

# Lesson 03 Demo 05

## Implementing Arrays Methods

**Objective:** To implement the practical application of JavaScript array properties, methods, and iterators by initializing, manipulating, and iterating through an array, ensuring a clear understanding of array operations and their correctness

**Tools required:** Visual Studio Code and Node.js

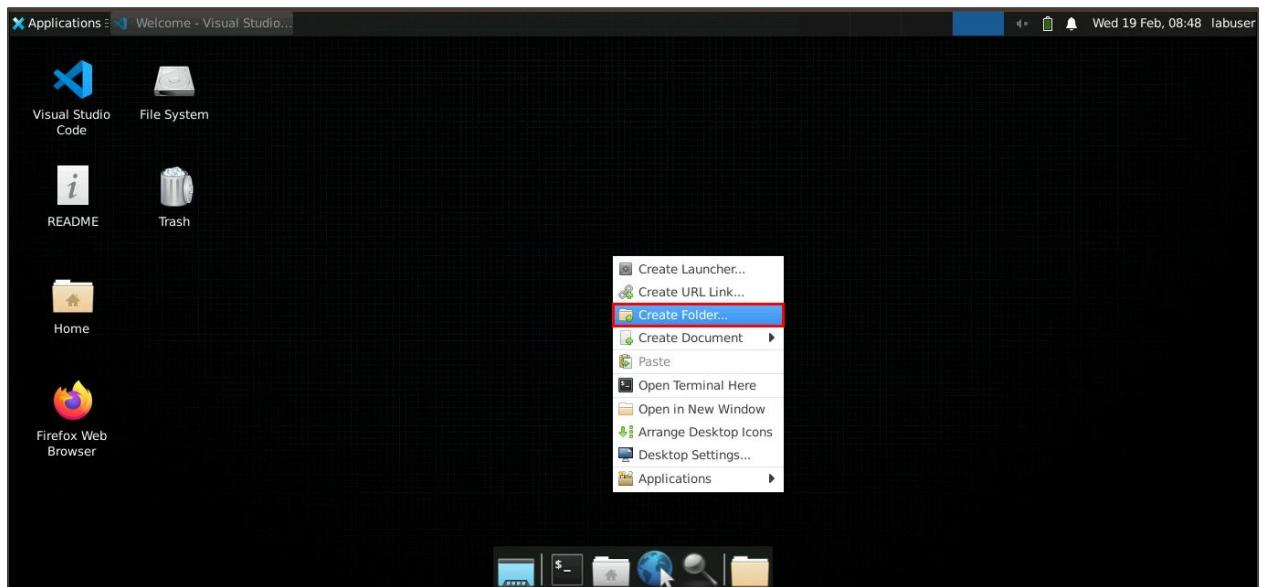
**Prerequisites:** A basic understanding of array properties, methods, and loops in JavaScript

Steps to be followed:

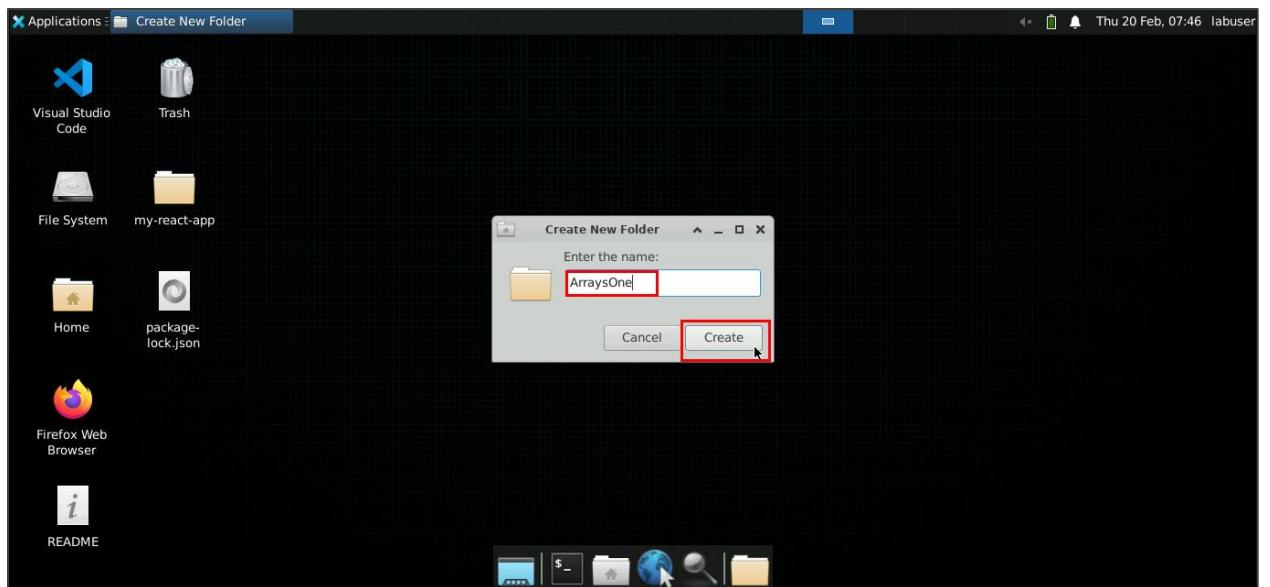
1. Create a folder named ArraysOne
2. Execute the JavaScript file

