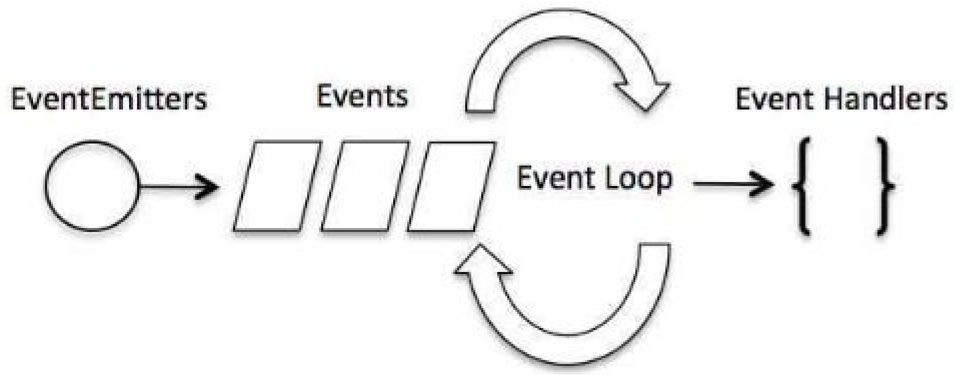
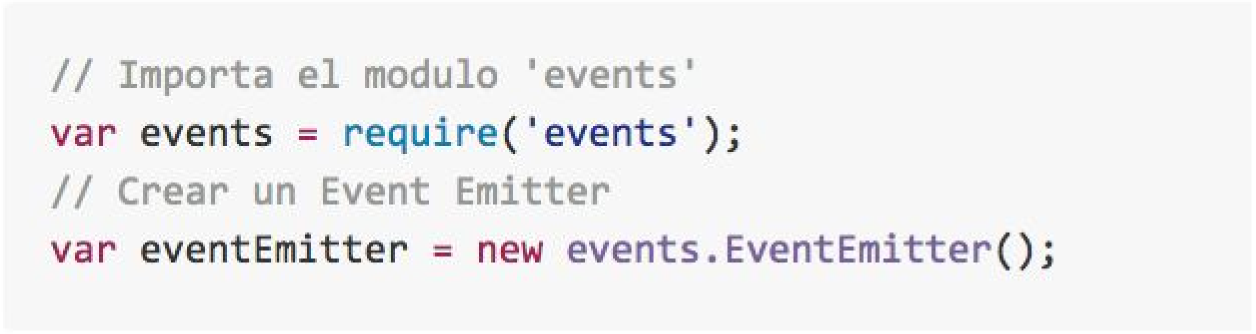
Event Emitters

