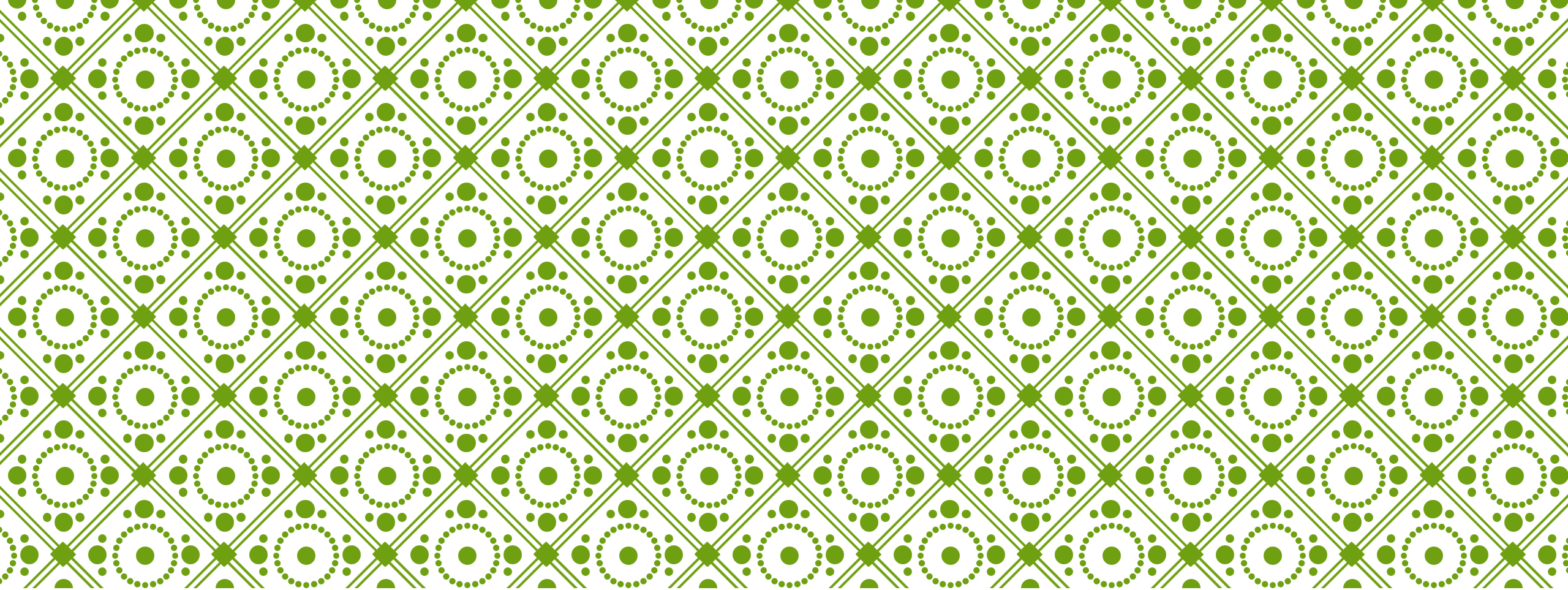


ASYNCHRONOUS EVENT PROGRAMMING WITH NODE

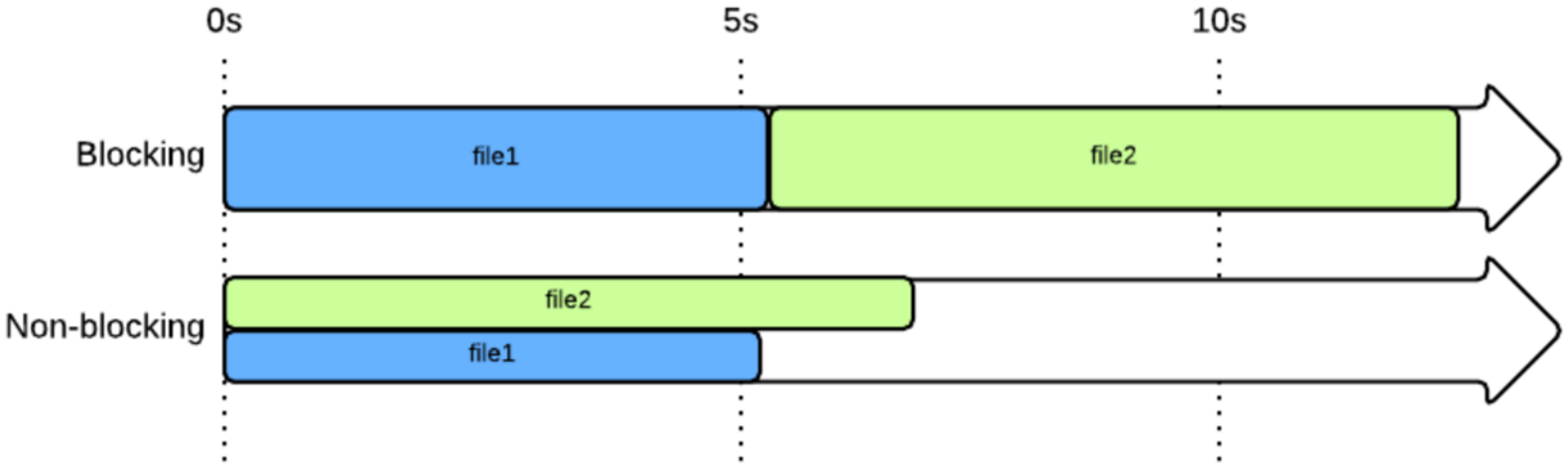
