**The Node.js Event emitter**

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/).

This module, in particular, offers the EventEmitter class, which we'll use to handle our events.

You initialize that using

const EventEmitter = require('events');

const eventEmitter = new EventEmitter();

This object exposes, among many others, the on and emit methods.

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

**Example-1:**

const E = require('events');

const ee = new E();

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

console.log('started');

});

ee.emit('start');

**Example-2:**

const E = require('events');

const ee = new E();

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

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

});

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

All EventEmitters emit the event **newListener** when new listeners are added and **removeListener** when existing listeners are removed.

**Listening events:** Before emits any event, it must register functions (callbacks) to listen to the events.

**Syntax:**

eventEmitter.addListener(event, listener)

eventEmitter.on(event, listener)

**eventEmitter.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.

**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])

**Example-3:**

// Importing events

const E = require('events');

// Initializing event emitter instances

var ee = new E();

var Mark1= (msg) => {

console.log("Message from Mark1: " + msg);

};

var Mark2 = (msg) => {

console.log("Message from Mark2: " + msg);

};

// Registering geek1 and geek2

ee.on('myEvent', Mark1);

ee.on('myEvent', Mark1);

ee.on('myEvent', Mark2);

// Removing listener geek1 that was

// registered on the line 13

ee.removeListener('myEvent', Mark1);

// Triggering myEvent

ee.emit('myEvent', "Event occurred");

// Removing all the listeners to myEvent

ee.removeAllListeners('myEvent');

// Triggering myEvent

ee.emit('myEvent', "Event occurred");

**Example-4(Used for sequence of Statements)**

var e=require("events");

var ee=new e();

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");

**Example-5 Sequence of operation with mesz printed :1) File Written 2) File Appended 3) Data to be written on console 4) Thanks for all operations to be done.**

var e=require("events");

var ps=require("fs");

var ee=new e.EventEmitter();

ee.on("connection",function()

{

ps.writeFile("b.txt","This is data",(err)=> {console.log()});

console.log("File Written");

ee.emit("data-received");

ee.emit("data-received1");

});

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

{

ps.appendFile("b.txt","That is data",(err)=> {console.log()});

console.log("File Appended");

});

ee.on("data-received1",function()

{

ps.readFile("b.txt",(err,data)=>

{

console.error();

console.log(data.toString());

console.log("thanks for using my program on console");

});

});

ee.emit("connection");