Gabriel Cuchacovich
Daniela Quiroz

Event Emitters

- Node usa event-driven

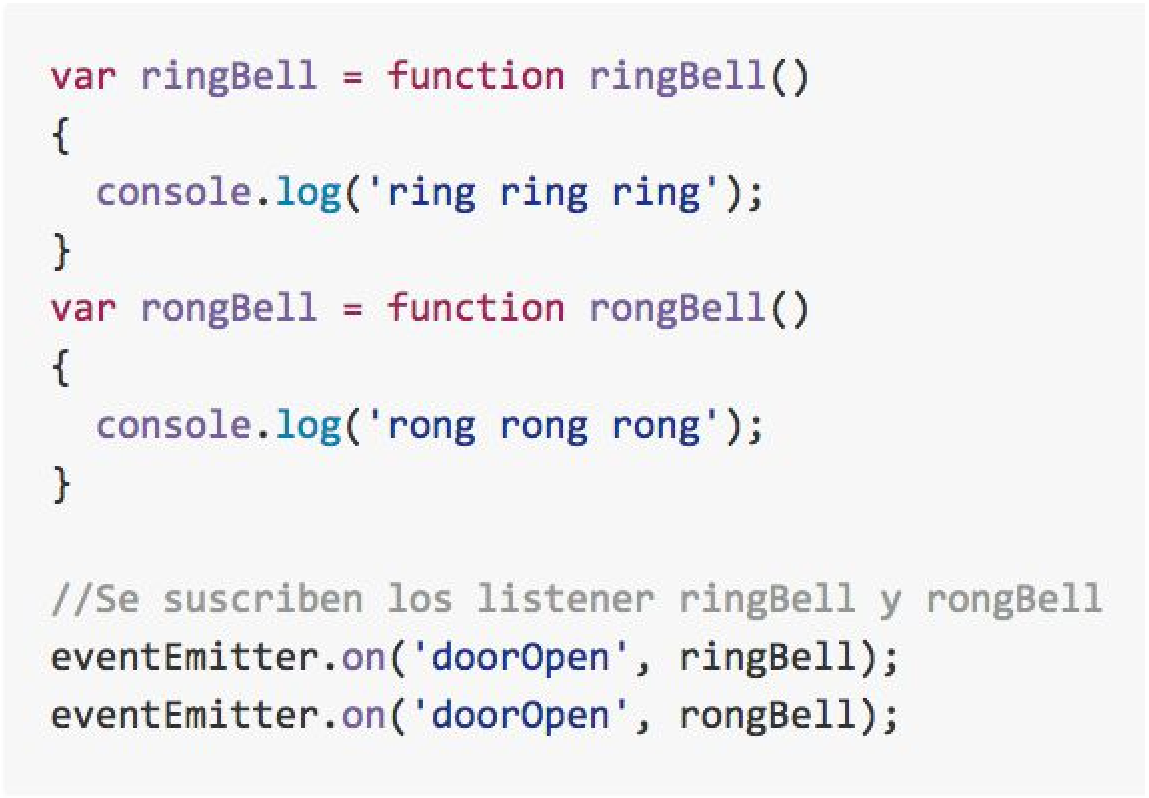




```
// Importa el modulo 'events'
var events = require('events');
// Crear un Event Emitter
var eventEmitter = new events.EventEmitter();
```

Métodos

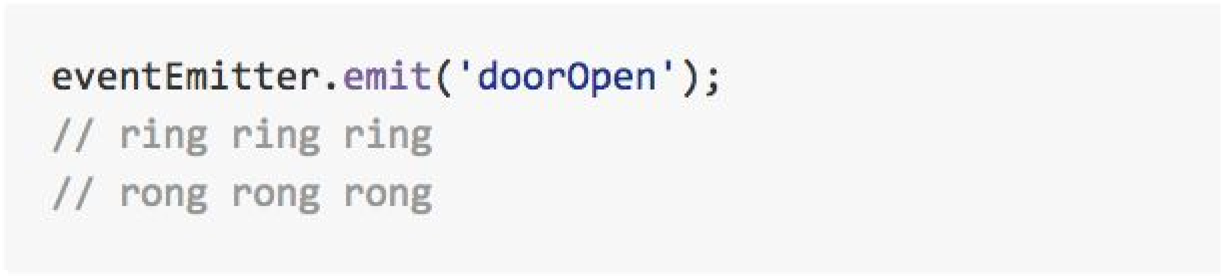
emitter.addListener(event, listener) ⇔ emitter.on(event, listener)



```
var ringBell = function ringBell()
{
  console.log('ring ring ring');
}
var rongBell = function rongBell()
{
  console.log('rong rong rong');
}

//Se suscriben los listener ringBell y rongBell
eventEmitter.on('doorOpen', ringBell);
eventEmitter.on('doorOpen', rongBell);
```

