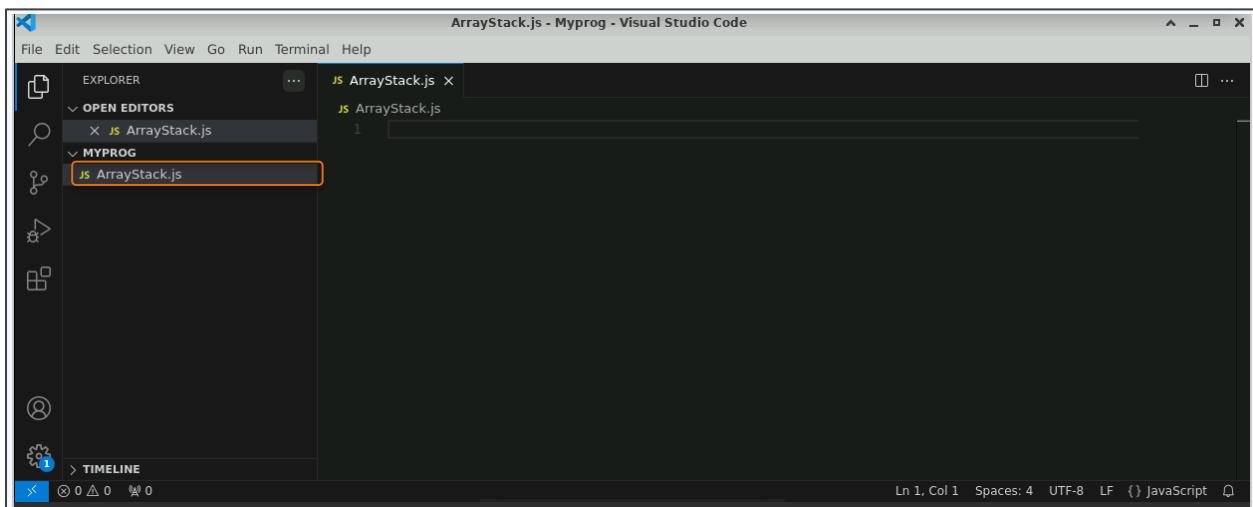
**Prerequisites:** A basic understanding of arrays in JavaScript

Steps to be followed:

1. Create a JavaScript file and execute it

#### Step 1: Create a JavaScript file and execute it

1.1 Open the Visual Studio Code editor and create a JavaScript file named **ArrayStack.js**



1.2 Add the following code to the file:

```
// Implementing a Stack Using an Array
class ArrayStack {
    constructor() {
        this.stack = [];
    }

    // Push operation
    push(item) {
        this.stack.push(item);
    }

    // Pop operation
    pop() {
        if (this.stack.length === 0) {
            return "Stack is empty";
        }
        return this.stack.pop();
    }

    // Top operation (Peek)
    top() {
        return this.stack[this.stack.length - 1];
    }

    // Check if the stack is empty
    isEmpty() {
        return this.stack.length === 0;
    }

    // Display the stack
    display() {
        console.log("Stack:", this.stack);
    }
}

// Using the ArrayStack
let stack = new ArrayStack();
// Pushing items
stack.push('Apple');
stack.push('Banana');
stack.push('Cherry');
```

```

// Displaying items
stack.display();

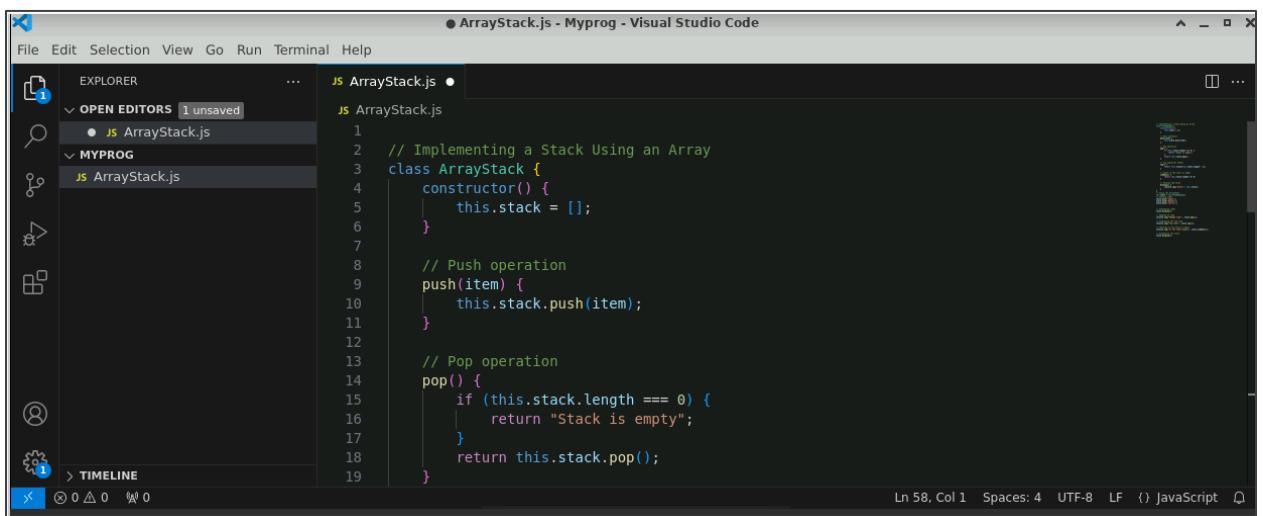
// Popping an item
console.log("Popped item:", stack.pop());

// Displaying the top item
console.log("Top item:", stack.top());

// Checking if the stack is empty
console.log("Is the stack empty?", stack.isEmpty());

// Displaying the stack
stack.display();

