Node.js EventEmitter

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.

The following example demonstrates EventEmitter class for raising and handling a custom event.

Example: Raise and Handle Node.js events

// get the reference of EventEmitter class of events module

var events = require('events');

//create an object of EventEmitter class by using above reference

var em = new events.EventEmitter();

//Subscribe for FirstEvent

em.on('FirstEvent', function (data) {

console.log('First subscriber: ' + data);

});

// Raising FirstEvent

em.emit('FirstEvent', 'This is my first Node.js event emitter example.');

In the above example, we first import the 'events' module and then create an object of EventEmitter class. We then specify event handler function using on() function. The on() method requires name of the event to handle and callback function which is called when an event is raised.

The emit() function raises the specified event. First parameter is name of the event as a string and then arguments. An event can be emitted with zero or more arguments. You can specify any name for a custom event in the emit() function.

You can also use addListener() methods to subscribe for an event as shown below.

Example: EventEmitter

var emitter = require('events').EventEmitter;

var em = new emitter();

//Subscribe FirstEvent

em.addListener('FirstEvent', function (data) {

console.log('First subscriber: ' + data);

});

//Subscribe SecondEvent

em.on('SecondEvent', function (data) {

console.log('First subscriber: ' + data);

});

// Raising FirstEvent

em.emit('FirstEvent', 'This is my first Node.js event emitter example.');

// Raising SecondEvent

em.emit('SecondEvent', 'This is my second Node.js event emitter example.');

The following table lists all the important methods of EventEmitter class.

| EventEmitter Methods | Description |
| --- | --- |
| [emitter.addListener(event, listener)](https://nodejs.org/api/events.html#events_emitter_addlistener_event_listener) | Adds a listener to the end of the listeners array for the specified event. No checks are made to see if the listener has already been added. |
| [emitter.on(event, listener)](https://nodejs.org/api/events.html#events_emitter_on_event_listener) | Adds a listener to the end of the listeners array for the specified event. No checks are made to see if the listener has already been added. It can also be called as an alias of emitter.addListener() |
| [emitter.once(event, listener)](https://nodejs.org/api/events.html#events_emitter_once_event_listener) | Adds a one time listener for the event. This listener is invoked only the next time the event is fired, after which it is removed. |
| [emitter.removeListener(event, listener)](https://nodejs.org/api/events.html#events_emitter_removelistener_event_listener) | Removes a listener from the listener array for the specified event. Caution: changes array indices in the listener array behind the listener. |
| [emitter.removeAllListeners([event])](https://nodejs.org/api/events.html#events_emitter_removealllisteners_event) | Removes all listeners, or those of the specified event. |
| [emitter.setMaxListeners(n)](https://nodejs.org/api/events.html#events_emitter_setmaxlisteners_n) | By default EventEmitters will print a warning if more than 10 listeners are added for a particular event. |
| [emitter.getMaxListeners()](https://nodejs.org/api/events.html#events_emitter_getmaxlisteners) | Returns the current maximum listener value for the emitter which is either set by emitter.setMaxListeners(n) or defaults to EventEmitter.defaultMaxListeners. |
| [emitter.listeners(event)](https://nodejs.org/api/events.html#events_emitter_listeners_event) | Returns a copy of the array of listeners for the specified event. |
| [emitter.emit(event[, arg1][, arg2][, ...])](https://nodejs.org/api/events.html#events_emitter_emit_event_arg1_arg2) | Raise the specified events with the supplied arguments. |
| [emitter.listenerCount(type)](https://nodejs.org/api/events.html#events_emitter_listenercount_type) | Returns the number of listeners listening to the type of event. |

## **Common Patterns for EventEmitters**

There are two common patterns that can be used to raise and bind an event using EventEmitter class in Node.js.

1. Return EventEmitter from a function
2. Extend the EventEmitter class

### **Return EventEmitter from a function**

In this pattern, a constructor function returns an EventEmitter object, which was used to emit events inside a function. This EventEmitter object can be used to subscribe for the events. Consider the following example.

Example: Return EventEmitter from a function

 Copy

var emitter = require('events').EventEmitter;

function LoopProcessor(num) {

var e = new emitter();

setTimeout(function () {

for (var i = 1; i <= num; i++) {

e.emit('BeforeProcess', i);

console.log('Processing number:' + i);

e.emit('AfterProcess', i);

}

}

, 2000)

return e;

}

var lp = LoopProcessor(3);

lp.on('BeforeProcess', function (data) {

console.log('About to start the process for ' + data);

});

lp.on('AfterProcess', function (data) {

console.log('Completed processing ' + data);

});

Output:

About to start the process for 1  
Processing number:1  
Completed processing 1  
About to start the process for 2  
Processing number:2  
Completed processing 2  
About to start the process for 3  
Processing number:3  
Completed processing 3

In the above LoopProcessor() function, first we create an object of EventEmitter class and then use it to emit 'BeforeProcess' and 'AfterProcess' events. Finally, we return an object of EventEmitter from the function. So now, we can use the return value of LoopProcessor function to bind these events using on() or addListener() function.

### **Extend EventEmitter Class**

In this pattern, we can extend the constructor function from EventEmitter class to emit the events.

Example: Extend EventEmitter Class

 Copy

var emitter = require('events').EventEmitter;

var util = require('util');

function LoopProcessor(num) {

var me = this;

setTimeout(function () {

for (var i = 1; i <= num; i++) {

me.emit('BeforeProcess', i);

console.log('Processing number:' + i);

me.emit('AfterProcess', i);

}

}

, 2000)

return this;

}

util.inherits(LoopProcessor, emitter)

var lp = new LoopProcessor(3);

lp.on('BeforeProcess', function (data) {

console.log('About to start the process for ' + data);

});

lp.on('AfterProcess', function (data) {

console.log('Completed processing ' + data);

});

Output:

About to start the process for 1  
Processing number:1  
Completed processing 1  
About to start the process for 2  
Processing number:2  
Completed processing 2  
About to start the process for 3  
Processing number:3  
Completed processing 3

In the above example, we have extended LoopProcessor constructor function with EventEmitter class using util.inherits() method of utility module. So, you can use EventEmitter's methods with LoopProcessor object to handle its own events.

In this way, you can use EventEmitter class to raise and handle custom events in Node.js.