Unit – 4 Node JS

Node Event Emitters

- The core of NodeJS is event-driven programming.
- In NodeJS, we achieve event-driven programming with the event-emitter class.
- Node.js allows us to create and handle custom events easily by using events module.
- Event module includes EventEmitter class which can be used to raise and handle custom events.
- With an event emitter, we can simply raise a new event from a different part of an application, and a listener will listen to the raised event and have some action performed for the event.

Method & Description
addListener(event, listener) Adds a listener at the end of the listeners array for the specified event. Multiple calls passing the same combination of event and listener will result in the listener being added multiple times..
on(event, listener) Adds a listener at the end of the listeners array for the specified event. Multiple calls passing the same combination of event and listener will result in the listener being added multiple times. It is alias of addListner
once(event, listener) Adds a **one time listener** to the event. This listener is invoked only the next time the event is fired, after which it is removed. Returns emitter, so calls can be chained.
removeListener(event, listener) Removes a listener from the listener array for the specified event. removeListener will remove, at most, one instance of a listener from the listener array. If any single listener has been added multiple times to the listener array for the specified event, then removeListener must be called multiple times to remove each instance.
removeAllListeners([event]) Removes all listeners, or those of the specified event. It's not a good idea to remove listeners that were added elsewhere in the code, especially when it's on an emitter that you didn't create (e.g. sockets or file streams).
setMaxListeners(n) By default, EventEmitters will print a warning if more than 10 listeners are added for a particular event. This is a useful default which helps finding memory leaks. This function allows that to be increased. Set to zero for unlimited.
listeners(event) Returns an array of listeners for the specified event.
emit(event, [arg1], [arg2], [...]) Execute each of the listeners in order with the supplied arguments. Returns true if the event had listeners, false otherwise.

event handling

```
var events = require('events');
var em = new events.EventEmitter();

// 1st way to handle/listen to the event
em.on('SwitchOn', (data) => {
    console.log('SwitchOn event triggered ' + data);
});

//2nd way to handle/listen to the event
em.addListener('SwitchOff', function (data) {
    console.log('SwitchOff  event triggered ' + data);
});

// publish an event
em.emit('SwitchOn','Fan');
em.emit('SwitchOn','Light');
em.emit('SwitchOff','Fan');
em.emit('SwitchOff','Light');
```

event handling 2

```
var events = require('events');
var em = new events.EventEmitter();

// 3RD way to handle/listen to the event
em.on('SwitchOn',function(data){
    console.log('SwitchOn event triggered ' + data);
});

//4TH way to handle/listen to the event
em.addListener('SwitchOff',function(data){
    console.log('SwitchOff  event triggered ' + data);
});

// publish an event
em.emit('SwitchOn','AC');
em.emit('SwitchOn','LAPTOP');
em.emit('SwitchOff','AC');
em.emit('SwitchOff','LAPTOP');
```

event handling 3

```
var events = require('events');
var em = new events.EventEmitter();

var SwitchOn = function(data){
    console.log('SwitchOn event triggered ' + data);
}
var SwitchOff = function(data){
    console.log('SwitchOff  event triggered ' + data);
}

//register for event
em.on('SwitchOnEvent',SwitchOn);
em.addListener('SwitchOffEvent',SwitchOff);

// publish an event
em.emit('SwitchOnEvent','AC');
em.emit('SwitchOnEvent','LAPTOP');
em.removeListener('SwitchOnEvent',SwitchOn);

em.emit('SwitchOnEvent','TV'); //event will not trigger due to listner is removed
em.emit('SwitchOffEvent','AC');
em.emit('SwitchOffEvent','LAPTOP');

em.removeListener('SwitchOffEvent',SwitchOff);
em.emit('SwitchOff','TV'); //event will not trigger due to listner is removed
```

```
var events = require('events');
var em = new events.EventEmitter();

var SwitchOn = function(data){
    console.log('SwitchOn event triggered ' + data);
}
var SwitchOff = function(data){
    console.log('SwitchOff  event triggered ' + data);
}

//register for event
//Events listened with once() will be triggered only once.
em.once('SwitchOnEvent',SwitchOn);
em.once('SwitchOffEvent',SwitchOff);

// publish an event
em.emit('SwitchOnEvent','AC');
em.emit('SwitchOnEvent','LAPTOP'); //will be ignored
em.emit('SwitchOnEvent','TV');  //will be ignored
em.emit('SwitchOffEvent','AC');
em.emit('SwitchOffEvent','LAPTOP'); //will be ignored
em.emit('SwitchOff','TV');  //will be ignored
```

Is an Event Emitter Synchronous or Asynchronous?

- The events raised by event emitters are synchronously executed by the listeners in the current event loop's iteration.

Order of Execution of the Listeners

- The listeners are executed in the order the listeners are created for an event emitter.

When should I use EventEmitter?

- The EventEmitter should be used when the same event can occur multiple times, or may not occur at all. A callback, in fact, is expected to be invoked exactly once, whether the operation is successful or not. Callback means call me when you are ready
- An API that uses callbacks can notify only one particular callback while using an EventEmitter allows us to register multiple listeners for the same event.
- Use event emitter if you need to notify the user of a state change.