

## Lesson 04 Demo 03

### Implementing Insertion Sort Algorithm

**Objective:** To demonstrate the insertion sort algorithm and explain its time and space complexity using JavaScript

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

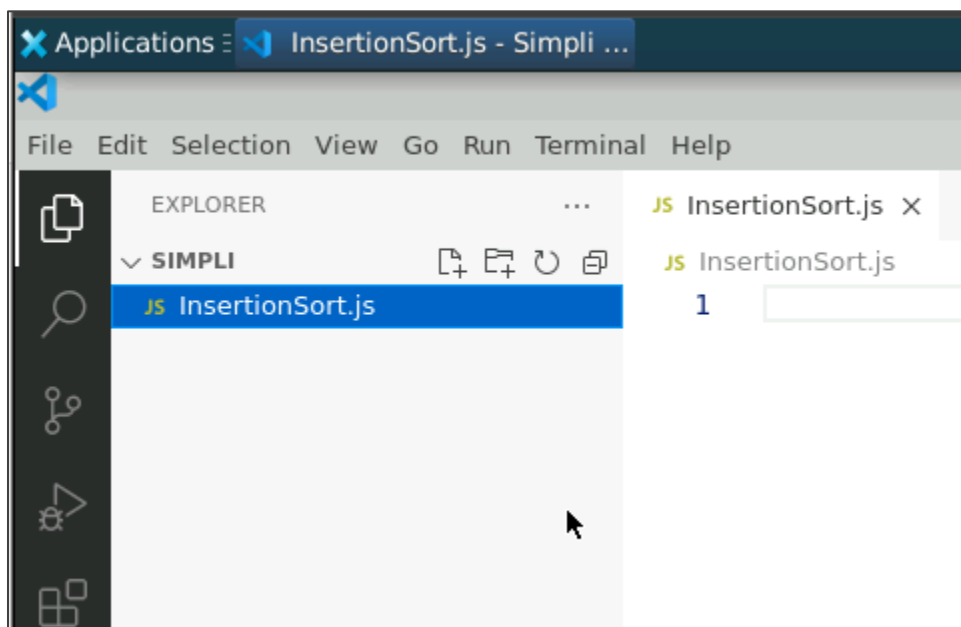
**Prerequisites:** Basic understanding of arrays and loops in JavaScript

Steps to be followed:

1. Create and execute the JS file

#### Step 1: Create and execute the JS file

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



1.2 Write the code given below in the **InsertionSort.js** file:

```
function insertionSort(array) {  
  // Time Complexity:  $O(n^2)$  - where n is the number of elements in the array  
  // Space Complexity:  $O(1)$  - constant space is used (no additional data structures)  
  
  // Start measuring execution time  
  console.time('insertionSort');  
  
  for (let i = 1; i < array.length; i++) {  
    let currentValue = array[i];  
    let j = i - 1;  
  
    // Shift elements greater than the current value to the right  
    while (j >= 0 && array[j] > currentValue) {  
      array[j + 1] = array[j];  
      j--;  
    }  
  
    // Insert the current value at the correct position  
    array[j + 1] = currentValue;  
  }  
  
  // End measuring execution time  
  console.timeEnd('insertionSort');  
  
  return array;  
}  
  
// Example usage:  
const unsortedArray = [5, 2, 4, 1, 3];  
const sortedArray = insertionSort(unsortedArray);  
console.log(sortedArray); // Output: [1, 2, 3, 4, 5]
```

```

1 function insertionSort(array) {
2     // Time Complexity: O(n^2) - where n is the number of elements in the array
3     // Space Complexity: O(1) - constant space is used (no additional data structures)
4
5     // Start measuring execution time
6     console.time('insertionSort');
7
8     for (let i = 1; i < array.length; i++) {
9         let currentValue = array[i];
10        let j = i - 1;
11
12        // Shift elements greater than the current value to the right
13        while (j >= 0 && array[j] > currentValue) {
14            array[j + 1] = array[j];
15            j--;
16        }
17
18        // Insert the current value at the correct position
19        array[j + 1] = currentValue;
20    }
21
22    // End measuring execution time
23    console.timeEnd('insertionSort');
24
25    return array;
26 }
27
28 // Example usage:
29 const unsortedArray = [5, 2, 4, 1, 3];
30 const sortedArray = insertionSort(unsortedArray);
31 console.log(sortedArray); // Output: [1, 2, 3, 4, 5]

```

1.3 Save the file and execute it in the terminal using the following command:

**node InsertionSort.js**

```

8         for (let i = 1; i < array.length; i++) {
9             let currentValue = array[i];
10            let j = i - 1;
11
12            // Shift elements greater than the current value to the right
13            while (j >= 0 && array[j] > currentValue) {
14                array[j + 1] = array[j];
15                j--;
16            }

```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL bash

```

priyanshurajsim@ip-172-31-40-74:~/Downloads/Simpli$ ls
InsertionSort.js
priyanshurajsim@ip-172-31-40-74:~/Downloads/Simpli$ node InsertionSort.js
insertionSort: 0.062ms
[ 1, 2, 3, 4, 5 ]
priyanshurajsim@ip-172-31-40-74:~/Downloads/Simpli$

```

In our example, we used the insertion sort algorithm in JavaScript to arrange the items in an array. It has a time complexity of  $O(n^2)$  and a space complexity of  $O(1)$ .

By following these steps, you have successfully implemented and executed the insertion sort algorithm in JavaScript, including measuring its execution time.