

# 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

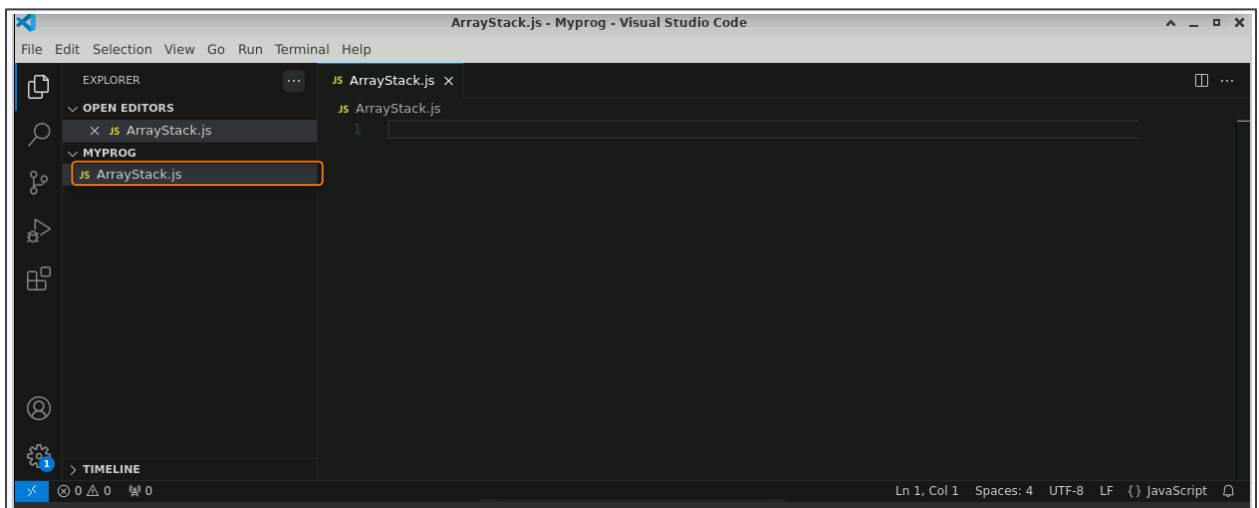
**Prerequisites:** A basic understanding of arrays in JavaScript

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');
```

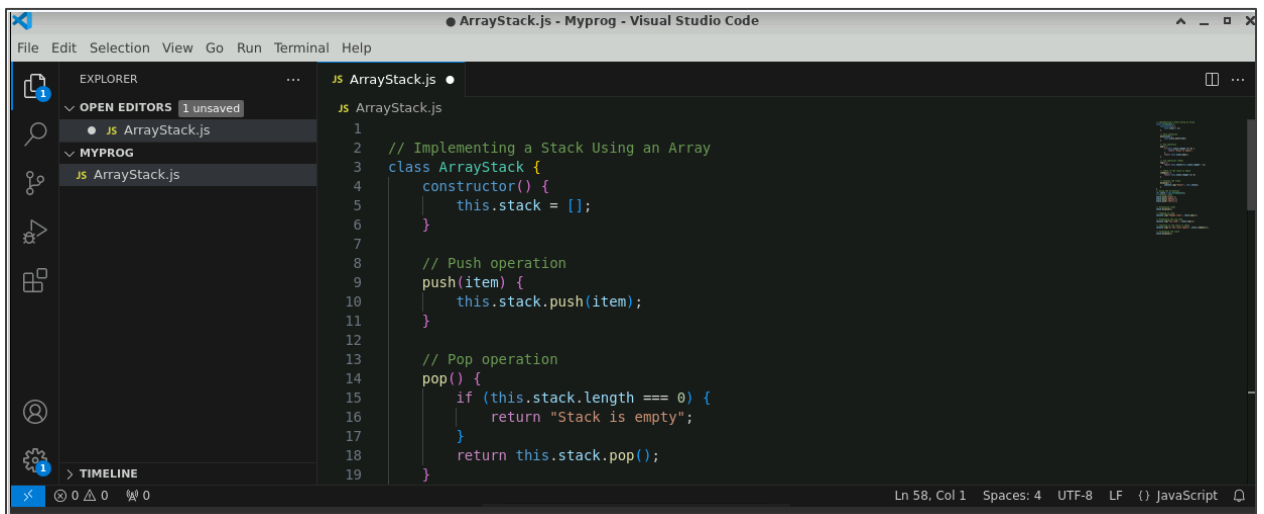
```
// Displaying items  
stack.display();
```

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

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

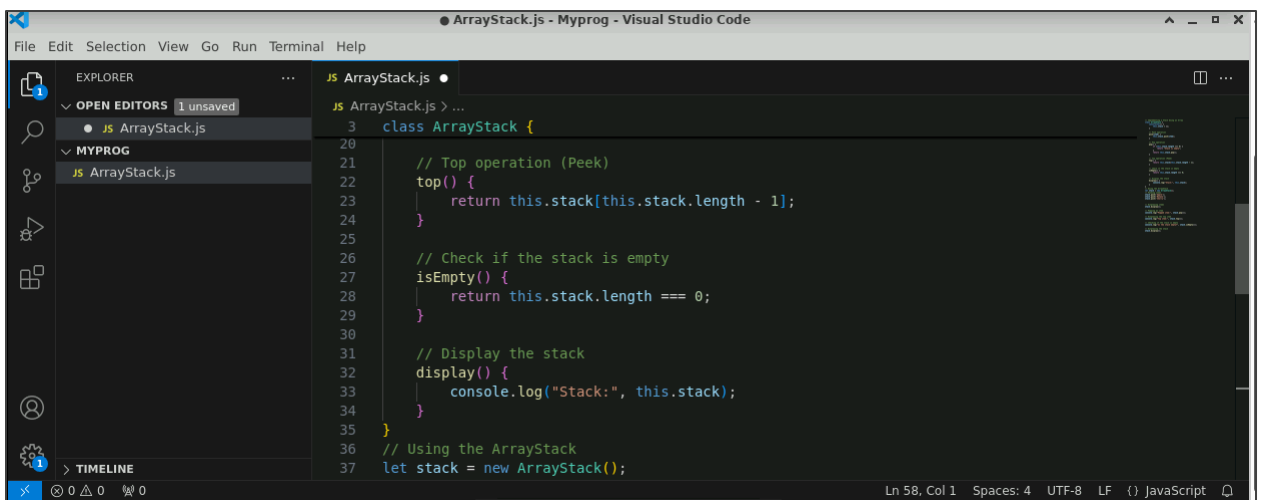
```
// Checking if the stack is empty  
console.log("Is the stack empty?", stack.isEmpty());
```

```
