**Events Module**

* In Node.js, an event emitter is like a messenger that helps different parts of a program communicate with each other. Imagine it as a broadcaster at a big event, letting everyone know what's happening.
* **Messenger** Role: Just like a messenger, an event emitter sends messages, or "events," to whoever is interested in hearing them. These events could be anything – a button click, a file being loaded, or a connection being established.
* **Listeners**: There are listeners, or the audience, who are interested in specific types of events. These listeners are set up to pay attention to certain messages. When the event emitter sends out an event, the relevant listeners respond to it.
* **Event** Types: Events have types, like categories. For instance, a 'click' event or a 'data received' event. When you're setting up an event emitter, you define these types, so others know what to expect.
* **Example**: Think of a light switch. The switch itself is like the event emitter. When you turn it on or off, it emits an event – the change in state. Now, imagine a smart home system with different devices, like lights and speakers. Each of these devices is a listener. They're interested in the 'state change' event. So, when the switch is toggled, these devices react accordingly.
* If you worked with JavaScript in the browser, you know how much of the interaction of the user is handled through events: mouse clicks, keyboard button presses, reacting to mouse movements, and so on.
* On the backend side, Node.js offers us the option to build a similar system using the [events module](https://nodejs.dev/en/api/events/).

**You initialize that using**

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

**const ee = new EventEmitter();**

* **emit** is used to trigger an event
* **on/addListener** is used to add a callback function that's going to be executed when the event is triggered

**Syntax:**

**eventEmitter.emit(event, [arg1], [arg2], [...])**

**eventEmitter.on(event, listener)**

**eventEmitter.addListener(event, listener)**

* **eventEmmitter.on(event, listener)** and **eventEmitter.addListener(event, listener)** are pretty much similar.
* It adds the listener at the end of the listener’s array for the specified event. Multiple calls to the same event and listener will add the listener multiple times and correspondingly fire multiple times.
* Both functions return emitter, so calls can be chained.

In code, using an event emitter in Node.js looks something like this:

const **eventemitter** = require('events');

const **ee** = new **eventemitter**();

**ee**.**on**(**'start'**, () => {

console.log('started');

});

**ee**.**emit**(**'start'**);

In this example, when the ‘start’ event is emitted, the listener responds by printing 'Started' to the console.

So, in Node.js, an event emitter is a handy tool for making different parts of a program talk to each other by sending and receiving messages. It's a way to create responsive and interactive applications.

**Example with arguments**

const **eventemitter** = require('events');

const ee = new **eventemitter**();

ee.**on**(**'start'**, **(start, end)** => {

    console.log(`started from ${start} to ${end}`);

  });

  ee.**emit**(**'start'**, **1, 100**);

**Output: started from 1 to 100**

**Example:**

var **eventemitter** =require("events");

var ee=new **eventemitter**();

ee.on("connection",function(){

    console.log("Connection successfully");

    ee.emit("data-received");

});

ee.on("data-received",function()

{

    console.log("data received successfully");

})

ee.emit("connection");

console.log("thanks");

**Output:**

**Connection successfully**

**data received successfully**

**thanks**

**Removing Listener:**

The **eventEmitter.removeListener()** takes two argument event and listener, and removes that listener from the listeners array that is subscribed to that event. While **eventEmitter.removeAllListeners()** removes all the listener from the array which are subscribed to the mentioned event.

**Syntax:**

**eventEmitter.removeListener(event, listener)**

**eventEmitter.removeAllListeners([event])**

**Note:**

* Removing the listener from the array will change the sequence of the listener’s array, hence it must be carefully used.
* The **eventEmitter.removeListener()** will remove at most one instance of the listener which is in front of the queue.

**Examples**

**//Importing events**

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

**// Initializing event emitter instances**

**var eventEmitter = new EventEmitter();**

**var fun1 = (msg) => {**

**console.log("Message from fun1: " + msg);**

**};**

**var fun2 = (msg) => {**

**console.log("Message from fun2: " + msg);**

**};**

**// Registering fun1, fun2**

**eventEmitter.on('myEvent1', fun1); //executes fun1 function on event myEvent "Line A"**

**eventEmitter.on('myEvent2', fun2); //executes fun2 function on event myEvent "Line B"**

**eventEmitter.on('myEvent1', fun1); //executes fun1 function on event myEvent "Line C"**

**eventEmitter.on('myEvent2', fun2); //executes fun2 function on event myEvent2 " Line D"**

**// Removing a listener of myEvent2**

**eventEmitter.removeListener('myEvent2', fun2); //remove fun2 "line B"**

**// Removing all the listeners of myEvent1**

**eventEmitter.removeAllListeners('myEvent1'); // This will remove all events named "myEvent1" Means (Line A) and (Line C)**