### Step 1: Create a folder named ArraysOne

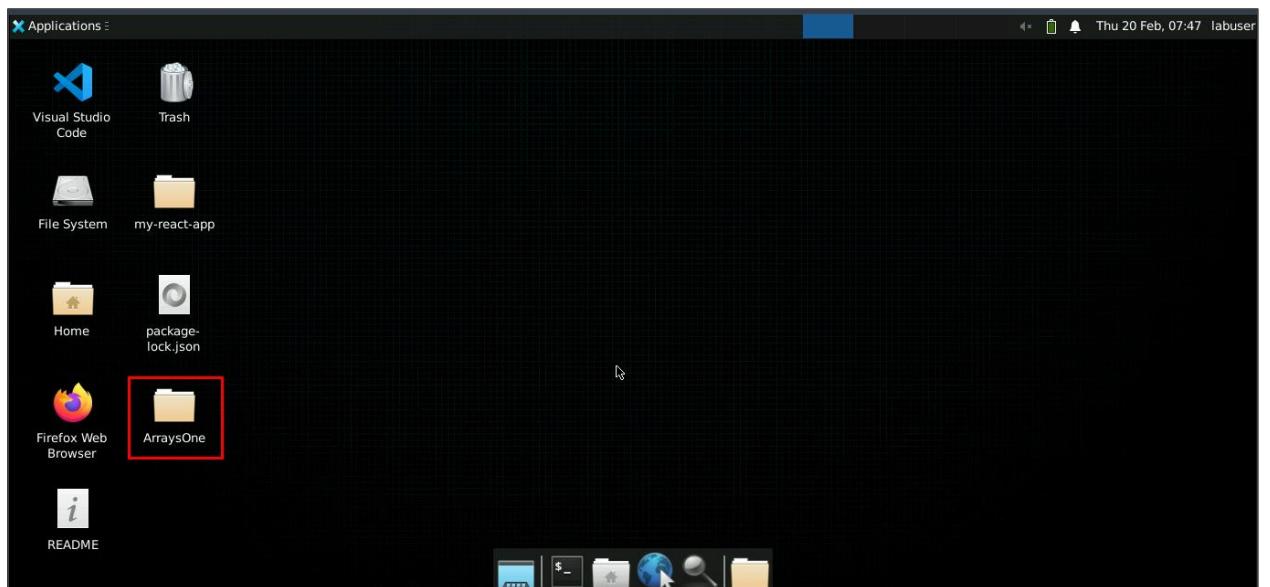
1.1 Right-click on the desktop and click on **Create Folder...**



1.2 Enter the name as **ArraysOne** and click on **Create**

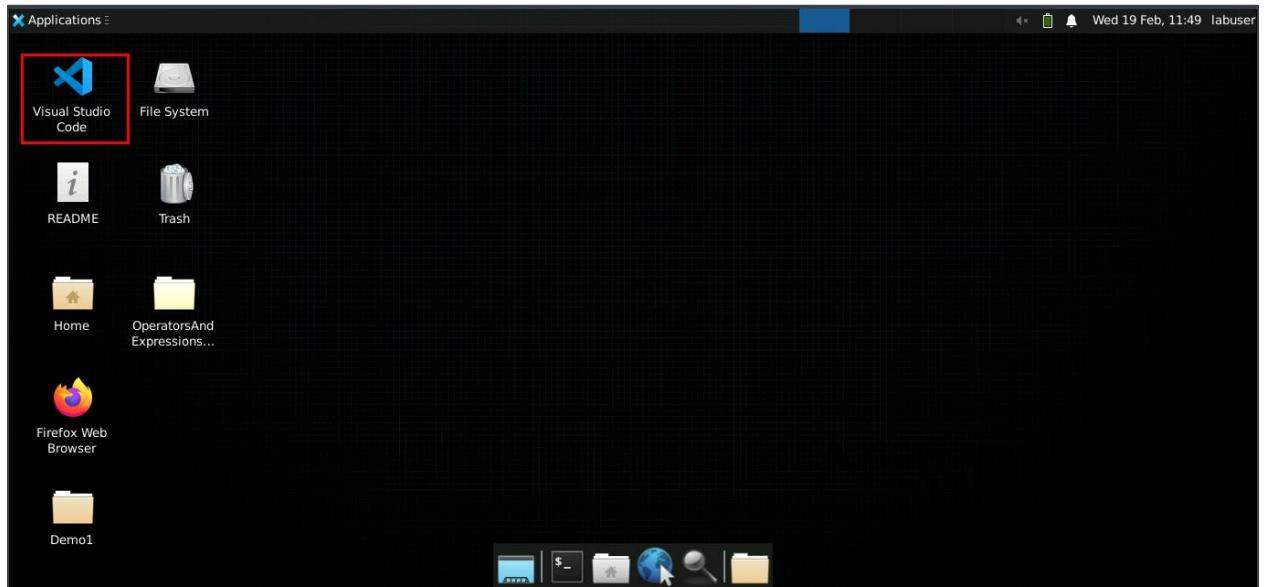


The **ArraysOne** folder gets created as shown below:

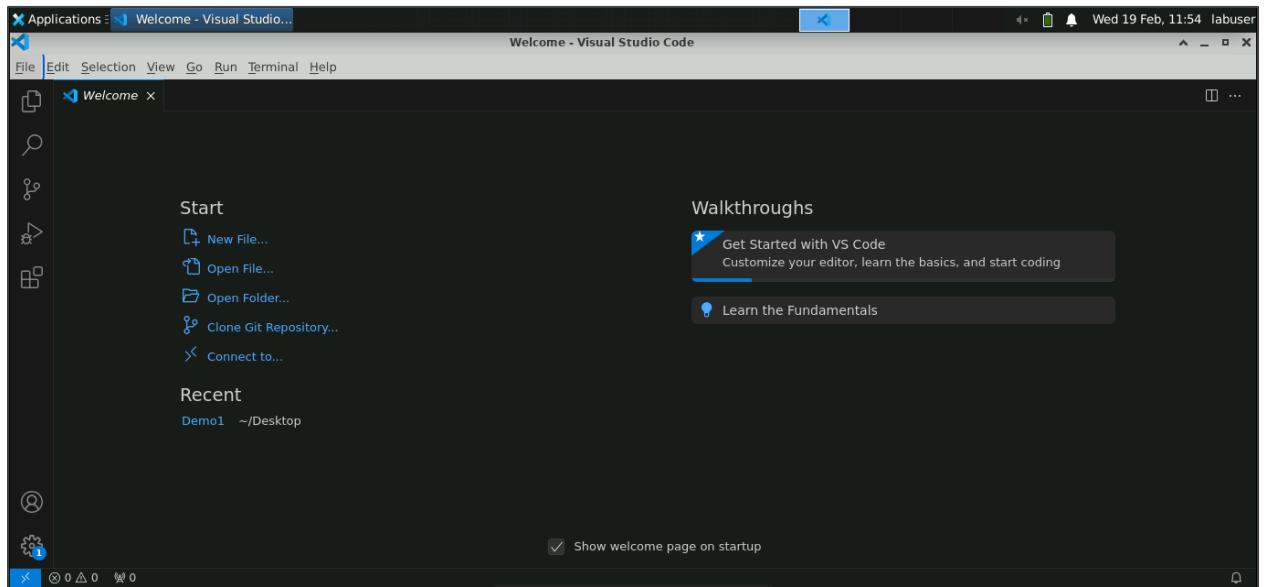


## Step 2: Execute the JavaScript file

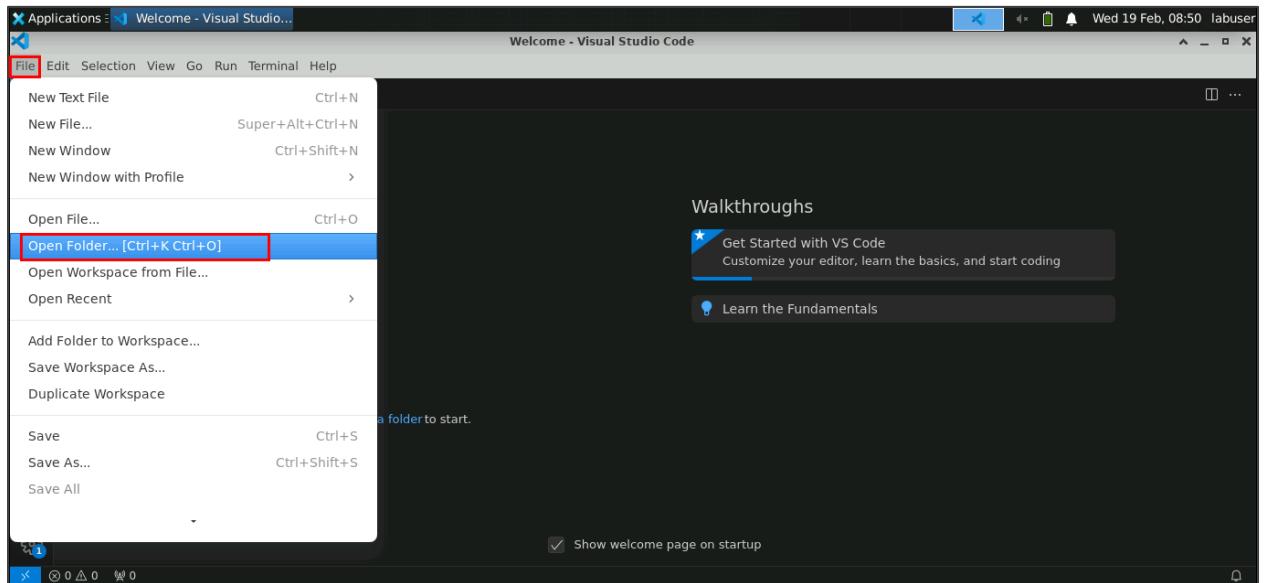
2.1 Double-click on the **Visual Studio Code** icon to open it