```

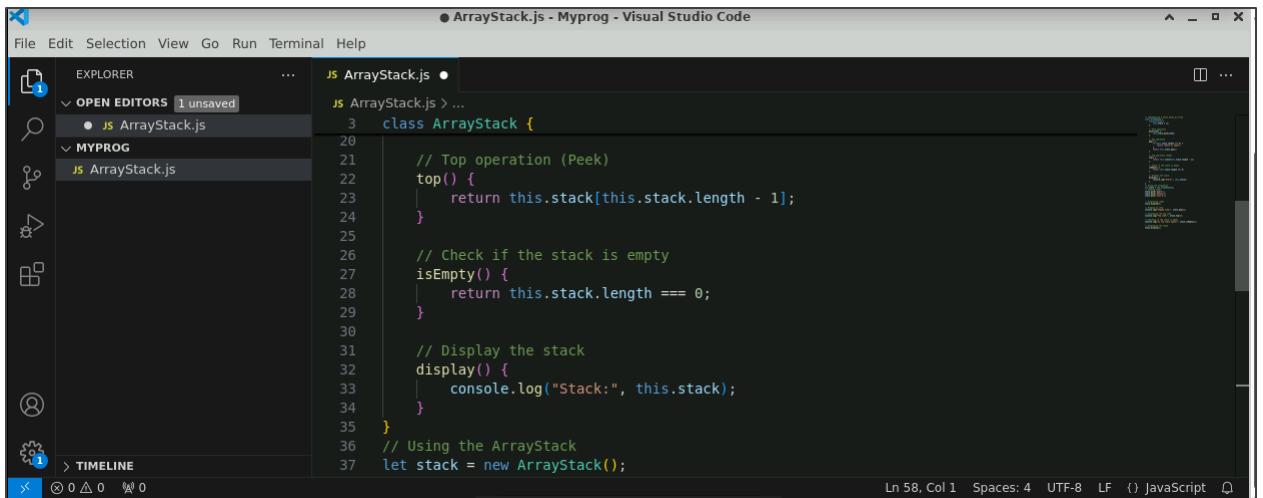


```

ArrayStack.js - Myprog - Visual Studio Code
File Edit Selection View Go Run Terminal Help
EXPLORER OPEN EDITORS 1 unsaved JS ArrayStack.js ...
JS ArrayStack.js
1 // Implementing a Stack Using an Array
2 class ArrayStack {
3     constructor() {
4         this.stack = [];
5     }
6
7     // Push operation
8     push(item) {
9         this.stack.push(item);
10    }
11
12     // Pop operation
13     pop() {
14         if (this.stack.length === 0) {
15             return "Stack is empty";
16         }
17         return this.stack.pop();
18     }
19

```

Ln 58, Col 1 Spaces: 4 UTF-8 LF {} JavaScript



```

ArrayStack.js - Myprog - Visual Studio Code
File Edit Selection View Go Run Terminal Help
EXPLORER OPEN EDITORS 1 unsaved JS ArrayStack.js ...
JS ArrayStack.js > ...
3 class ArrayStack {
20
21     // Top operation (Peek)
22     top() {
23         return this.stack[this.stack.length - 1];
24     }
25
26     // Check if the stack is empty
27     isEmpty() {
28         return this.stack.length === 0;
29     }
30
31     // Display the stack
32     display() {
33         console.log("Stack:", this.stack);
34     }
35 }
36 // Using the ArrayStack
let stack = new ArrayStack();

