EVENT EMITTERS

DONE BY :
AAKAR MUTHA
JASON KARTER

Node.js uses events module to create and handle custom events. The EventEmitter class can be used to create and handle custom events module.

SYNTAX

const EventEmitter = require('events');

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.

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.

USING ANGULAR

addListener(event, listener) :

Adds a listener at the end of the listeners array for the specified event. No checks are made to see if the listener has already been added. Multiple calls passing the same combination of event and listener will result in the listener being added multiple times. Returns emitter, so calls can be chained.

1:importing events

2:creating one emitter object em (eventemitter is one class it is present in events module used to create and handle custom events)

3:on method needs an event name ,it can handle and the callback function to call when the event is raised.

4:emit function (it will invoke the event), the first parameter is the event name as a string and the remaining are arguments.

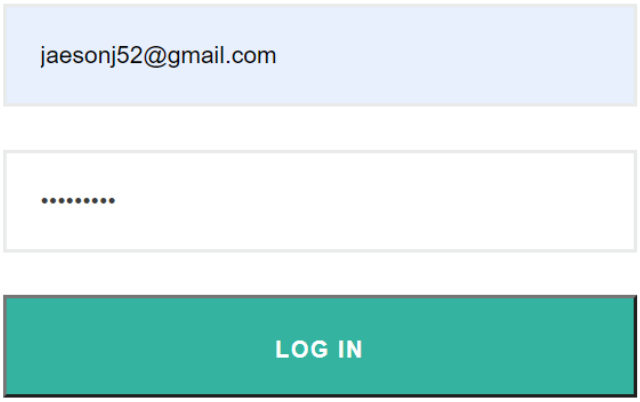
```html
<div class="wrapper">
  <div class="container">
    <div class="login-form">
      <form [formGroup]="myForm" (ngSubmit)="onSubmit(myForm)">
        <input
          type="text"
          placeholder="Email or Username"
          class="input"
          formControlName="name"
        /><br />
        <input
          type="password"
          placeholder="Password"
          class="input"
          formControlName="password"
        /><br />
        <button class="btn" type="submit">log in</button>
      </form>
    </div>
  </div>
</div>
```

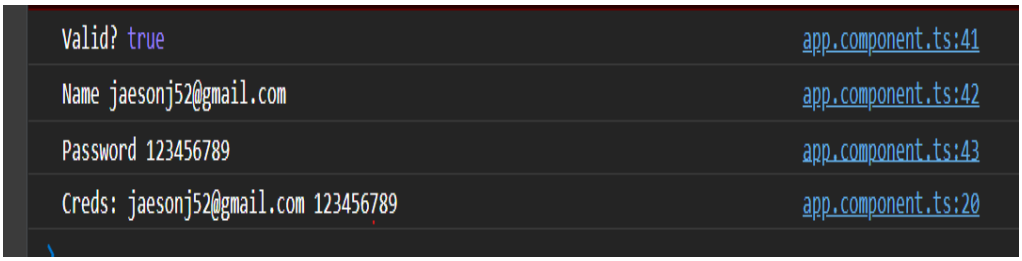
```typescript
import { Component, OnInit } from '@angular/core';
import { FormControl, FormGroup } from '@angular/forms';
declare var require: any;
var events = require('events');

@Component({
  selector: 'app-root',
  templateUrl: './app.component.html',
  styleUrls: ['./app.component.css'],
})
export class AppComponent implements OnInit {
  title = 'nodetask';
  myForm!: FormGroup;
  formData: string = '';
  bellring: string = '';

  eventEmitter = new events.EventEmitter();
```

```
onSubmit(form: FormGroup) {
  console.log('Valid?', form.valid); // true or false
  console.log('Name', form.value.name);
  console.log('Password', form.value.password);
  this.eventEmitter.on('Login', this.myfun);
  this.eventEmitter.emit('Login', form.value.name, form.value.password);
}
```

```
ngOnInit(): void {
  this.myForm = new FormGroup({
    name: new FormControl(''),
    password: new FormControl(''),
  });
}
```

OUTPUT:





