Name of Student: Naveen Gummella			
Roll Number: A-16		**LAB Assignment Number:** 3	
Title of LAB Assignment: Create an application to demonstrate various Node.js Events in event emitter class Create functions to sort, reverse and search for an element in an array. Register and trigger these functions using events.			
DOP: 09/09/2023		**DOS:** 26/09/2023	
CO Mapped: CO1	**PO Mapped:** PO3,PO5,PSO1, PSO2	**Signature:**	**Marks:**

Aim:

Create an application to demonstrate various Node.js Events in event emitter class
Create functions to sort, reverse and search for an element in an array. Register and trigger these functions using events.

Description:

In Node.js, you can perform various operations on arrays such as creating, sorting, reversing, and searching for elements. Here are some theoretical explanations and examples for each of these operations:

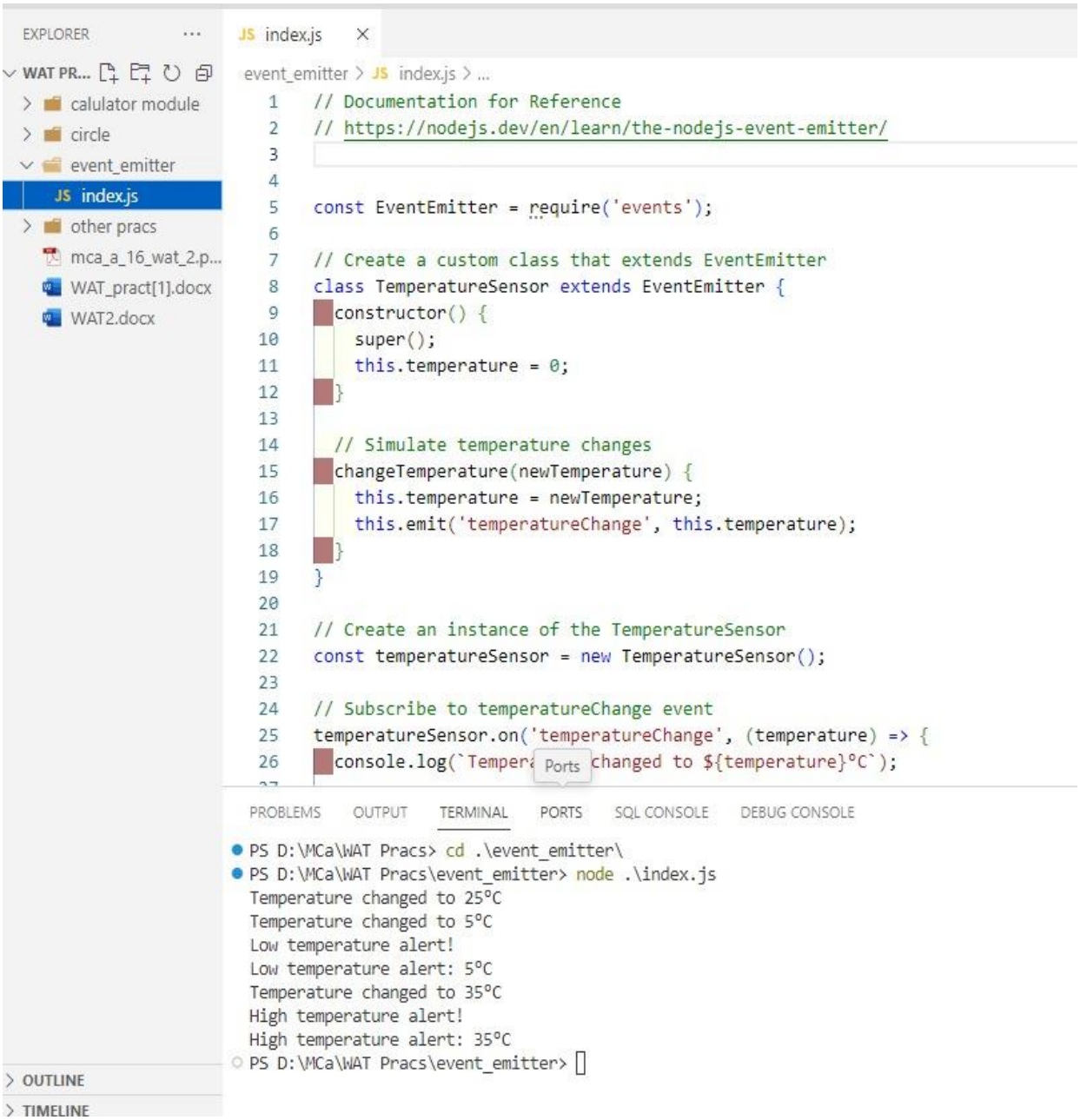
1. Creating an Array: In Node.js, you can create an array by simply declaring it using square brackets. Arrays can hold elements of different data types. Creating an array itself doesn't involve events, as it's a synchronous operation. However, if you're working in a real-world application, you might trigger events when data is fetched or received from an external source (e.g., a database or an API) and populate an array with that data.