Andrew Sheehan
MET CS602

EVENT-DRIVEN

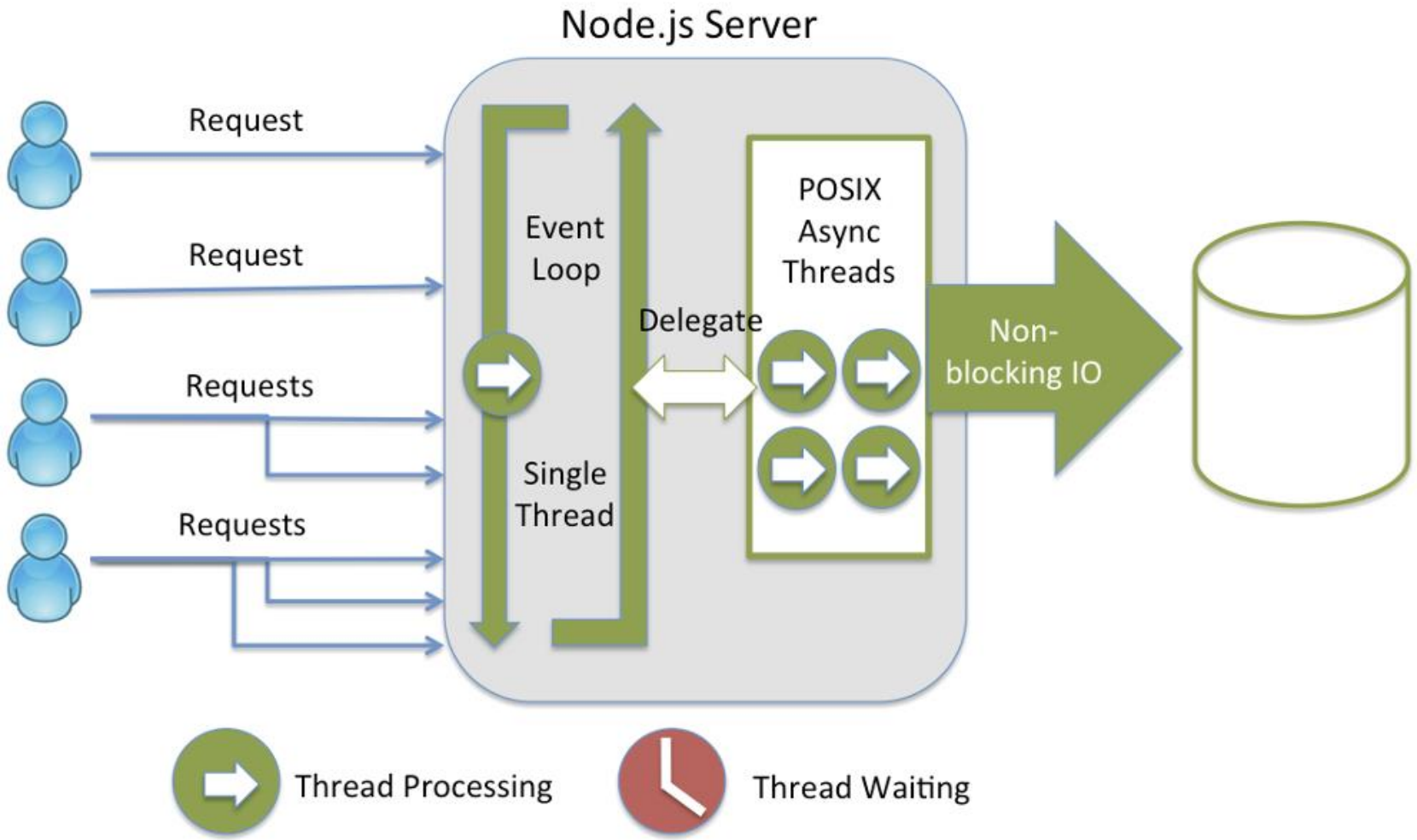
One of Node's Core Strengths: Faster Processing.