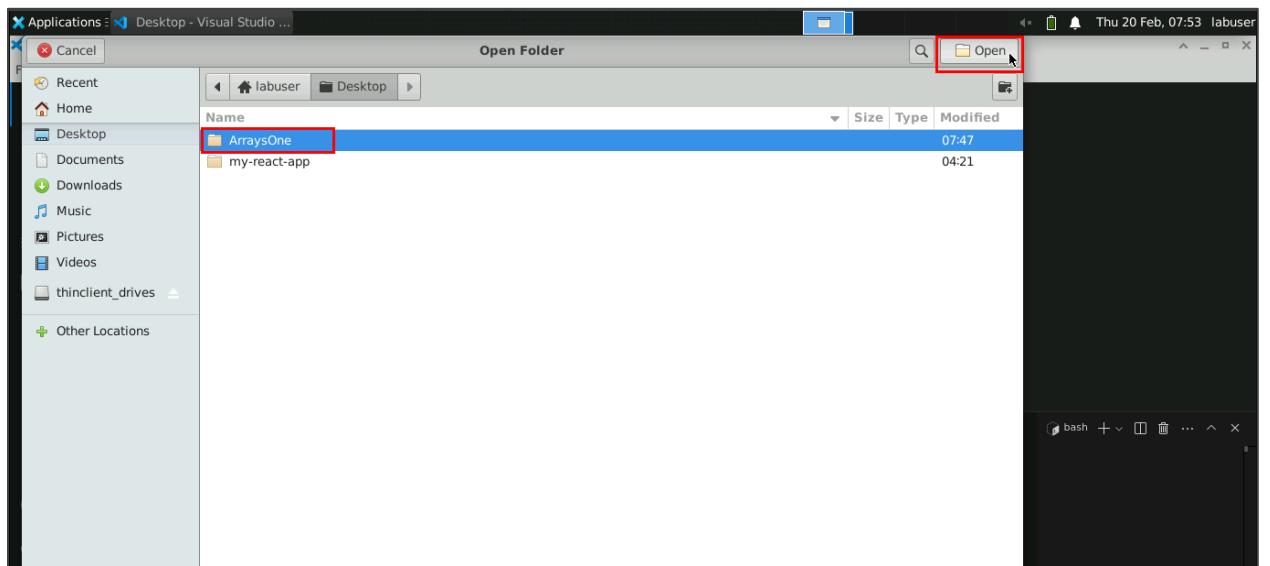
The **Visual Studio Code** opens as shown below:



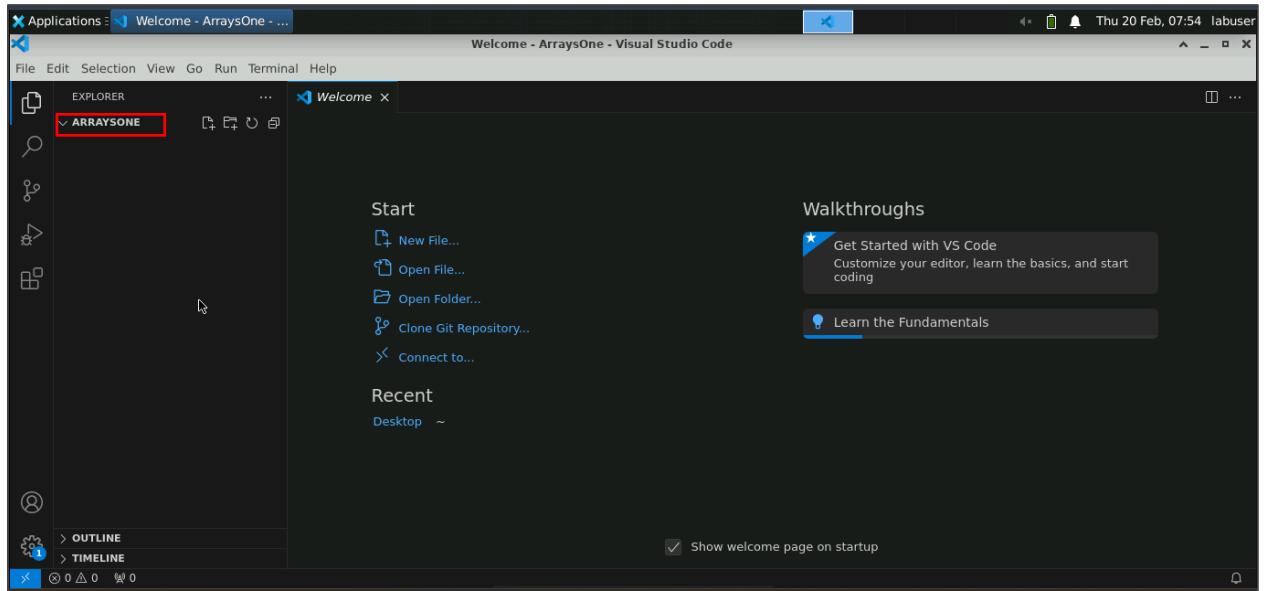
## 2.2 Click on **File**, then click on **Open Folder...**