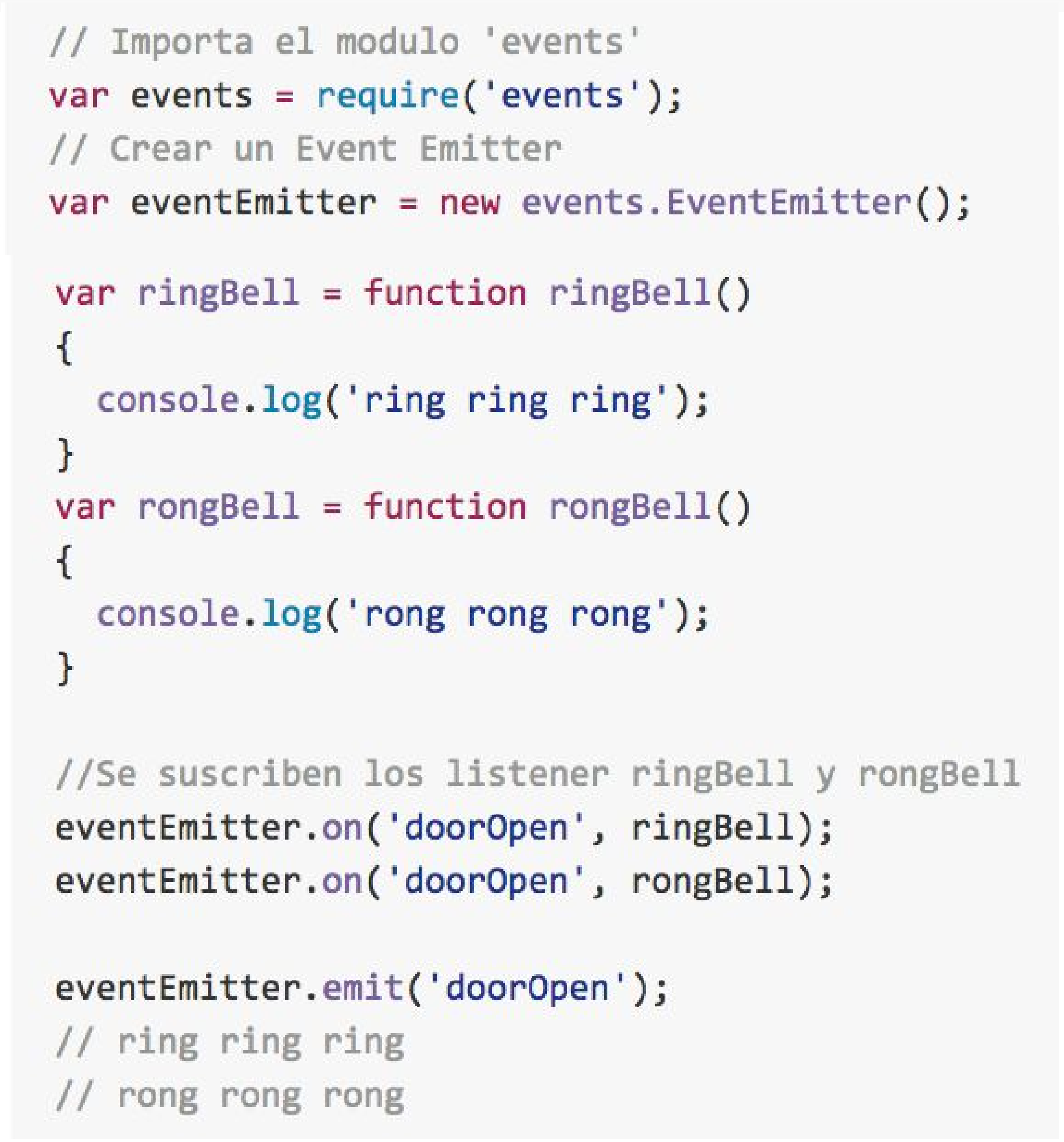
Métodos

emitter.emit(event, [arg1], [arg2], [...])



```
eventEmitter.emit('doorOpen');
// ring ring ring
// rong rong rong
```