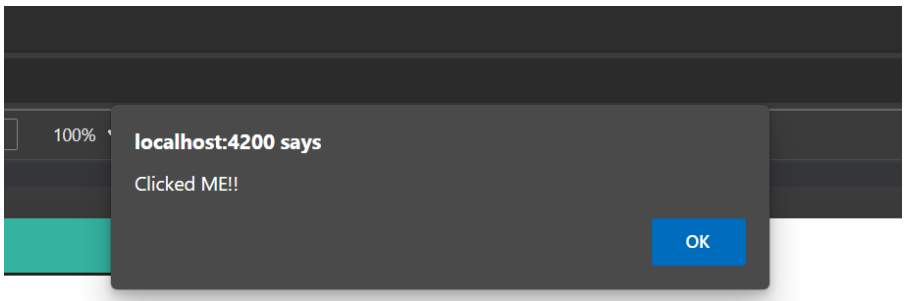
CHAIN OF EVENTS AND DATA PASSING:

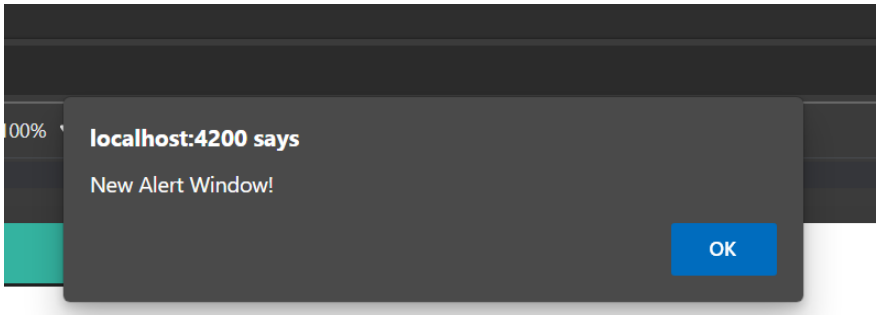
```
<!-- ------------------------------------------------------------------- -->
<div class="wrapper">
  <button class="btn" type="button" (click)="onClick()">Click Me</button>
</div>
<!-- ------------------------------------------------------------------- -->
```

```
myClick = () => {
  alert('Clicked ME!!');
  this.eventEmitter.on('alert', this.anotherAlert);
  this.eventEmitter.emit('alert', 'New Alert Window!');
};

anotherAlert = function (data: string) {
  alert(data);
};
```

```
onClick() {
  this.eventEmitter.on('click', this.myClick);
  this.eventEmitter.emit('click');
}
```

OUTPUT:





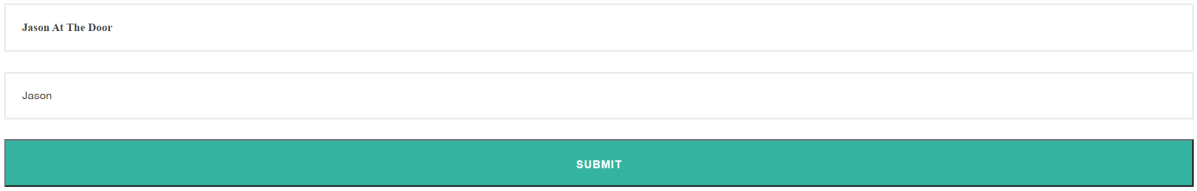
MULTIPLE EVENT LISTENERS:

```
<div class="wrapper">
  <div class="login-form">
    <h3 class="input">{{bellring}}</h3>
    <input
      type="text"
      placeholder="Enter Value"
      class="input"
      [(ngModel)]="formData"
    /><br />
    <button class="btn" type="submit" (click)="onButtonClick()">Submit</button>
  </div>
</div>
<app-fileupload></app-fileupload>
```

```
onButtonClick() {
  this.eventEmitter.addListener('bellRing', this.bellRingHandler1);
  this.eventEmitter.addListener('bellRing', this.bellRingHandler2);
  this.eventEmitter.emit('bellRing', this.formData);

}
```

```
bellRingHandler1 = (who: string) => {
  if (who == 'Jason') {
    this.bellring = 'Jason At The Door';
    return;
  }
};

bellRingHandler2 = () => {
  console.log('\n');
  console.log('The Bell Ringing..... (Handler 2)');
  this.eventEmitter.addListener('nobodyIsAtHome', this.nobodyIsAtHomeHandler);
  this.eventEmitter.emit('nobodyIsAtHome');
};

nobodyIsAtHomeHandler = ()=>  {
  console.log("\n");
  console.log(" Sorry, Nobody is at home now, Please leave your message!")
}
```

OUTPUT:




```
                                                    app.component.ts:61
  The Bell Ringing..... (Handler 2)                 app.component.ts:62
                                                    app.component.ts:68
   Sorry, Nobody is at home now, Please leave your message!  app.component.ts:69
>
```

on(event, listener):

Adds a listener at the end of the listeners array for the specified event. No checks are made to see if the listener has already been added. Multiple calls passing the same combination of event and listener will result in the listener being added multiple times. Returns emitter, so calls can be chained.