2. Sorting an Array: You can sort an array in Node.js using the sort() method. By default, this method sorts elements alphabetically for strings and numerically for numbers. Sorting an array is typically a synchronous operation, but you can use events to notify other parts of your application when sorting is complete or if any specific sorting criteria are met.

3. Reversing an Array: To reverse the order of elements in an array, you can use the reverse() method. Reversing an array is also a synchronous operation, but you can use events to signal when the reversal has been performed.

4. Searching for Elements in an Array: You can search for elements in an array using various methods like indexOf(), find(), and includes(). indexOf() returns the index of the first occurrence of an element in the array or -1 if not found. find() returns the first element in the array that satisfies a provided testing function. includes() checks if an element is present in the array and returns a boolean value. Searching for elements in an array can be synchronous or asynchronous, depending on the data source. Events can be useful to notify when a search operation has been completed or when a specific element is found.

Code & Output:


EXPLORER
index.js
WAT PR...
calulator module
circle
event_emitter
index.js
other pracs
mca_a_16_wat_2.p...
WAT_pract[1].docx
WAT2.docx
event_emitter > index.js > ...
// Documentation for Reference
// https://nodejs.dev/en/learn/the-nodejs-event-emitter/
const EventEmitter = require('events');
// Create a custom class that extends EventEmitter
class TemperatureSensor extends EventEmitter {
constructor() {
super();
this.temperature = 0;
}
// Simulate temperature changes
changeTemperature(newTemperature) {
this.temperature = newTemperature;
this.emit('temperatureChange', this.temperature);
}
}
// Create an instance of the TemperatureSensor
const temperatureSensor = new TemperatureSensor();
// Subscribe to temperatureChange event
temperatureSensor.on('temperatureChange', (temperature) => {
console.log(`Tempera changed to ${temperature}°C`);
Ports
PROBLEMS
OUTPUT
TERMINAL
PORTS
SQL CONSOLE
DEBUG CONSOLE
PS D:\MCa\WAT Pracs> cd .\event_emitter\
PS D:\MCa\WAT Pracs\event_emitter> node .\index.js
Temperature changed to 25°C
Temperature changed to 5°C
Low temperature alert!
Low temperature alert: 5°C
Temperature changed to 35°C
High temperature alert!
High temperature alert: 35°C
PS D:\MCa\WAT Pracs\event_emitter>
OUTLINE
TIMELINE

JS index.js JS main.js ×

event_emitter > JS main.js > ...

```
1
2   // Create functions to sort, reverse and search for an element in an array.
3   //  Register and trigger these functions using events.
4
5   const EventEmitter = require('events');
6
7   // Create an instance of EventEmitter
8   const emitter = new EventEmitter();
9
10  // Sample array
11  const myArray = [3, 1, 4, 1, 5, 9, 2, 6, 5, 3, 5];
12
13  // Function to add an element to the array
14  function addElement(element) {
15      myArray.push(element);
16      emitter.emit('elementAdded', element);
17  }
18
19  // Function to sort the array
20  function sortArray() {
21      myArray.sort((a, b) => a - b);
22      emitter.emit('arraySorted', myArray);
23  }
24
25  // Function to reverse the array
26  function reverseArray() {
```

PROBLEMS OUTPUT TERMINAL PORTS SQL CONSOLE DEBUG CONSOLE

```
  4, 5, 5, 5, 6, 7,
  9
]
Array reversed: [
  9, 7, 6, 5, 5, 5,
  4, 3, 3, 2, 1, 1,
  0
]
Element 5 found at index 3
Element 8 not found in the array
PS D:\MCa\WAT Pracs\event_emitter>
```

Conclusion:

The use of events in Node.js is often associated with handling asynchronous operations like file I/O, network requests, or event-driven architectures, where you can emit events to signal the completion or occurrence of specific tasks within your application.
These are some basic operations you can perform on arrays in Node.js. Depending on your specific use case and data, you may need to use additional methods and customize these operations further.