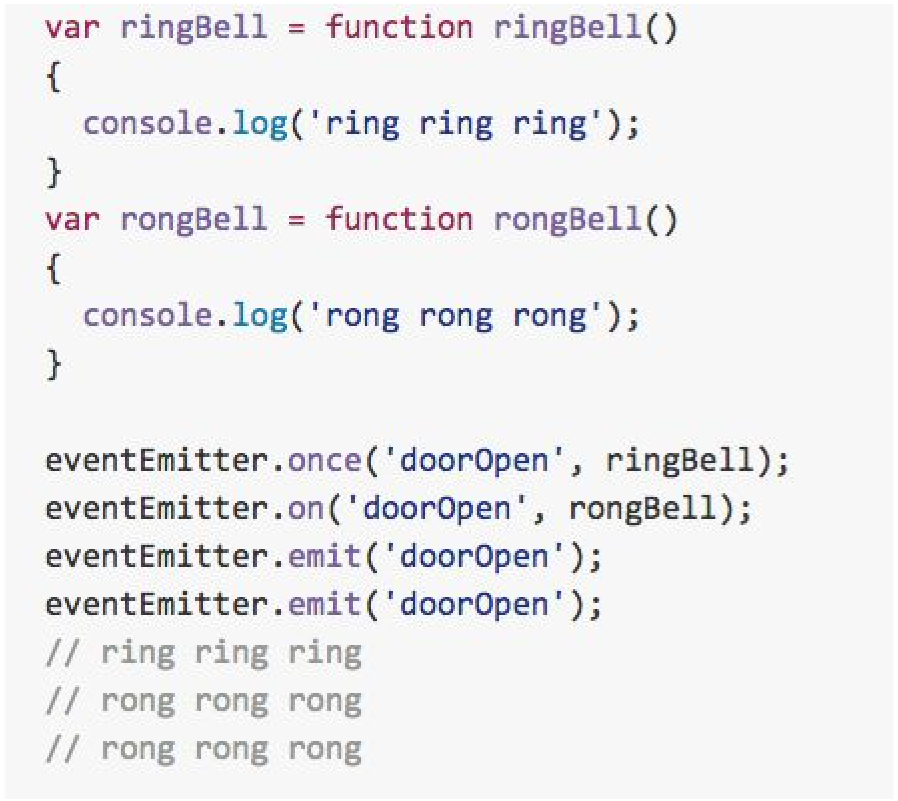
Métodos



```
// Importa el modulo 'events'
var events = require('events');
// Crear un Event Emitter
var eventEmitter = new events.EventEmitter();

var ringBell = function ringBell()
{
  console.log('ring ring ring');
}
var rongBell = function rongBell()
{
  console.log('rong rong rong');
}

//Se suscriben los listener ringBell y rongBell
eventEmitter.on('doorOpen', ringBell);
eventEmitter.on('doorOpen', rongBell);

eventEmitter.emit('doorOpen');
// ring ring ring
// rong rong rong
```

