

Lesson 04 Demo 01

Implementing the Bubble Sort Algorithm

Objective: To sort an array using bubble sort in JavaScript, learning basic sorting logic and complexity for handling near-sorted data

Tools required: Visual Studio Code and Node.js

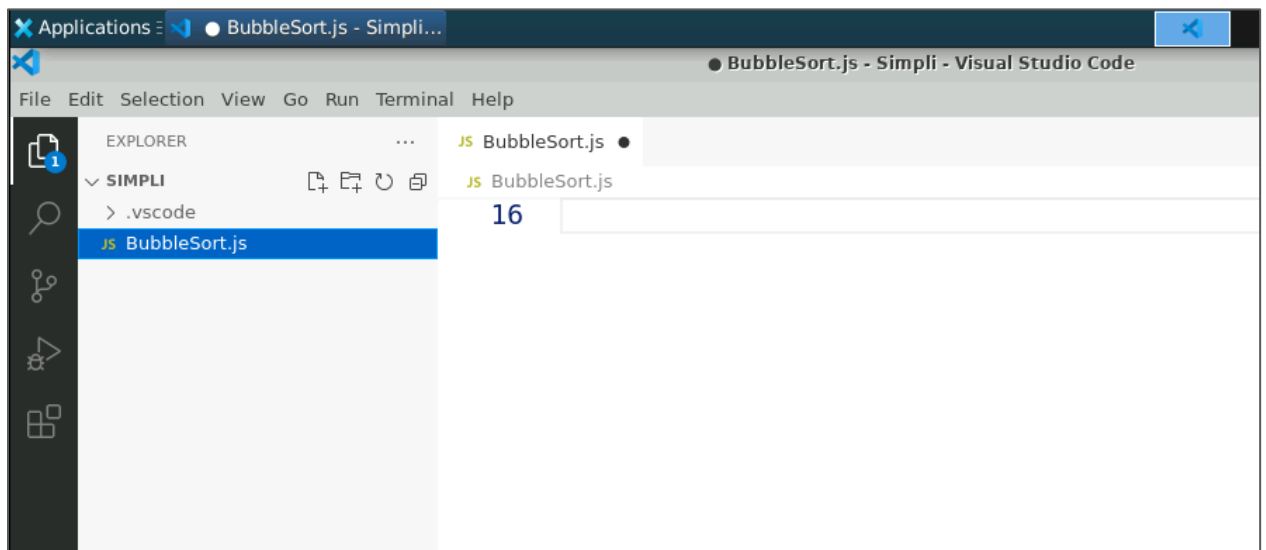
Prerequisites: A basic understanding of arrays and loops in JavaScript

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 **BubbleSort.js**



1.2 Add the following code to the BubbleSort.js file:

```
// Function to perform Bubble Sort on an array
function bubbleSort(array) {
  // Time Complexity:  $O(n^2)$  in the worst case
  // Space Complexity:  $O(1)$ 
  for (let i = 0; i < array.length - 1; i++) {
    for (let j = 0; j < array.length - 1; j++) {
      if (array[j] > array[j + 1]) {
        [array[j], array[j + 1]] = [array[j + 1], array[j]];
      }
    }
  }
  return array;
}

// Example unsorted array
const unsortedArray = [5, 2, 4, 1, 3];

// Measure the execution time using console.time and console.timeEnd
console.time('bubbleSort');
const sortedArray = bubbleSort(unsortedArray);
console.timeEnd('bubbleSort');

console.log('Sorted Array:', sortedArray);
```

