**Example 1**

[https://betterprogramming.pub/using-events-in-node-js-the-basics-9a9c44dcbe30](about:blank)

One of many articles showing how you can pass data to an event emitter and then use that data.

NOTE:  all the examples have a fixed function.

const EventEmitter = require('events');  
class Emitter extends EventEmitter {}  
const eventEmitter = new Emitter();eventEmitter.on('event', (a, b) => {  
 console.log(a, b);  
});eventEmitter.emit('event', 'a', 'b');

**Example 2**

Another example from [https://stackabuse.com/handling-events-in-node-js-with-evenemitter/](about:blank) is

const timerEventEmitter = new EventEmitter();

To publish an event from this object is as easy as:

timerEventEmitter.emit("update");

We've specified the event name and published it as an event. Nothing happens since there's no listener to react to this event, though. Let's make this event repeat every second.

Using the setInterval() method, a timer is created which will publish the update event every second:

let currentTime = 0;

*// This will trigger the update event each passing second*

setInterval(() => {

currentTime++;

timerEventEmitter.emit('update', currentTime);

}, 1000);

The EventEmitter instance accepts an event name and an arbitrary set of arguments. In this case, we've passed the eventName as update and the currentTime as the time from the start of the application.

We trigger the emitter via the emit() method, which pushes the event with the information we've provided.

With our event-emitter ready, let's subscribe an event-listener to it:

timerEventEmitter.on('update', (time) => {

console.log('Message Received from publisher');

console.log(`${time} seconds passed since the program started`);

});

Using the on() method, passing the event name to specify which one we'd like to attach a listener to, allows us to create listeners. *On* the update event, a method is run which logs the time. You can add the same listener over and over again, and each one will subscribe to the event.

The second argument of the on() function is a callback that can accept any number of the extra data that was emitted by the event. Each listener can choose what data they want, once the order is kept.

**Examples 3 and 3a - see my easy\_pets3 and easy\_pets3a**  
In these examples I wanted to set up the event handlers in the pets module, a module which was extended to cat and dog.  
So I needed to either put the event emit and event handler inside the constructor function for the pet (easy\_pets3) or make use of closures (easy\_pets3a), but send it the necessary data.  
In easy\_pets3a I chose to make the handler a previously defined function so that when you extended pets in cat or dog you could change that function (over-ride its definition), the same way the species in over-ridden in cat and dog.  
The code is in the pets folder of the simple projects at [http://web.simmons.edu/~menzin/CS321/Unit\_8\_Server\_Side\_Node/SimpleProjects/](about:blank)

**Example 4**

Finally, there was an example of using a list of event handlers at [https://blog.logrocket.com/how-build-custom-node-js-event-emitters/](about:blank)