Métodos

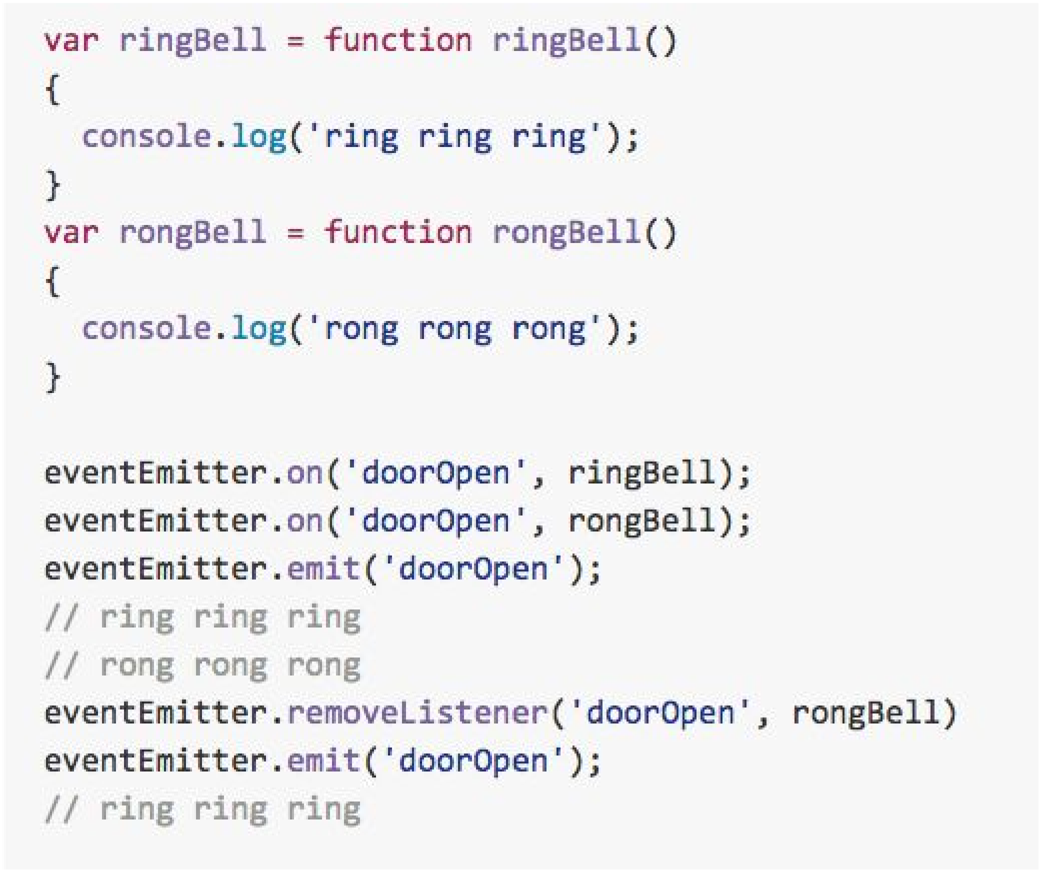
emitter.once(event, listener)



```
var ringBell = function ringBell()
{
  console.log('ring ring ring');
}
var rongBell = function rongBell()
{
  console.log('rong rong rong');
}

eventEmitter.once('doorOpen', ringBell);
eventEmitter.on('doorOpen', rongBell);
eventEmitter.emit('doorOpen');
eventEmitter.emit('doorOpen');
// ring ring ring
// rong rong rong
// rong rong rong
```

Métodos

emitter.removeListener(event, listener)



```
var ringBell = function ringBell()
{
  console.log('ring ring ring');
}
var rongBell = function rongBell()
{
  console.log('rong rong rong');
}

eventEmitter.on('doorOpen', ringBell);
eventEmitter.on('doorOpen', rongBell);
eventEmitter.emit('doorOpen');
// ring ring ring
// rong rong rong
eventEmitter.removeListener('doorOpen', rongBell)
eventEmitter.emit('doorOpen');
// ring ring ring
```

Métodos

emitter.removeAllListeners([event])

emitter.prependListener(eventName, listener)

emitter.prependOnceListener(eventName, listener)

emitter.eventNames()

emitter.listeners(event)

emitter.listenerCount(event)