BLOCKING VS NON BLOCKING

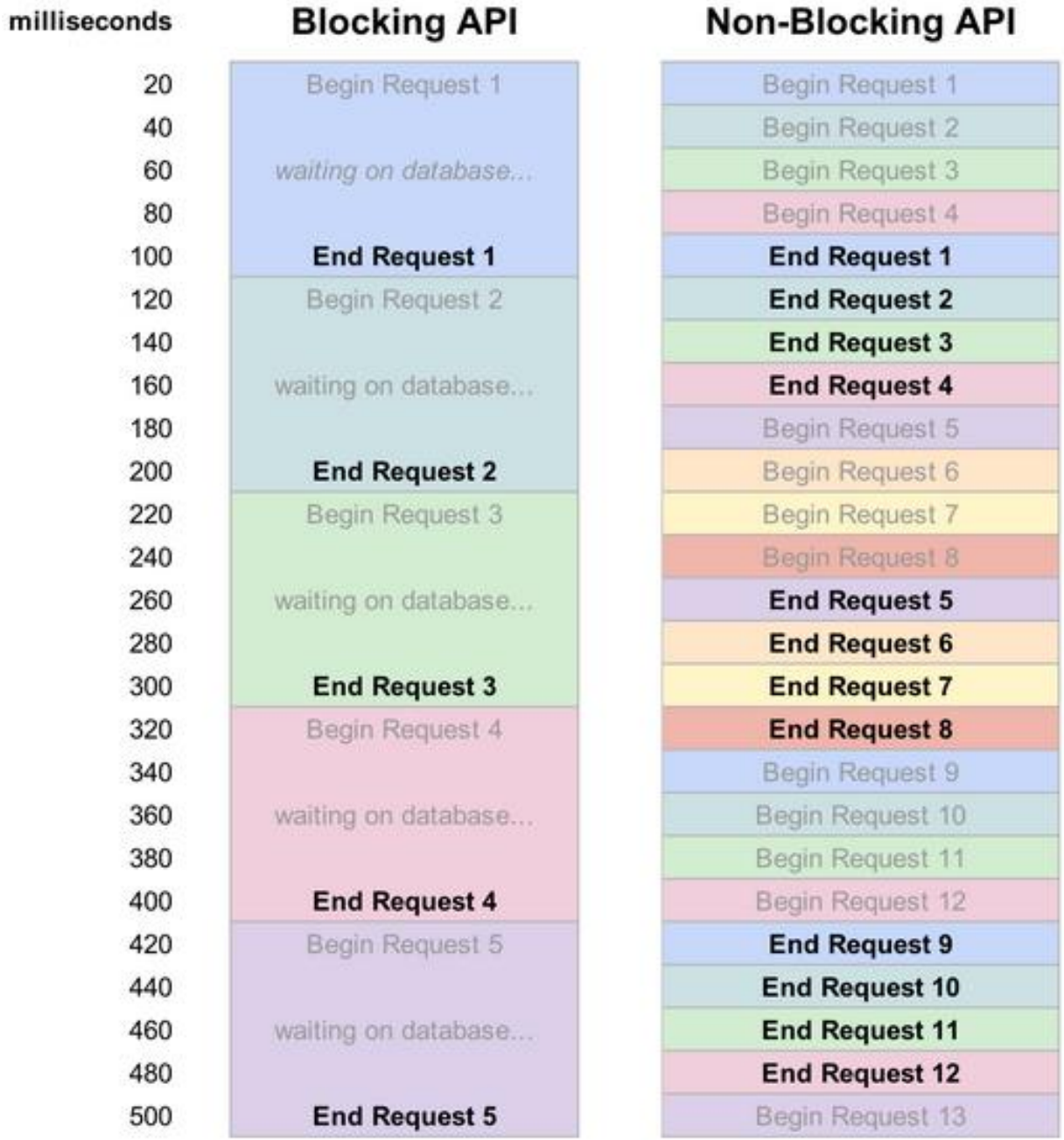
Rather than waiting for an operation to finish, create a callback that will be invoked when the operation ends.