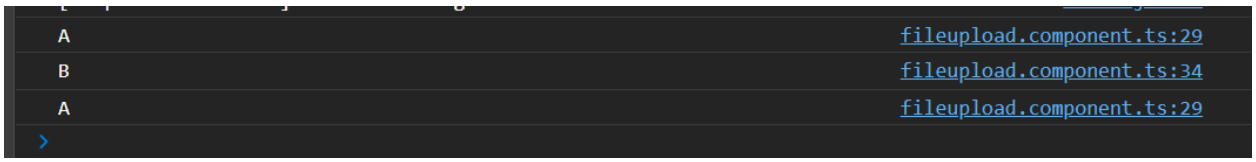
removeListner(event,listener):

Removes a listener from the listener array for the specified event. Caution − It changes the array indices in the listener array behind the listener. 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. Returns emitter, so calls can be chained.

```
<div class="login-form">
  <button class="btn" type="button" (click)="onNumberClick()">Click</button>
</div>
```

```
handler1 = () => {
  console.log('A');
  this.eventEmitter.removeListener('event', this.handler2);
};

handler2 = () => {
  console.log('B');
};

ngOnInit(): void {
  this.myForm = new FormGroup({
    number: new FormControl()
  });
}

onNumberClick() {
  this.eventEmitter.on('event', this.handler1);
  this.eventEmitter.on('event', this.handler2);
  this.eventEmitter.emit('event');
  this.eventEmitter.emit('event');
}
```

OUTPUT:



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.

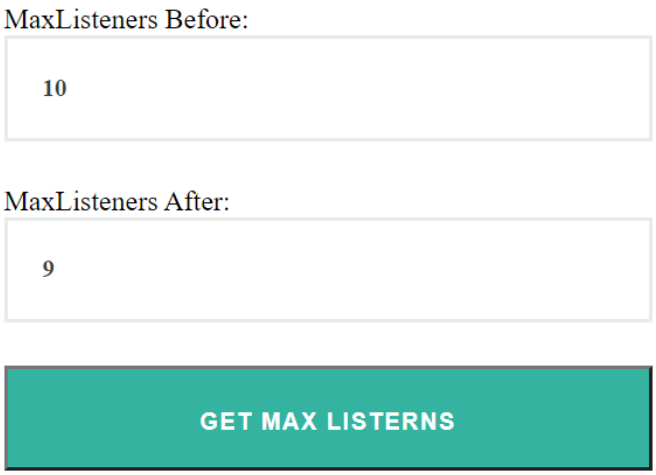
getMaxListeners():

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. Obviously not all Emitters should be limited to 10. This function allows that to be increased. Set to zero for unlimited.

```
<div class="login-form">
    <label for="">MaxListeners Before: </label>
    <h3 class="input">{{maxListnersBefore}}</h3>
    <label for="">MaxListeners After: </label>
    <h3 class="input">{{maxListnmaxListnersAfter}}</h3>
    <button class="btn" type="button" (click)="getMaxListners()">Get Max Listerns</button>
  </div>
</div>
```

```
getMaxListners() {
  this.eventEmitter.on('test', () => console.log('Test from on'));

  this.eventEmitter.once('test', () => {
    console.log('getMaxListeners Before', this.eventEmitter.getMaxListeners());
    this.maxListnersBefore=this.eventEmitter.getMaxListeners();

    this.eventEmitter.setMaxListeners(Math.max(this.eventEmitter.getMaxListeners() - 1, 0)
    this.maxListnmaxListnersAfter=this.eventEmitter.getMaxListeners();

    console.log('getMaxListeners After', this.eventEmitter.getMaxListeners());
  });

  this.eventEmitter.emit('test');
  this.eventEmitter.emit('test');
}
```

OUTPUT:




```
Test from on                                    fileupload.component.ts:51
getMaxListeners Before 10                       fileupload.component.ts:54
getMaxListeners After 9                         fileupload.component.ts:60
Test from on                                    fileupload.component.ts:51
>
```

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. Obviously not all Emitters should be limited to 10. This function allows that to be increased. Set to zero for unlimited.

listeners():

Returns an array of listeners for the specified event.