```

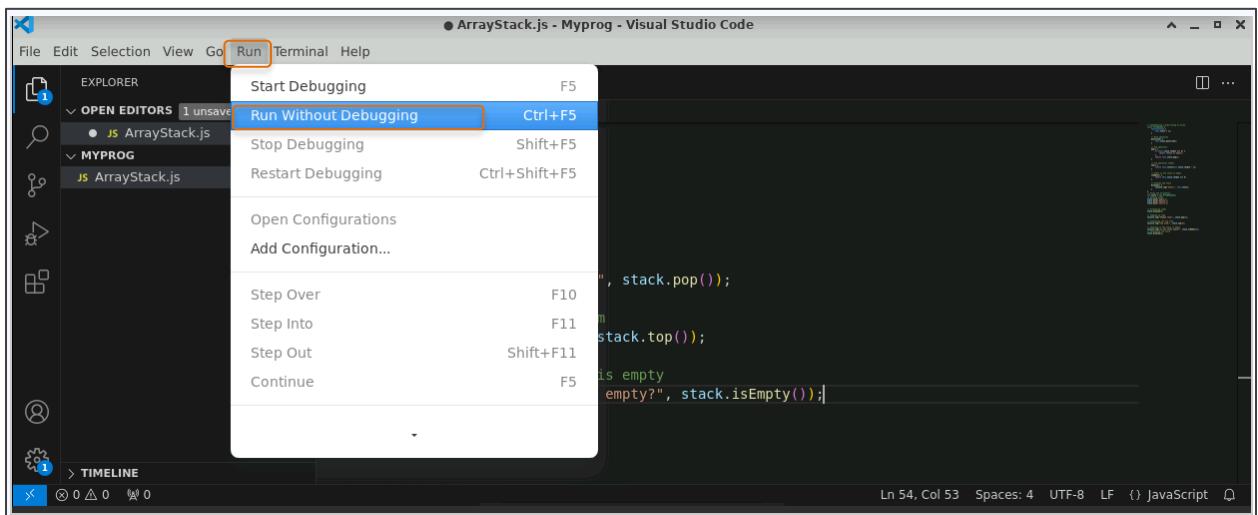
Ln 58, Col 1 Spaces: 4 UTF-8 LF {} JavaScript

The screenshot shows the Visual Studio Code interface with the title bar "ArrayStack.js - Myprog - Visual Studio Code". The left sidebar has sections for "EXPLORER", "OPEN EDITORS" (with "ArrayStack.js" listed), "MYPROG", and "TIMELINE". The main editor area displays the following JavaScript code:

```
JS ArrayStack.js ●
JS ArrayStack.js > ...
38 // Pushing items
39 stack.push('Apple');
40 stack.push('Banana');
41 stack.push('Cherry');
42
43
44 // Displaying items
45 stack.display();
46
47 // Popping an item
48 console.log("Popped item:", stack.pop());
49
50 // Displaying the top item
51 console.log("Top item:", stack.top());
52
53 // Checking if the stack is empty
54 console.log("Is the stack empty?", stack.isEmpty());
55 // Displaying the stack
56 stack.display();
```

The status bar at the bottom right shows "Ln 54, Col 53 Spaces: 4 UTF-8 LF {} JavaScript".

1.3 Click Run and then Run Without Debugging. Select Node.js to check the output in the DEBUG CONSOLE.



The screenshot shows the Visual Studio Code interface with the title bar "ArrayStack.js - Myprog - Visual Studio Code". The "File" menu is open, showing options like "Edit", "Selection", "View", "Go", "Run", "Terminal", and "Help". The "OPEN EDITORS" section shows "ArrayStack.js" in the "MYPROG" folder. A context menu titled "Select debugger" is open, with "Node.js" highlighted. The main editor area contains JavaScript code for a stack implementation. The status bar at the bottom right shows "Ln 54, Col 53" and "JavaScript".

#### 1.4 View the output in the **DEBUG CONSOLE** as shown below:

The screenshot shows the Visual Studio Code interface with the title bar "ArrayStack.js - Myprog - Visual Studio Code". The "File" menu is open, showing options like "Edit", "Selection", "View", "Go", "Run", "Terminal", and "Help". The "OPEN EDITORS" section shows "ArrayStack.js" in the "MYPROG" folder. The "DEBUG CONSOLE" tab is active, showing the output of the script execution. The output window displays the following text:  
/usr/bin/node ./ArrayStack.js  
> Stack: (3) ['Apple', 'Banana', 'Cherry']  
Popped item: Cherry  
Top item: Banana  
Is the stack empty? false  
> Stack: (2) ['Apple', 'Banana']  
The status bar at the bottom right shows "Ln 54, Col 53" and "JavaScript".

**Note:** This example illustrates the creation of a stack with an array in JavaScript.

By following these steps, you have successfully implemented a stack using arrays in JavaScript, covering essential operations like push, pop, peek, and displaying the stack's contents to enhance your data structure manipulation skills.