JS BubbleSort.js •

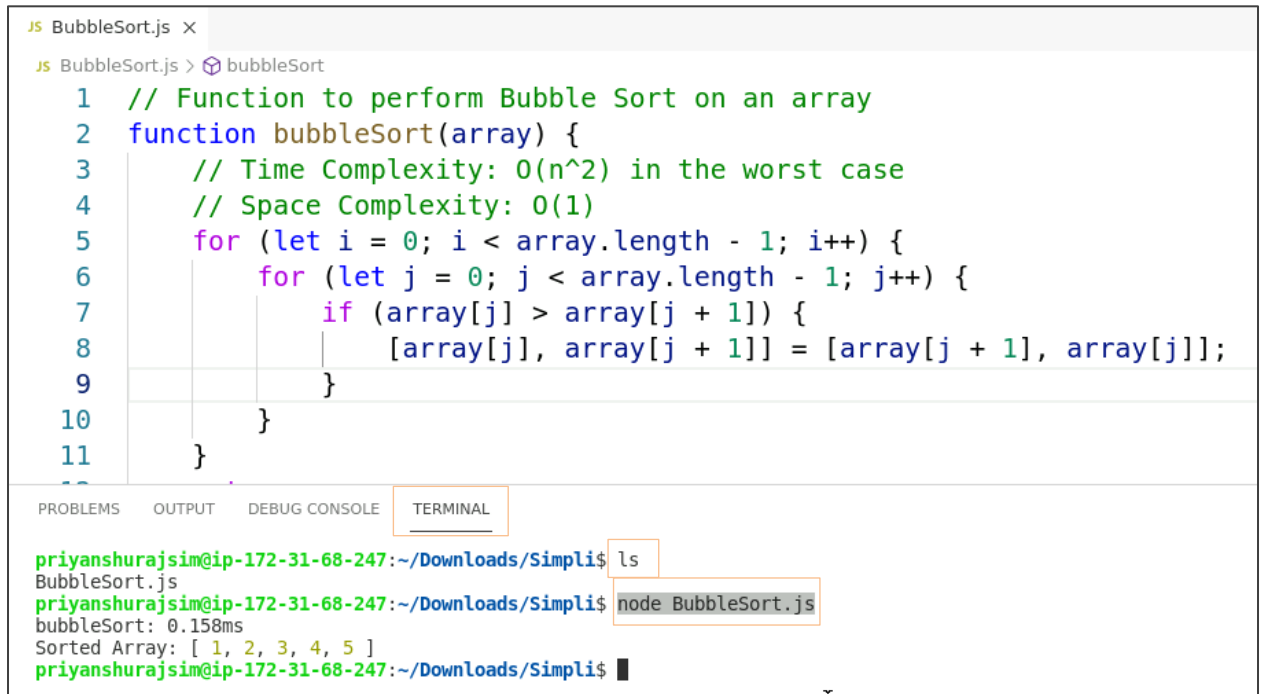
JS BubbleSort.js > ...

```
1 // Function to perform Bubble Sort on an array
2 function bubbleSort(array) {
3     // Time Complexity:  $O(n^2)$  in the worst case
4     // Space Complexity:  $O(1)$ 
5     for (let i = 0; i < array.length - 1; i++) {
6         for (let j = 0; j < array.length - 1; j++) {
7             if (array[j] > array[j + 1]) {
8                 [array[j], array[j + 1]] = [array[j + 1], array[j]];
9             }
10        }
11    }
12    return array;
13 }
14
15 // Example unsorted array
16 const unsortedArray = [5, 2, 4, 1, 3];
17
18 // Measure the execution time using console.time and console.timeEnd
19 console.time('bubbleSort');
20 const sortedArray = bubbleSort(unsortedArray);
21 console.timeEnd('bubbleSort');
22
23 console.log('Sorted Array:', sortedArray);
```

1.3 Press **Ctrl + S** to save the file and then execute it in the **TERMINAL** using the following commands:

ls

node BubbleSort.js



The screenshot shows a code editor with a file named `BubbleSort.js`. The code defines a `bubbleSort` function that sorts an array in ascending order using bubble sort. The function includes comments for time complexity ($O(n^2)$) and space complexity ($O(1)$). Below the code, the terminal window shows the execution of the file. The user runs `ls` and `node BubbleSort.js`, which outputs the execution time and the sorted array.

```
JS BubbleSort.js x
JS BubbleSort.js > bubbleSort
1 // Function to perform Bubble Sort on an array
2 function bubbleSort(array) {
3     // Time Complexity:  $O(n^2)$  in the worst case
4     // Space Complexity:  $O(1)$ 
5     for (let i = 0; i < array.length - 1; i++) {
6         for (let j = 0; j < array.length - 1; j++) {
7             if (array[j] > array[j + 1]) {
8                 [array[j], array[j + 1]] = [array[j + 1], array[j]];
9             }
10        }
11    }
12 }

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL
priyanshurajsim@ip-172-31-68-247:~/Downloads/Simpli$ ls
BubbleSort.js
priyanshurajsim@ip-172-31-68-247:~/Downloads/Simpli$ node BubbleSort.js
bubbleSort: 0.158ms
Sorted Array: [ 1, 2, 3, 4, 5 ]
priyanshurajsim@ip-172-31-68-247:~/Downloads/Simpli$
```

By following these steps, you have successfully used bubble sort in JavaScript to sort an array, understood how it works through adjacent swaps, and explored its use in near-sorted data scenarios with $O(n^2)$ time and $O(1)$ space complexity.