// Displaying the stack  
stack.display();
```



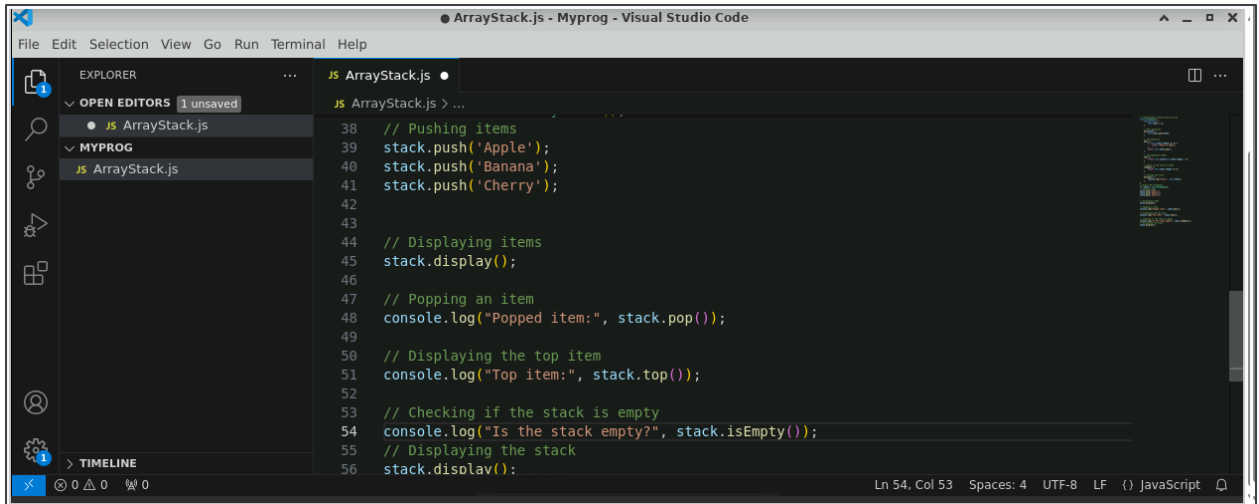
The screenshot shows the Visual Studio Code editor with the file 'ArrayStack.js' open. The code implements a stack using an array. The class 'ArrayStack' has a constructor that initializes 'this.stack' as an empty array. It includes methods for 'push(item)' which adds an item to the stack, and 'pop()' which removes the top item and returns it, with a check for an empty stack. The status bar at the bottom indicates 'Ln 58, Col 1', 'Spaces: 4', 'UTF-8', 'LF', and 'JavaScript'.