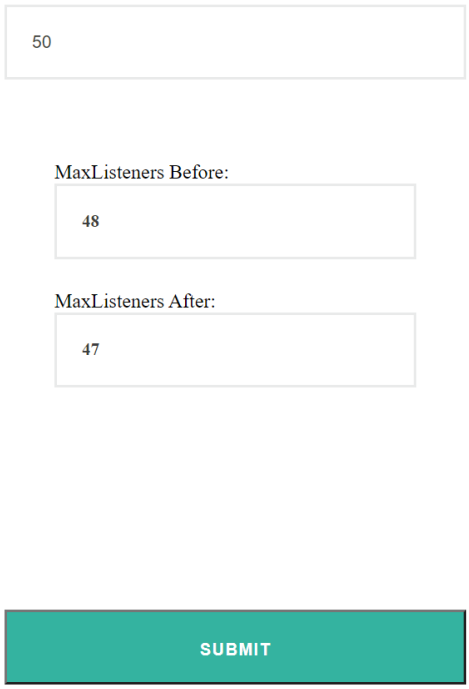
listenerCount(event):

Returns the number of listeners for a given event.

```
<div class="wrapper">
  <div class="container">
    <div class="login-form">
      <form [formGroup]="myForm" (ngSubmit)="onSubmit(myForm)">
        <input
          type="number"
          placeholder="Enter Number Of Listeners"
          class="input"
          formControlName="number"
        /><br />
        <div class="login-form">
          <label for="">MaxListeners Before: </label>
          <h3 class="input">{{maxListnersBefore1}}</h3>
          <label for="">MaxListeners After: </label>
          <h3 class="input">{{maxListnmaxListnersAfter1}}</h3>
        </div>
        <button class="btn" type="submit">Submit</button>
      </form>
    </div>
  </div>
</div>
```

```
onSubmit(form: FormGroup){
  this.eventEmitter.addListener('test2', () => console.log('Test from on 2'));
  this.eventEmitter.setMaxListeners(form.value.number);
  this.eventEmitter.on('test2', () => {

    //console.log('getMaxListeners Before', this.eventEmitter.getMaxListeners());
    this.maxListnersBefore1=this.eventEmitter.getMaxListeners();

    this.eventEmitter.setMaxListeners(Math.max(this.eventEmitter.getMaxListeners() - 1, 0)
    this.maxListnmaxListnersAfter1=this.eventEmitter.getMaxListeners();

    //console.log('getMaxListeners After', this.eventEmitter.getMaxListeners());
  });

  this.eventEmitter.emit('test2');
  this.eventEmitter.emit('test2');
  this.eventEmitter.emit('test2');

  console.log(this.eventEmitter.listeners('test2'));
  console.log(this.eventEmitter.listenerCount('test2'));


}
```

OUTPUT:



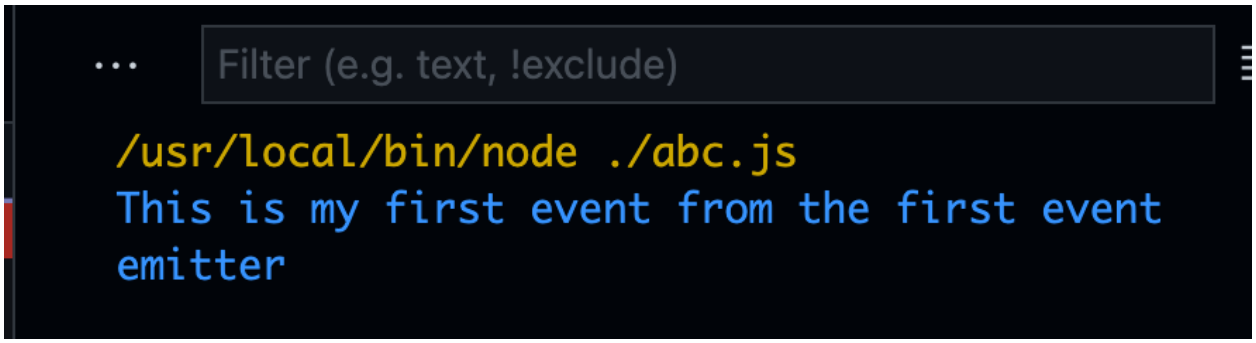
```
Test from on 2                                  fileupload.component.ts:67
Test from on 2                                  fileupload.component.ts:67
Test from on 2                                  fileupload.component.ts:67
▶ (2) [f, f]                                    fileupload.component.ts:84
2                                               fileupload.component.ts:85
>
```