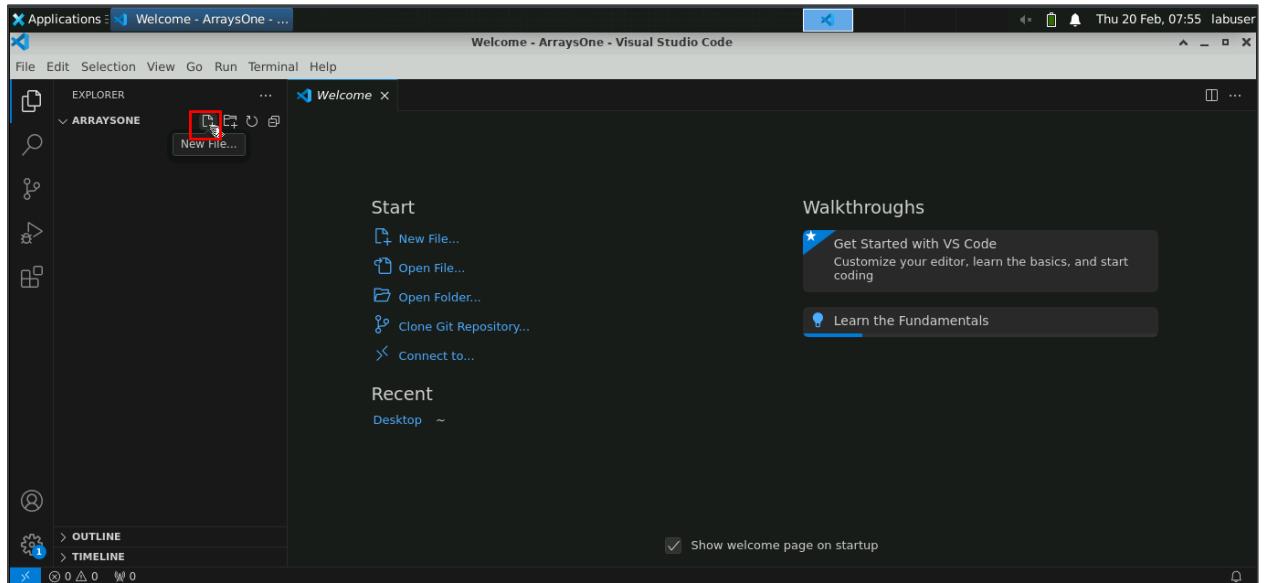
## 2.3 Select the **ArraysOne** folder and click on the **Open** icon to open the folder in Visual Studio Code



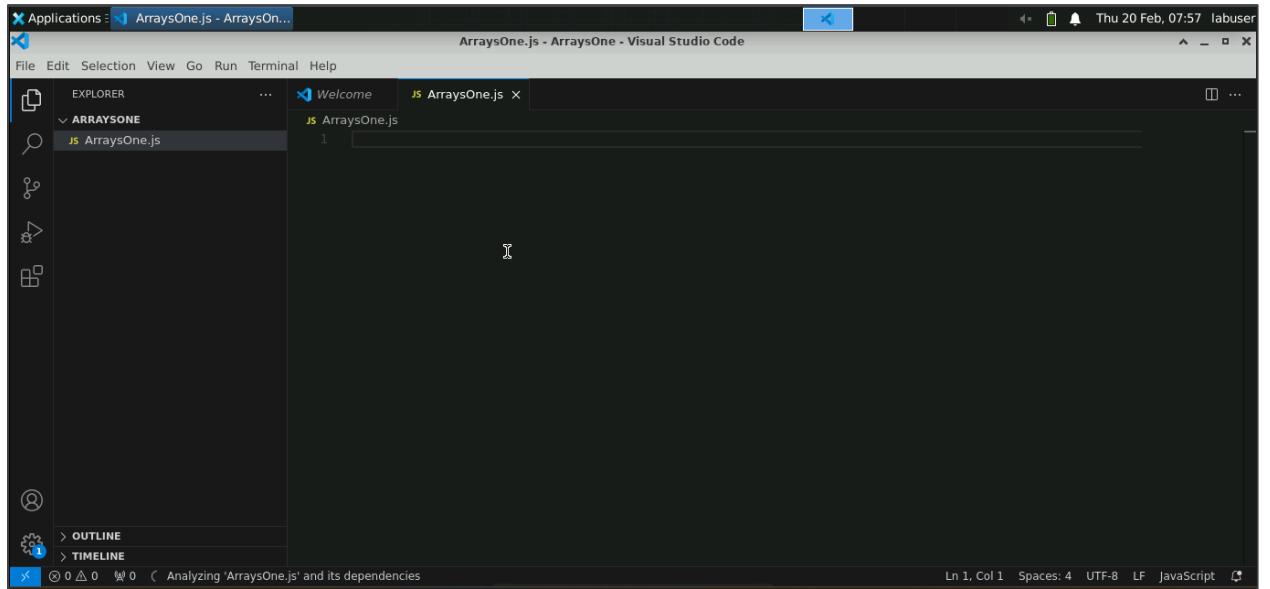
The folder opens in **Visual Studio Code** as shown below:



#### 2.4 Click on the **File** icon to create a new file named **ArraysOne.js**



The file gets created as shown below:



2.5 Enter the code below and save the file:

```
// Explore Array Properties
// Initialize an array and highlight the length property:
let myArray = [1, 2, 3, 4, 5];
console.log("Array Length:", myArray.length);

// Display the array's constructor property:
console.log("Array Constructor:", myArray.constructor);

// Access the prototype property of the array:
console.log("Array Prototype:", myArray.constructor.prototype);

//Utilize Array Methods
//Demonstrate the push method to add elements to the end of the array:

myArray.push(6, 7);
console.log("Array after push:", myArray);

// Use the pop method to remove the last element:

let poppedElement = myArray.pop();
console.log("Popped Element:", poppedElement);
console.log("Array after pop:", myArray);
```

```
// Apply the shift method to remove the first element:  
  
let shiftedElement = myArray.shift();  
console.log("Shifted Element:", shiftedElement);  
console.log("Array after shift:", myArray);  
  
// Utilize the unshift method to add elements to the beginning of the array:  
  
myArray.unshift(0, -1);  
console.log("Array after unshift:", myArray);  
  
// Create a new array and demonstrate the concat method to merge arrays:  
  
let anotherArray = [8, 9, 10];  
let mergedArray = myArray.concat(anotherArray);  
console.log("Merged Array:", mergedArray);  
  
// Use the join method to convert the array elements into a string:  
  
let joinedString = myArray.join(" | ");  
console.log("Joined String:", joinedString);  
  
// Employ the slice method to extract a portion of the array:  
  
let slicedArray = mergedArray.slice(2, 6);  
console.log("Sliced Array:", slicedArray);  
  
// Highlight the splice method to add and remove elements at a specific position:  
  
let splicedElements = mergedArray.splice(2, 3, "a", "b", "c");  
console.log("Spliced Elements:", splicedElements);  
console.log("Array after splice:", mergedArray);  
  
// Implement Iterator Methods  
  
// Use a traditional for-loop to iterate through the array:  
  
console.log("For-Loop Iteration:");  
for (let i = 0; i < mergedArray.length; i++) {  
  console.log(mergedArray[i]);  
}  
// Apply the forEach method for a cleaner iteration:  
console.log("forEach Iteration:");  
mergedArray.forEach(element => {
```

```

        console.log(element);
    });

// Utilize the map method to transform array elements:

let squaredValues = mergedArray.map(element => element * element);
console.log("Squared Values:", squaredValues);

// Demonstrate the filter method to create a new array with selected elements:

let filteredArray = mergedArray.filter(element => element % 2 === 0);
console.log("Filtered Array (Even Numbers):", filteredArray);

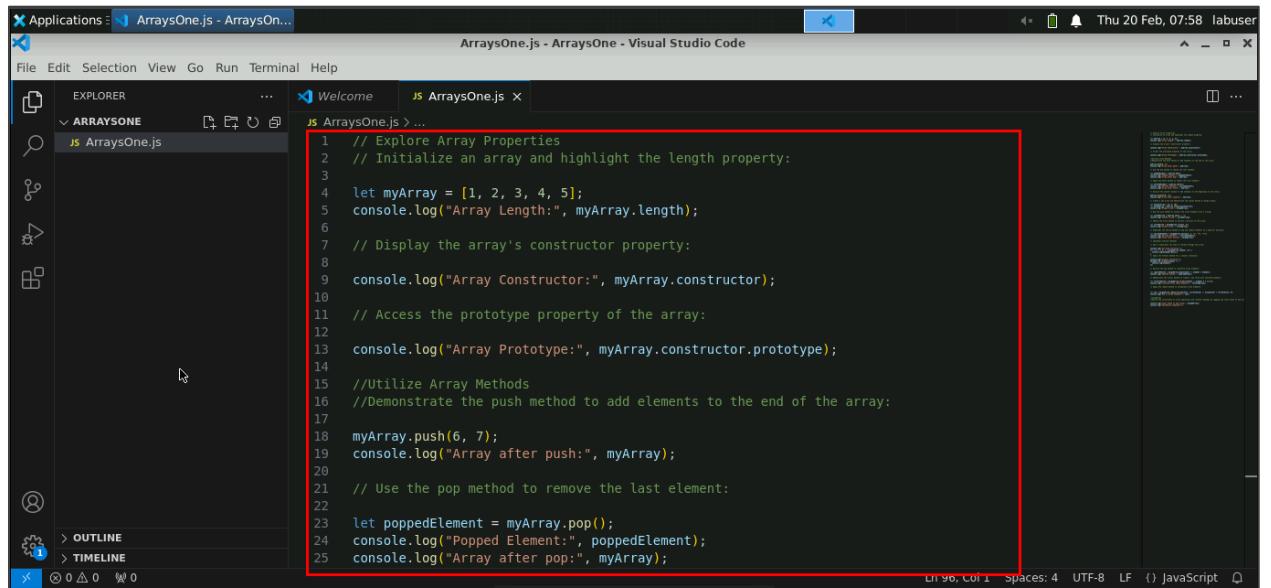
// Apply the reduce method to accumulate array elements:

let sum = mergedArray.reduce((accumulator, currentValue) => accumulator +
currentValue, 0);
console.log("Sum of Array Elements:", sum);

//validation
//Verify the correctness of array operations and iterator methods by logging the final
state of the array and the results of each method.

console.log("Final State of the Array:", mergedArray);
console.log("Validation Complete!");