```
JS ArrayStack.js  
1  
2 // Implementing a Stack Using an Array  
3 class ArrayStack {  
4   constructor() {  
5     this.stack = [];  
6   }  
7  
8   // Push operation  
9   push(item) {  
10    this.stack.push(item);  
11  }  
12  
13  // Pop operation  
14  pop() {  
15    if (this.stack.length === 0) {  
16      return "Stack is empty";  
17    }  
18    return this.stack.pop();  
19  }
```

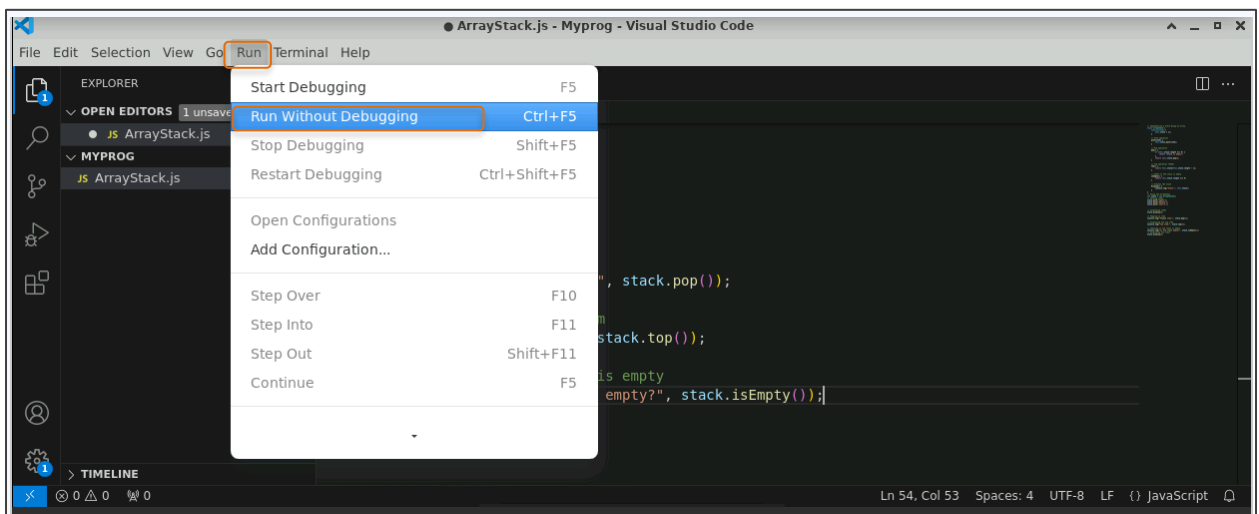


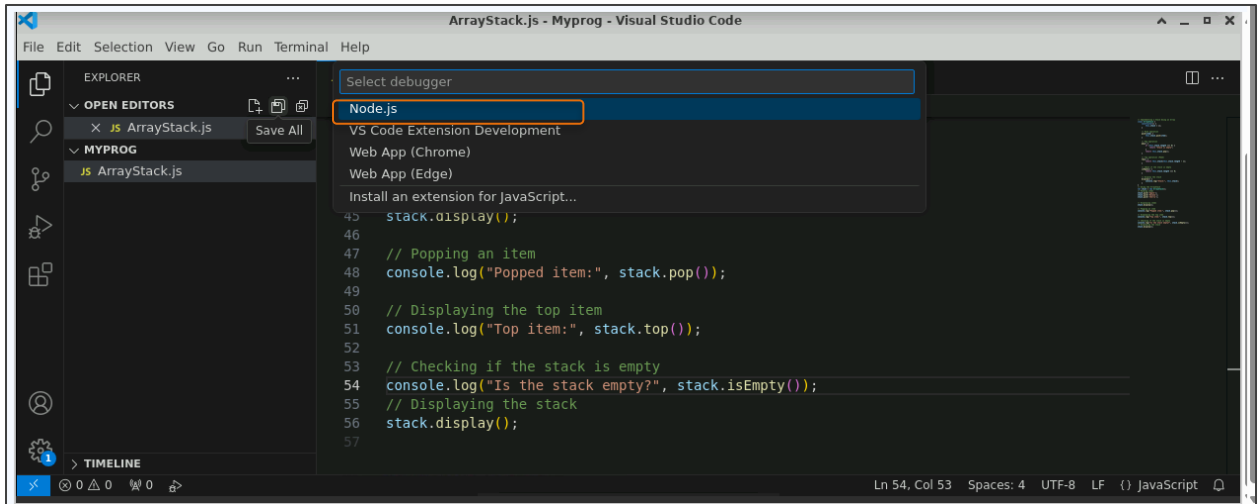
The screenshot shows the continuation of the 'ArrayStack.js' file in Visual Studio Code. It adds methods for 'top()' which returns the top element, 'isEmpty()' which