# Node.js - Event Emitter

Many objects in a Node emit events, for example, a net.Server emits an event each time a peer connects to it, an fs.readStream emits an event when the file is opened. All objects which emit events are the instances of events.EventEmitter.

# EventEmitter Class

As we have seen in the previous section, EventEmitter class lies in the events module. It is accessible via the following code −

```
// Import events module
var events = require('events');

// Create an eventEmitter object
var eventEmitter = new events.EventEmitter();
```

When an EventEmitter instance faces any error, it emits an 'error' event. When a new listener is added, 'newListener' event is fired and when a listener is removed, 'removeListener' event is fired.

EventEmitter provides multiple properties like **on** and **emit**. **on** property is used to bind a function with the event and **emit** is used to fire an event.

# Methods

| Sr.No. | Method & Description |
|---|---|
| 1 | **addListener(event, listener)**<br>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. |

| | |
|---|---|
| 2 | **on(event, listener)**<br>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. |
| 3 | **once(event, listener)**<br>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. |
| 4 | **removeListener(event, listener)**<br>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. |
| 5 | **removeAllListeners([event])**<br>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). Returns emitter, so calls can be chained. |
| 6 | **setMaxListeners(n)**<br>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. |
| 7 | **listeners(event)**<br>Returns an array of listeners for the specified event. |
| 8 | **emit(event, [arg1], [arg2], [...])**<br>Execute each of the listeners in order with the supplied arguments. Returns true if the event had listeners, false otherwise. |

# Class Methods

| Sr.No. | Method & Description |
| --- | --- |
| 1 | **listenerCount(emitter, event)**<br>Returns the number of listeners for a given event. |

# Events

| Sr.No. | Events & Description |
| --- | --- |
| 1 | **newListener**<br>• **event** − String: the event name<br>• **listener** − Function: the event handler function<br><br>This event is emitted any time a listener is added. When this event is triggered, the listener may not yet have been added to the array of listeners for the event. |
| 2 | **removeListener**<br>• **event** − String The event name<br>• **listener** − Function The event handler function<br><br>This event is emitted any time someone removes a listener. When this event is triggered, the listener may not yet have been removed from the array of listeners for the event. |

# Example

Create a js file named main.js with the following Node.js code −

Live Demo

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

// listener #1
var listner1 = function listner1() {
   console.log('listner1 executed.');
}

// listener #2
var listner2 = function listner2() {
   console.log('listner2 executed.');
```

```
}

// Bind the connection event with the listner1 function
eventEmitter.addListener('connection', listner1);

// Bind the connection event with the listner2 function
eventEmitter.on('connection', listner2);

var eventListeners = require('events').EventEmitter.listenerCount
   (eventEmitter,'connection');
console.log(eventListeners + " Listner(s) listening to connection event");

// Fire the connection event
eventEmitter.emit('connection');

// Remove the binding of listner1 function
eventEmitter.removeListener('connection', listner1);
console.log("Listner1 will not listen now.");

// Fire the connection event
eventEmitter.emit('connection');

eventListeners = require('events').EventEmitter.listenerCount(eventEmitter,'connection');
console.log(eventListeners + " Listner(s) listening to connection event");

console.log("Program Ended.");
```

Now run the main.js to see the result −

```
$ node main.js
```

Verify the Output.

```
2 Listner(s) listening to connection event
listner1 executed.
listner2 executed.
Listner1 will not listen now.
listner2 executed.
1 Listner(s) listening to connection event
Program Ended.
```

---

Revision #1
Created 16 December 2022 10:24:31 by Admin
Updated 16 December 2022 10:25:15 by Admin