Name of Student: Samyak s Lamsoge		
Roll Number: 29		**LAB Assignment Number: 3**
Title of LAB Assignment: Create an application to demonstrate Node.js Event Emitter		
DOP: 25-09-2023		**DOS 26-09-2023**
CO Mapped: **CO1**	**PO Mapped:** **PO3, PO5, PSO1, PSO2**	**Signature:**

Practical No: 3

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:

Understanding the Node.js EventEmitter Module

Event emitters are objects in Node.js that trigger an event by sending a message to signal that an action was completed. JavaScript developers can write code that listens to events from an event emitter, allowing them to execute functions every time those events are triggered. In this context, events are composed of an identifying string and any data that needs to be passed to the listeners

Key Components:

1. EventEmitter Class:

- The core of the EventEmitter module is the EventEmitter class. It provides methods for emitting events and registering event listeners.
- Developers can create instances of this class to manage custom events within their applications.

2.Event:

- An event is a signal or notification that something specific has happened. It is identified by a unique name or identifier.
- Events can be emitted (triggered) by an EventEmitter instance when a particular action or condition occurs.

3. Event Listener:

- An event listener is a function that is registered to respond to a specific event when it is emitted.
- Event listeners are responsible for handling events by executing custom code in response to the event.

Example:

Let's illustrate the usage of the EventEmitter module with a simple example:

```
const EventEmitter = require('events');

const myEmitter = new EventEmitter();

// Register an event listener

myEmitter.on('greet', (name) => {

console.log(`Hello, ${name}!`);

});

// Emit the 'great' event

myEmitter.emit('greet', 'John');
```

In this example, we:

- Import the `events` module and create an instance of `EventEmitter`.
- Register an event listener for the 'greet' event.
- Emit the 'greet' event with a 'name' argument, triggering the event listener to execute and print a greeting.
- This simple example demonstrates the EventEmitter's role in event-driven programming, where events are emitted and listeners respond to them.

Practical Applications:

1. **HTTP Servers:** EventEmitter is extensively used in building Node.js HTTP servers. HTTP requests, such as 'request' and 'response' events, are handled using EventEmitter to make servers highly responsive.

2. **File I/O:** Reading and writing files asynchronously often involve EventEmitter. Events like 'data' and 'end' are emitted during file streams to handle data chunks and stream completion.

3. **Custom Applications:** Developers can create custom events and listeners for various scenarios within their applications. For instance, handling user

interactions, real-time updates, or custom triggers.

(1) Create an application to demonstrate various Node.js Events in event emitter class.

Code:

Thermostat.js

```
const EventEmitter = require('events');

class Thermostat extends EventEmitter {
 constructor() {
  super();
  this.temperature = 0;
 }

 increaseTemperature() {
  this.temperature++;
  this.emit('temperatureChange', this.temperature);
 }

 decreaseTemperature() {
  this.temperature--;
  this.emit('temperatureChange', this.temperature);
 }

 overheating() {
  this.emit('overheat', this.temperature);
 }

 cooling() {
  this.emit('cool', this.temperature);
 }
}

module.exports = Thermostat;
```

main.js

```
const Thermostat = require('./thermostat');

const thermostat = new Thermostat();

// Listen to the 'temperatureChange' event
thermostat.on('temperatureChange', (temperature) => {
console.log(`Temperature changed to ${temperature}°C`);
});
```

```
// Listen to the 'overheat' event
thermostat.on('overheat', (temperature) => {
 console.log(`Warning: Overheating at ${temperature}°C!`);
});

// Listen to the 'cool' event
thermostat.on('cool', (temperature) => {
 console.log(`Cooling initiated at ${temperature}°C`);
});

// Simulate temperature changes
thermostat.increaseTemperature();
thermostat.increaseTemperature();
thermostat.decreaseTemperature();
thermostat.overheating();
thermostat.cooling();
```

Output:

```
Temperature changed to 1°C
Temperature changed to 2°C
Temperature changed to 1°C
Warning: Overheating at 1°C!
Cooling initiated at 1°C
```

2. Create functions to sort, reverse and search for an element in an array. Register and trigger these functions using events.

Code:

arrayOperations.js

```
const EventEmitter = require('events');

class ArrayOperations extends EventEmitter {
 constructor() {
```

```
    super();
  }

  // Function to sort an array
  sortArray(arr) {
    const sortedArray = arr.slice().sort((a, b) => a - b);
    this.emit('sorted', sortedArray);
  }

  // Function to reverse an array
  reverseArray(arr) {
    const reversedArray = arr.slice().reverse();
    this.emit('reversed', reversedArray);
  }

  // Function to search for an element in an array
  searchArray(arr, target) {
    const index = arr.indexOf(target);
    this.emit('searched', { target, index });
  }
}
module.exports = ArrayOperations;
```

main_array.js

```
const ArrayOperations = require('./arrayOperations');

const arrayOps = new ArrayOperations();

// Register event listeners
arrayOps.on('sorted', (sortedArray) => {
console.log('Sorted Array:', sortedArray);
});

arrayOps.on('reversed', (reversedArray) => {
  console.log('Reversed Array:', reversedArray);
});

arrayOps.on('searched', ({ target, index }) => {
  if (index !== -1) {
    console.log(`${target} found at index ${index}`);
  } else {
    console.log(`${target} not found in the array`);
```

```
 }
});

// Sample array
const myArray = [5, 2, 8, 1, 9, 4, 3, 7, 6];

// Trigger the functions using events
arrayOps.sortArray(myArray);
arrayOps.reverseArray(myArray);
arrayOps.searchArray(myArray, 4);
arrayOps.searchArray(myArray, 10);
```

Output:

```
Sorted Array: [
  1, 2, 3, 4, 5,
  6, 7, 8, 9
]
Reversed Array: [
  6, 7, 3, 4, 9,
  1, 8, 2, 5
]
4 found at index 5
10 not found in the array
```

Conclusion:

In conclusion, the Node.js EventEmitter module is a cornerstone of Node.js development, enabling event-driven programming in both core modules and custom applications. It serves as a powerful communication mechanism, allowing different parts of an application to interact efficiently by emitting and responding to events.

The EventEmitter module is vital for building scalable and responsive applications that can handle numerous simultaneous events and asynchronous operations. Its use cases extend from HTTP servers and file I/O to custom application scenarios, making it a fundamental skill for Node.js developers.

By understanding and effectively utilizing EventEmitter, developers can harness the full potential of Node.js's event-driven architecture, resulting in robust, modular, and highly responsive applications that meet the demands of modern web and server development.