checks if the stack is empty, and 'display()' which logs the stack's contents. Finally, it shows the instantiation of the stack with 'let stack = new ArrayStack();'. The status bar at the bottom indicates 'Ln 58, Col 1', 'Spaces: 4', 'UTF-8', 'LF', and 'JavaScript'.

```
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  
37 let stack = new ArrayStack();
```

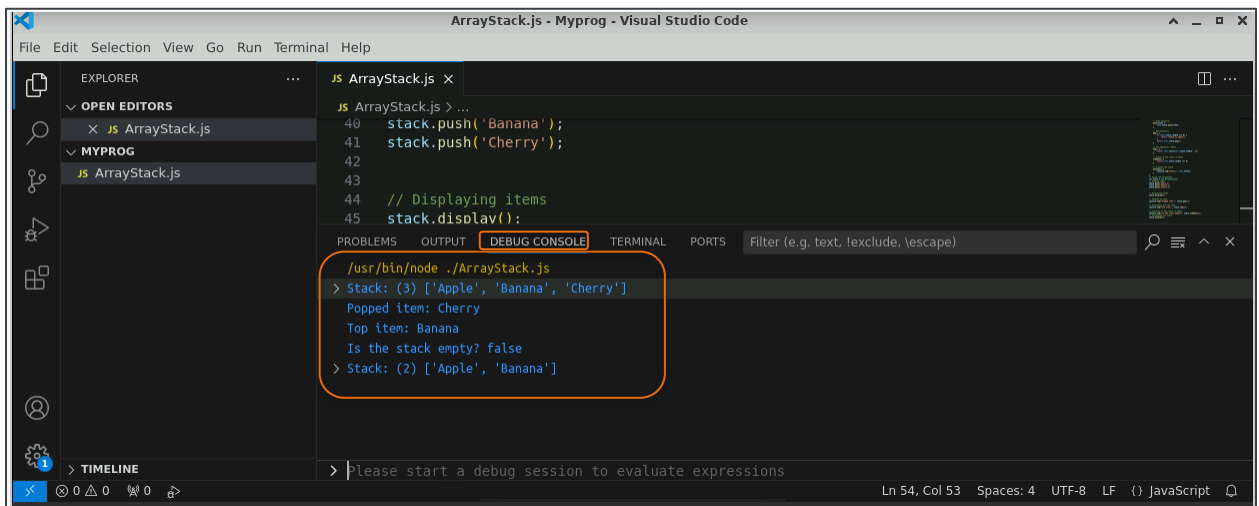


1.3 Click **Run** and then **Run Without Debugging**. Select **Node.js** to check the output in the **DEBUG CONSOLE**.





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



**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.