USING JAVASCRIPT:

Basic Emitter:

```
// importing eventemitter and fs
const { EventEmitter } = require("events");
var fs = require('fs');

const firstEmitter = new EventEmitter();
firstEmitter.on("FirstEvent", () => {
  console.log("This is my first event from the first event emitter");
});

firstEmitter.emit('FirstEvent');
```

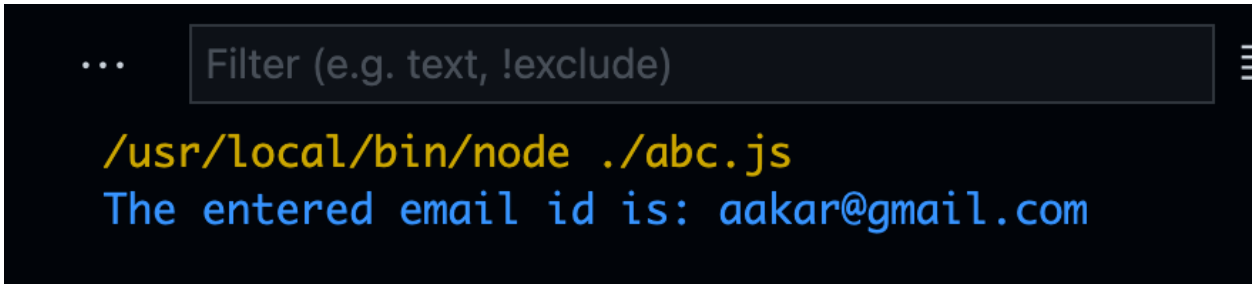
OUTPUT:



Emitter where we can pass data to the event handler:

```
const { EventEmitter } = require("events");
var fs = require('fs');

const secondEmitter = new EventEmitter();
secondEmitter.on("SecondEvent", (data) => {
   console.log('The entered email id is: ' + data);
})

secondEmitter.emit('SecondEvent', 'aakar@gmail.com');
```

OUTPUT:

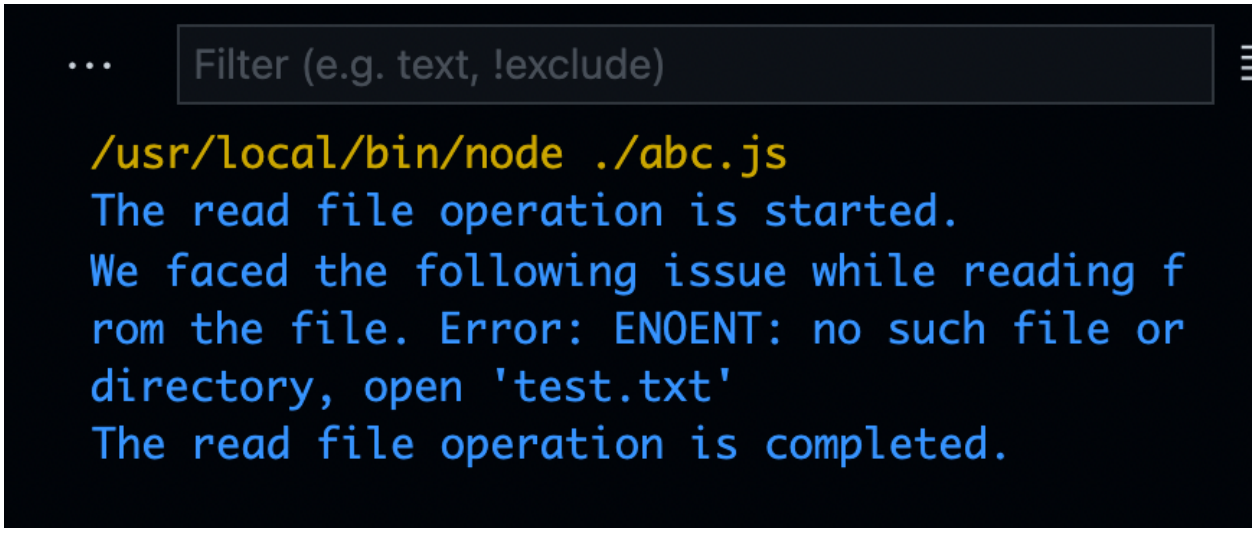


Emitter which reads files and throws an error if not present.