**// Triggering myEvent2**

**eventEmitter.emit('myEvent2', "LJ University"); //Executes a function fun2 (Line D)**

**eventEmitter.emit('myEvent1', "LJU"); // Nothing will be displayed as all events have been removed**

**Output:**

**Message from fun2: LJ University**

**Write node js script to create two listeners for a common event. Print number of events associated with an emitter. Remove one of the listeners and call remaining listener again. Also print number of remaining listener**

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

**var ee = new EventEmitter();**

**var listener1 = function listen1() {**

**console.log("Listener1 executed");**

**};**

**var listener2 = function listen2() {**

**console.log("Listener2 executed");**

**};**

**ee.on("conn",listener1) // addListener/on both functions perform same task. You can use addListener or on to execute a task.**

**ee.on("conn", listener2);**

**//Count initially all the events.**

**let count = ee.listenerCount("conn");**

**console.log("Count 1: " + count );**

**ee.emit("conn");**

**//To remove 1st listner**

**ee.removeListener('conn', listener1);**

**count = ee.listenerCount("conn");**

**console.log("Counting again: " + count );**

**ee.emit("conn");**

**// Above program ends here as per the question. Below is additional task of remove all listeners. and count the listener.**

**//To remove all listners**

**ee.removeAllListeners('conn', listener1);**

**count = ee.listenerCount("conn");**

**console.log("Again Count afetr removing all listeners: " + count );**

**ee.emit("conn");**

**eventEmitter.listenerCount():** It returns the number of listeners listening to the specified event.

**Write node js script to handle events as asked below.**

1. **Check the radius is negative or not. If negative then display message “Radius” must be positive” else calculate the perimeter of circle.**
2. **Check side is negative or not. If negative then display message “Side must be positive” else calculate the perimeter of square.**

var eventemitter = require("events");

var ee = new eventemitter();

ee.on("negradius", () => {

console.log("Radius must be positive");

});

ee.on("negside", () => {

console.log("Side must be positive");

});

ee.on("findval", (r,s) => {

if (r < 0) {

ee.emit("negradius");

}else{

var rperi = 2 \* 3.14 \* r;

console.log(rperi);

}

if (s < 0)

{

ee.emit("negside");

}

else{

var speri = 4\*s;

console.log(speri);

}

})

ee.emit("findval",10,3);

**Write node js script to handle event of write a data in file, append data in file and then read the file and display data in console.**

var e=require("events");

var fs=require("fs");

var ee=new e();

ee.on("data-write",function()

{

fs.writeFile("b.txt","Hello, ",(err)=> {console.log()});

console.log("Data Written");

ee.emit("data-append");

ee.emit("data-read");

});

ee.on("data-append",function()

{

fs.appendFile("b.txt","Good Morning!",(err)=> {console.log()});

console.log("Data Appended");

});

ee.on("data-read",function()

{

fs.readFile("b.txt","utf-8",(err,data)=>

{

if(err){

console.error(err);

}

console.log(data);

});

});

ee.emit("data-write");

**Write node.js script to create a class named person by assigning name and age in form of members. Create two objects and a method named elder which returns elder person object. Details of elder person should be printed in console as well as in file.**

class person

{

    constructor(name,age)

    {

        this.age=age;

        this.name=name;

    }

    elder(p)

    {

        if(this.age>p.age)

        {

            return this;

        }

        else{

            return p;

        }

    }

}

var p1= new person("xyz",23);

var p2= new person("abc",34);

var p3=p1.elder(p2);

const jsonstr=JSON.stringify(p3);

var ps=require("fs");

ps.writeFileSync("d2.txt",jsonstr);

**Output:**

person { age: 34, name: 'abc' }

**Write node.js script to create a class named time and assign members hour, minute and second. Create two objects of time class and add both the time objects so that it should return the value in third time object. The third time object should have hour , minute and second such that if seconds exceed 60 then minute value should be incremented and if minute exceed 60 then hour value should be incremented. The value should be printed in console as well as in file.**

class time

{

    constructor(hour,min,sec)

    {

        this.hour=hour;

        this.min=min;

        this.sec=sec;

    }

    timer(p)

    {

        var t=new time();

        t.hour=this.hour+p.hour;

        t.min=this.min+p.min;

        t.sec=this.sec+p.sec;

        if(t.sec>60)

        {

            t.sec%=60;

            t.min++;

        }

        if(t.min>60)

        {

            t.min%=60;

            t.hour++;

        }

        return t;

    }

}

var t1= new time(1,50,50);

var t2= new time(2,30,50);

var t3=t1.timer(t2);

console.log(t3);

const jsonstr=JSON.stringify(t3);

var ps=require("fs");

ps.writeFileSync("time.txt",jsonstr);

**Output:**

**time { hour: 4, min: 21, sec: 40 }**