```



The screenshot shows the Visual Studio Code interface with a dark theme. The title bar reads "Applications > ArraysOne.js - ArraysOne". The menu bar includes File, Edit, Selection, View, Go, Run, Terminal, Help. The left sidebar has an Explorer tab with a folder named "ARRAYSOne" containing "ArraysOne.js". The main editor area shows the following JavaScript code:

```

1 // Explore Array Properties
2 // Initialize an array and highlight the length property:
3
4 let myArray = [1, 2, 3, 4, 5];
5 console.log("Array Length:", myArray.length);
6
7 // Display the array's constructor property:
8
9 console.log("Array Constructor:", myArray.constructor);
10
11 // Access the prototype property of the array:
12
13 console.log("Array Prototype:", myArray.constructor.prototype);
14
15 //Utilize Array Methods
16 //Demonstrate the push method to add elements to the end of the array:
17
18 myArray.push(6, 7);
19 console.log("Array after push:", myArray);
20
21 // Use the pop method to remove the last element:
22
23 let poppedElement = myArray.pop();
24 console.log("Popped Element:", poppedElement);
25 console.log("Array after pop:", myArray);

```

The code from line 1 to line 25 is highlighted with a red rectangle. The status bar at the bottom right shows "Ln 96 Col 1 Spaces: 4 UTF-8 LF {} JavaScript".

A screenshot of Visual Studio Code showing a file named `ArraysOne.js`. The code demonstrates various array methods:

```
25 console.log("Array after pop:", myArray);
26
27 // Apply the shift method to remove the first element:
28
29 let shiftedElement = myArray.shift();
30 console.log("Shifted Element:", shiftedElement);
31 console.log("Array after shift:", myArray);
32
33 // Utilize the unshift method to add elements to the beginning of the array:
34
35 myArray.unshift(0, -1);
36 console.log("Array after unshift:", myArray);
37
38 // Create a new array and demonstrate the concat method to merge arrays:
39
40 let anotherArray = [8, 9, 10];
41 let mergedArray = myArray.concat(anotherArray);
42 console.log("Merged Array:", mergedArray);
43
44 // Use the join method to convert the array elements into a string:
45
46 let joinedString = myArray.join(" | ");
47 console.log("Joined String:", joinedString);
48
49 // Employ the slice method to extract a portion of the array:
```

A screenshot of Visual Studio Code showing the same `ArraysOne.js` file. The code continues from the previous snippet, demonstrating more array methods:

```
51 let slicedArray = mergedArray.slice(2, 6);
52 console.log("Sliced Array:", slicedArray);
53
54 // Highlight the splice method to add and remove elements at a specific position:
55
56 let splicedElements = mergedArray.splice(2, 3, "a", "b", "c");
57 console.log("Spliced Elements:", splicedElements);
58 console.log("Array after splice:", mergedArray);
59
60 // Implement Iterator Methods
61
62 // Use a traditional for-loop to iterate through the array:
63
64 console.log("For-Loop Iteration:");
65 for (let i = 0; i < mergedArray.length; i++) {
66 | console.log(mergedArray[i]);
67 }
68 // Apply the forEach method for a cleaner iteration:
69
70 console.log("forEach Iteration:");
71 mergedArray.forEach(element => {
72 | console.log(element);
73 });
74
75 // Utilize the map method to transform array elements:
```