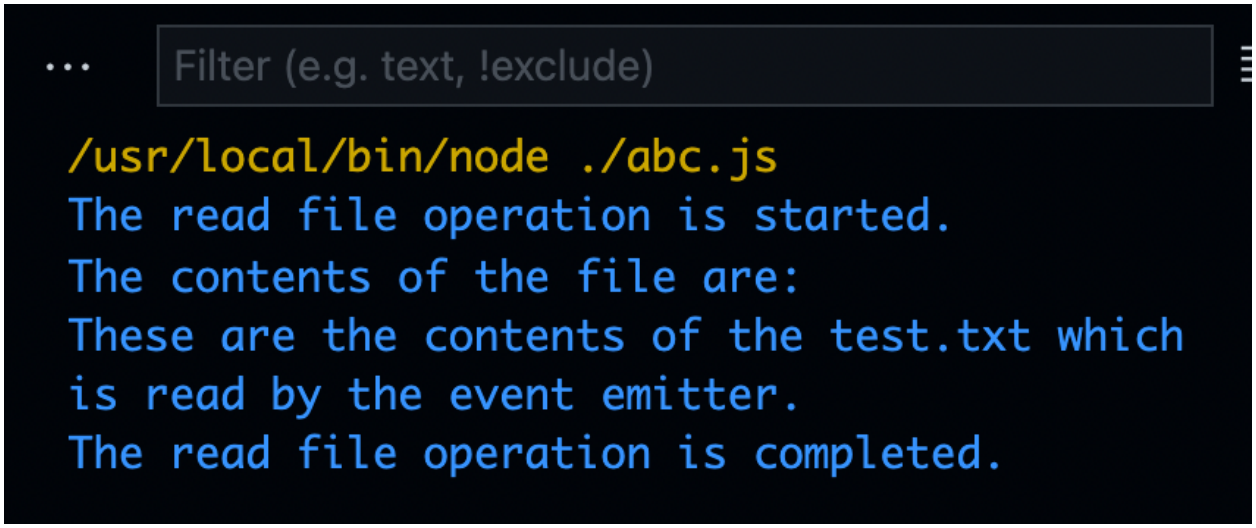
```
// importing eventemitter and fs
const { EventEmitter } = require("events");
var fs = require('fs');

const thirdEmitter = new EventEmitter();
thirdEmitter.on('start_read', function () {
   console.log("The read file operation is started.");
   fs.readFile('test.txt', 'utf8', function (err, data) {
       if (err) {
           thirdEmitter.emit('error',err);
       }
       else {
           thirdEmitter.emit('print_content', data);
       }
   });
});

thirdEmitter.on('print_content',function(data){
   console.log("The contents of the file are: ");
   console.log(data);
   thirdEmitter.emit('done');
});

thirdEmitter.on('error',function(type){
   console.log("We faced the following issue while reading from the file. "+type);
   thirdEmitter.emit('done');
});
thirdEmitter.on('done',function(){
   console.log("The read file operation is completed.");
});

thirdEmitter.emit('start_read');
```