Métodos

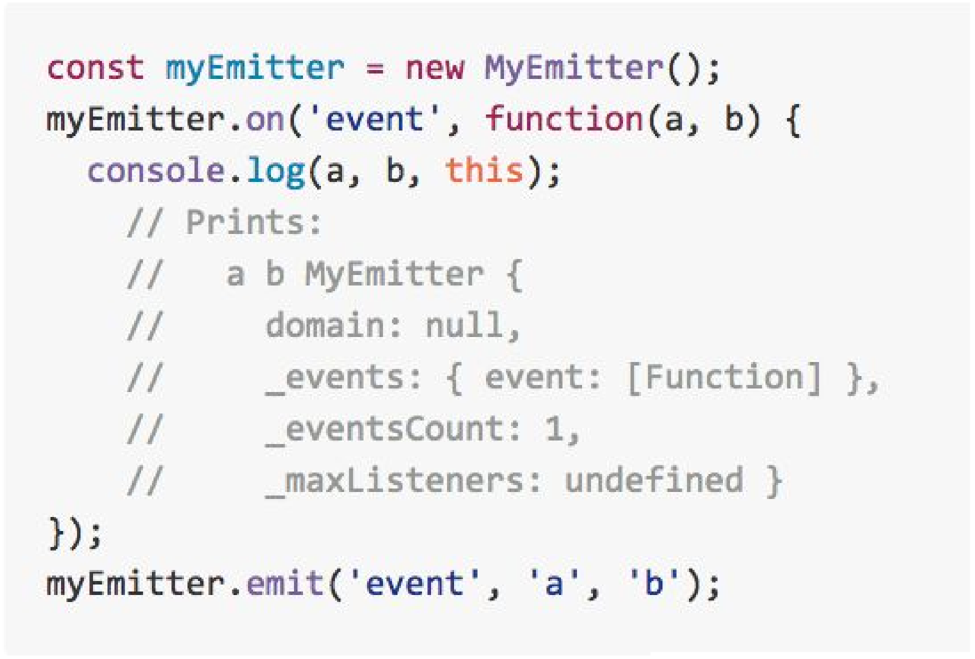
Por defecto máximo 10 listeners

emitter.setMaxListeners(n)

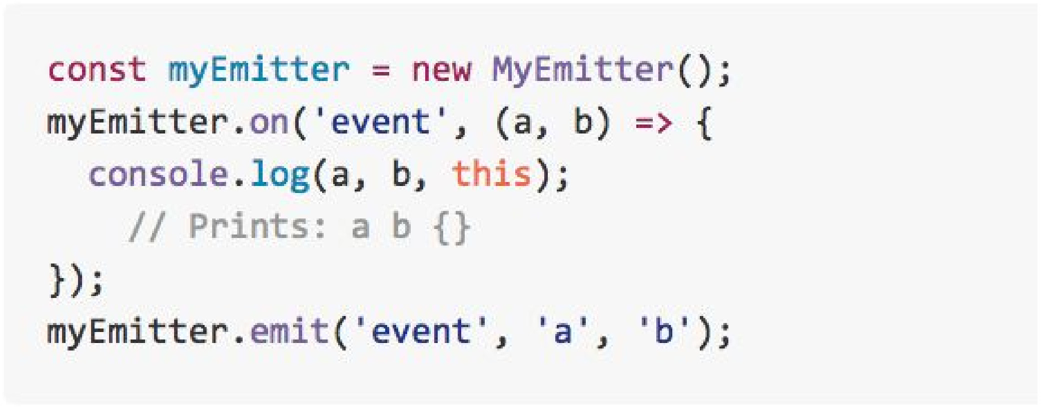
emitter.getMaxListeners()

-> property EventEmitter.defaultMaxListeners (varíara todos)

Consideraciones con Array Functions



```
const myEmitter = new MyEmitter();
myEmitter.on('event', function(a, b) {
  console.log(a, b, this);
    // Prints:
    //   a b MyEmitter {
    //     domain: null,
    //     _events: { event: [Function] },
    //     _eventsCount: 1,
    //     _maxListeners: undefined }
});
myEmitter.emit('event', 'a', 'b');
```



```
const myEmitter = new MyEmitter();
myEmitter.on('event', (a, b) => {
  console.log(a, b, this);
    // Prints: a b {}
});
myEmitter.emit('event', 'a', 'b');
```

Ejemplos

Referencias

https://nodejs.org/api/events.html

http://www.tutorialspoint.com/nodejs

Consideraciones Asíncrono vs. Síncrono

Para permitir un modo asíncrono se puede usar setImmediate o process.nextTick

```
const myEmitter = new MyEmitter();
myEmitter.on('event', (a, b) => {
  setImmediate(() => {
    console.log('this happens asynchronously');
  });
});
myEmitter.emit('event', 'a', 'b');
```