A screenshot of the Visual Studio Code interface. The title bar shows "Applications: ArraysOne.js - ArraysOne...". The main area displays the content of "ArraysOne.js" with several lines of code highlighted by a red rectangular selection. The code uses various array methods like `forEach`, `map`, `filter`, `reduce`, and `push` to demonstrate array operations. The status bar at the bottom right indicates "Ln 96, Col 1 Spaces: 4 UTF-8 LF () JavaScript".

```
1 // Utilize the map method to transform array elements:  
2 let squaredValues = mergedArray.map(element => element * element);  
3 console.log("Squared Values:", squaredValues);  
4  
5 // Demonstrate the filter method to create a new array with selected elements:  
6 let filteredArray = mergedArray.filter(element => element % 2 === 0);  
7 console.log("Filtered Array (Even Numbers):", filteredArray);  
8  
9 // Apply the reduce method to accumulate array elements:  
10 let sum = mergedArray.reduce((accumulator, currentValue) => accumulator + currentValue, 0);  
11 console.log("Sum of Array Elements:", sum);  
12  
13 //validation  
14 //Verify the correctness of array operations and iterator methods by logging the final state of the array  
15  
16 console.log("Final State of the Array:", mergedArray);  
17 console.log("Validation Complete!");  
18
```

2.6 Save the file and run it using Node.js in the terminal:

**node ArraysOne.js**

A screenshot of the Visual Studio Code interface, similar to the previous one but with a different file open. The title bar shows "Applications: ArraysOne.js - ArraysOne...". The main area displays the content of "ArraysOne.js". At the bottom, the "TERMINAL" tab is active, showing a terminal window with the command "node ArraysOne.js" entered. The status bar at the bottom right indicates "Ln 96, Col 1 Spaces: 4 UTF-8 LF () JavaScript".

```
1 // Explore Array Properties  
2 // Initialize an array and highlight the length property:  
3  
4 let myArray = [1, 2, 3, 4, 5];  
5 console.log("Array Length:", myArray.length);  
6  
7 // Display the array's constructor property:  
8  
9 console.log("Array Constructor:", myArray.constructor);  
10  
11 // Access the prototype property of the array:  
12  
13 console.log("Array Prototype:", myArray.constructor.prototype);  
14  
15 //Utilize Array Methods  
16 //Demonstrate the push method to add elements to the end of the array:  
17  
18 myArray.push(6, 7);
```

Applications: ArraysOne.js - ArraysOne...

File Edit Selection View Go Run Terminal Help

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS

EXPLORER ARRAYSONE

JS ArraysOne.js

```
● labuser@ip-172-31-47-93:~/Desktop/ArraysOne$ node ArraysOne.js
Array Length: 5
Array Constructor: [Function: Array]
Array Prototype: Object(0) []
Array after push: [
  1, 2, 3, 4,
  5, 6, 7
]
Popped Element: 7
Array after pop: [ 1, 2, 3, 4, 5, 6 ]
Shifted Element: 1
Array after shift: [ 2, 3, 4, 5, 6 ]
Array after unshift: [
  0, -1, 2, 3,
  4, 5, 6
]
Merged Array: [
  0, -1, 2, 3,
  5, 6, 8, 9, 10
]
Joined String: 0 | -1 | 2 | 3 | 4 | 5 | 6
Sliced Array: [ 2, 3, 4, 5 ]
Spliced Elements: [ 2, 3, 4 ]
Array after splice: [
  0, -1, 'a', 'b', 'c',
  5, 6, 8, 9, 10
]
For-Loop Iteration:
0
```

Ln 96, Col 1 Spaces: 4 UTF-8 LF {} JavaScript

Applications: ArraysOne.js - ArraysOne...

File Edit Selection View Go Run Terminal Help

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS

EXPLORER ARRAYSONE

JS ArraysOne.js

```
] For-Loop Iteration:
0
-1
a
b
c
5
6
8
9
10
forEach Iteration:
0
-1
a
b
c
5
6
8
9
10
Squared Values: [
  0, 1, NaN, NaN, NaN,
  25, 36, 64, 81, 100
]
Filtered Array (Even Numbers): [ 0, 6, 8, 10 ]
Sum of Array Elements: 1abc568910
```

Ln 96, Col 1 Spaces: 4 UTF-8 LF {} JavaScript

The screenshot shows a Visual Studio Code interface with a dark theme. The title bar indicates the file is 'ArraysOne.js - ArraysOne...'. The main area displays the following JavaScript code:

```
C
5
6
8
9
10
forEach Iteration:
0
-1
a
b
c
5
6
8
9
10
Squared Values: [
  0, 1, NaN, NaN, NaN,
  25, 36, 64, 81, 100
]
Filtered Array (Even Numbers): [ 0, 6, 8, 10 ]
Sum of Array Elements: -1abc568910
Final State of the Array: [
  0, -1, 'a', 'b', 'c',
  5, 6, 8, 9, 10
]
Validation Complete!
```

The code demonstrates various array operations: creating an array, using `forEach` with a function that includes negative indices and non-numeric values, calculating squared values, filtering even numbers, summing the array elements (which results in a string due to a bug), and finally outputting the final state of the array.

The code initializes, manipulates, and iterates through an array, highlighting properties such as length, constructor, and prototype. Essential array methods—including push, pop, shift, unshift, concat, join, slice, and splice—along with iterator methods such as forEach, map, filter, and reduce are demonstrated. The final state of the array and validation messages confirm the correctness of the operations.

By following these steps, you have successfully demonstrated the practical application of JavaScript array properties, methods, and iterators, enhancing your programming proficiency.