OUTPUT

a) No file found



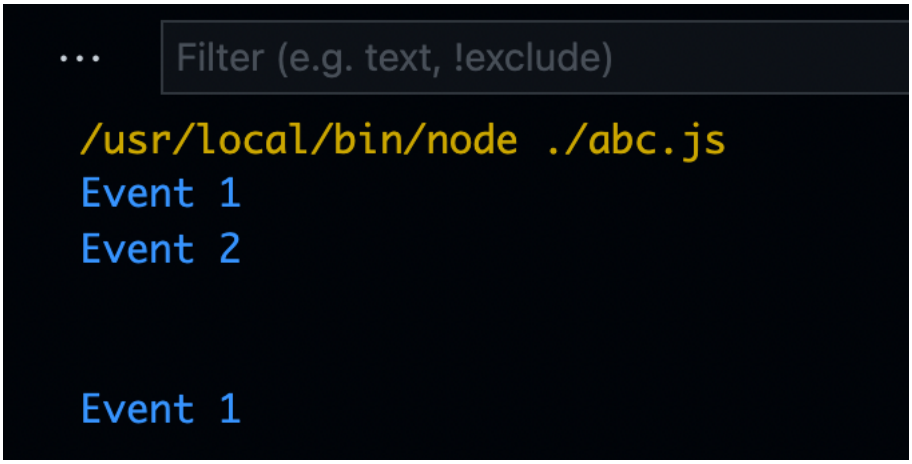
b) File is found



Emitter with remove emitter method.

```
// importing eventemitter and fs
const { EventEmitter } = require("events");
var fs = require('fs');

var forthEmitter = new EventEmitter();
var event1 = () => {
   console.log("Event 1");
   forthEmitter.removeListener('event', event2);
}

var event2 = () => {
   console.log("Event 2");
}

forthEmitter.addListener('event', event1);
forthEmitter.addListener('event', event2);

forthEmitter.emit('event');
// outputs
// Event 1
// Event 2
console.log("\n");
forthEmitter.emit('event');
// outputs
// Event 1
// We removed the second event from the event emitter and hence
// The event2 function is not called when we emit the event for the second time.
```

OUTPUT:



By default, 10 event listeners can be attached to an event otherwise it shows a warning message.

```
// importing eventemitter and fs
const { EventEmitter } = require("events");
var fs = require('fs');

var ev1 = () => { console.log("Event 1"); }
var ev2 = () => { console.log("Event 2"); }
var ev3 = () => { console.log("Event 3"); }
var ev4 = () => { console.log("Event 4"); }
var ev5 = () => { console.log("Event 5"); }
var ev6 = () => { console.log("Event 6"); }
var ev7 = () => { console.log("Event 7"); }
var ev8 = () => { console.log("Event 8"); }
var ev9 = () => { console.log("Event 9"); }
var ev10 = () => { console.log("Event 10"); }
var ev11 = () => { console.log("Event 11"); }

var fifthEmitter = new EventEmitter();
fifthEmitter.addListener('event', ev1);
fifthEmitter.addListener('event', ev2);
fifthEmitter.addListener('event', ev3);
fifthEmitter.addListener('event', ev4);
fifthEmitter.addListener('event', ev5);
fifthEmitter.addListener('event', ev6);
fifthEmitter.addListener('event', ev7);
fifthEmitter.addListener('event', ev8);
fifthEmitter.addListener('event', ev9);
fifthEmitter.addListener('event', ev10);
fifthEmitter.addListener('event', ev11);

var sixthEmitter = new EventEmitter();
sixthEmitter.on('event', ev1);
sixthEmitter.on('event', ev2);
sixthEmitter.on('event', ev3);
sixthEmitter.on('event', ev4);
sixthEmitter.on('event', ev5);
sixthEmitter.on('event', ev6);
sixthEmitter.on('event', ev7);
sixthEmitter.on('event', ev8);
sixthEmitter.on('event', ev9);
sixthEmitter.on('event', ev10);

fifthEmitter.emit('event');
// Output
// Event 1
// Event 2
// Event 3
```

```
// Event 4
// Event 5
// Event 6
// Event 7
// Event 8
// Event 9
// Event 10
// Event 11
// MaxListenersExceededWarning

sixthEmitter.emit('event');
// Output
// Event 1
// Event 2
// Event 3
// Event 4
// Event 5
// Event 6
// Event 7
// Event 8
// Event 9
// Event 10
```

OUTPUT:

```
/usr/local/bin/node ./abc.js
Event 1
Event 2
Event 3
Event 4
Event 5
Event 6
Event 7
Event 8
Event 9
Event 10
Event 11
```

```
Event 1
Event 2
Event 3
Event 4
Event 5
Event 6
Event 7
Event 8
Event 9
Event 10
(node:2716) MaxListenersExceededWarning: Possible
EventEmitter memory leak detected. 11 event listen
ers added to [EventEmitter]. Use emitter.setMaxLis
teners() to increase limit
(Use `node --trace-warnings ...` to show where the
warning was created)
```

Since the emit function is an async function, the error is printed as it comes.

We can use the .once property to show the output only once if the event is emitted again.

```
// importing eventemitter and fs
const { EventEmitter } = require("events");
var fs = require('fs');

var seventhEmitter = new EventEmitter();
var event10 = () => {
   console.log("Event 10 is only going to be emitter once. \n");
}
seventhEmitter.once('events', event10);
seventhEmitter.emit('events');
// Output
// Event 10 is only going to be emitter once.
seventhEmitter.emit('events');
// Output
//
```

OUTPUT:

```
/usr/local/bin/node ./abc.js
Event 10 is only going to be emitter once.
```