.

Lesson 04 Demo 03

Implementing the Insertion Sort Algorithm

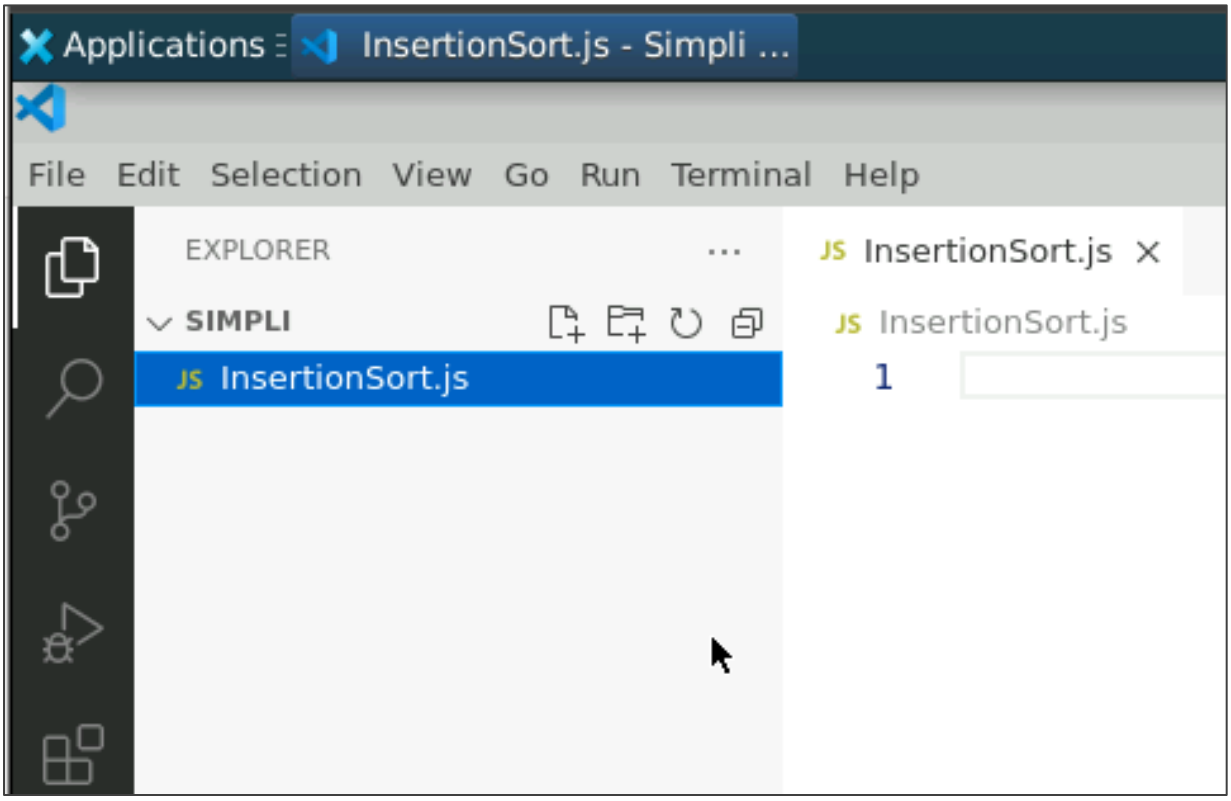
Objective: To sort data using insertion sort in JavaScript for real-time organization of inputs
Tools required: Visual Studio Code and Node.js
Prerequisites: A basic understanding of arrays and loops in JavaScript

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 **InsertionSort.js**



.

1.2 Add the following code to the 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);
```

.

```
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;
  }

```

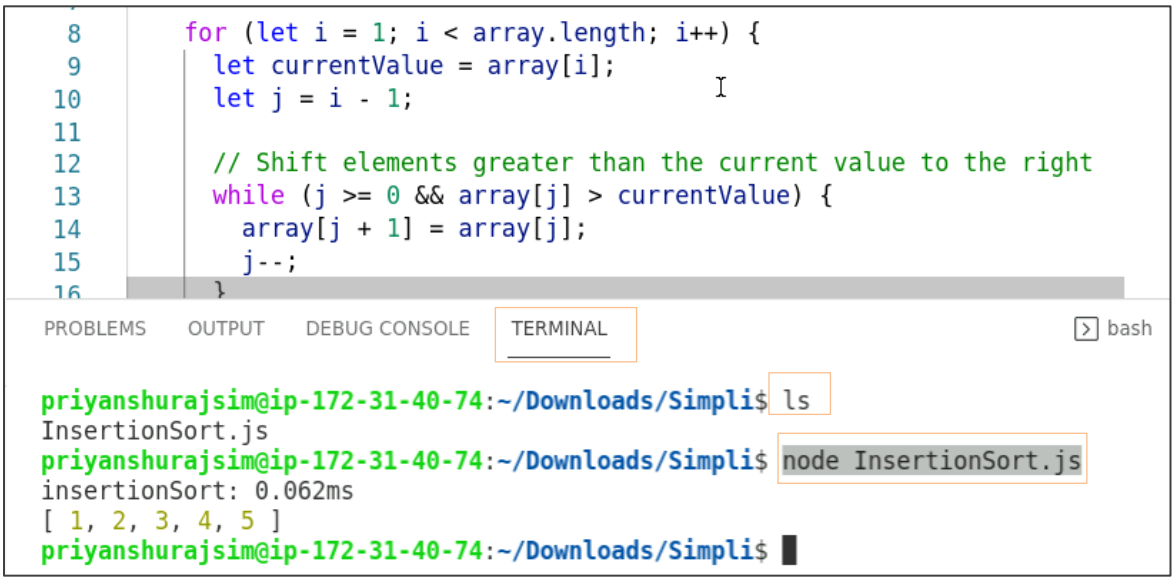
```

// Measure the execution time of the selectionSort function
console.time("selectionSort");
const sortedArray = selectionSort(unsortedArray);
console.timeEnd("selectionSort");

console.log(sortedArray);

```

.

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

ls

node InsertionSort.js



By following these steps, you have successfully used the insertion sort algorithm in JavaScript to organize inputs like live scores or card sequences by placing each element in its correct position, and learned that it has a time complexity of $O(n^2)$ and space complexity of $O(1)$.