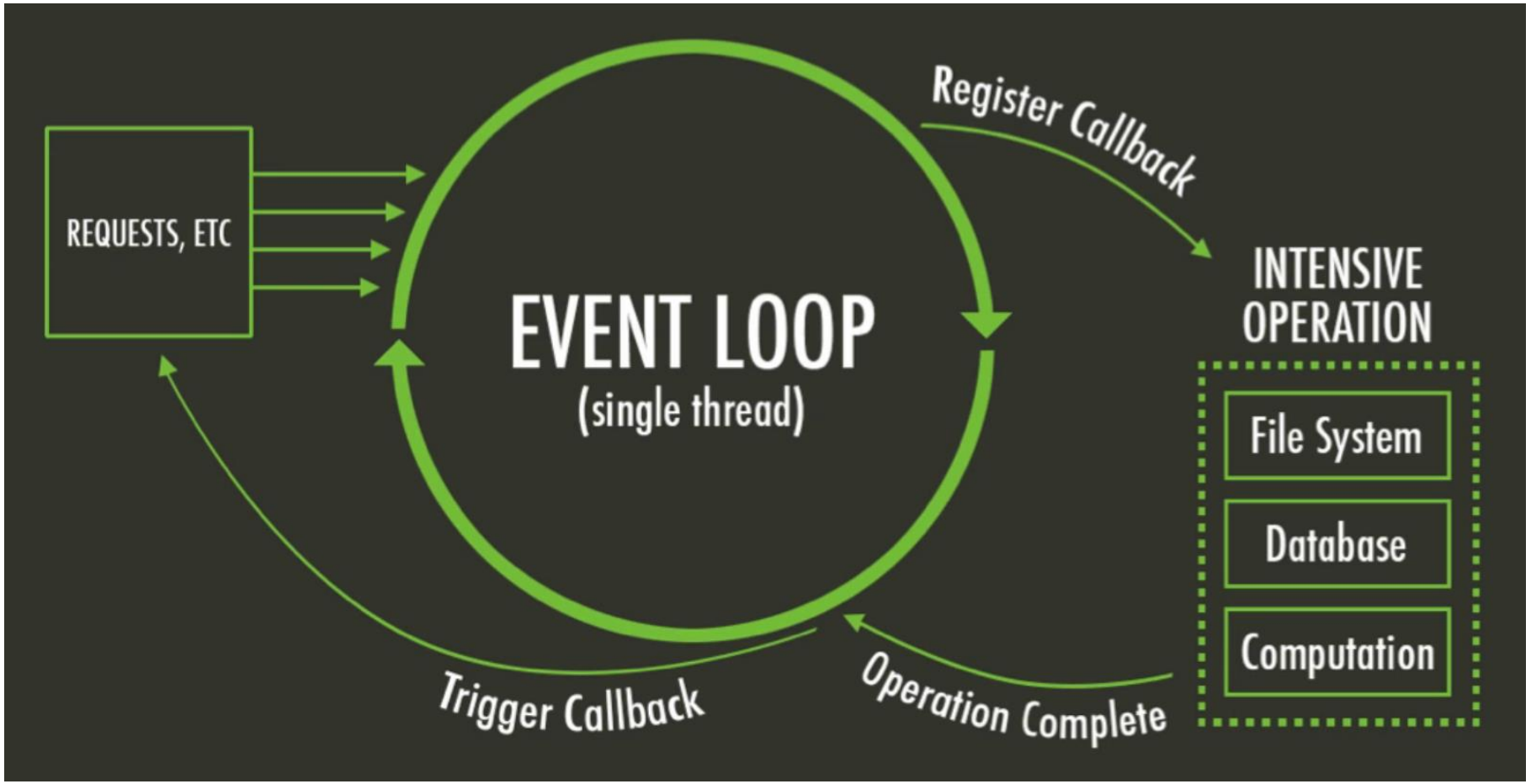
HIGH LEVEL OVERVIEW

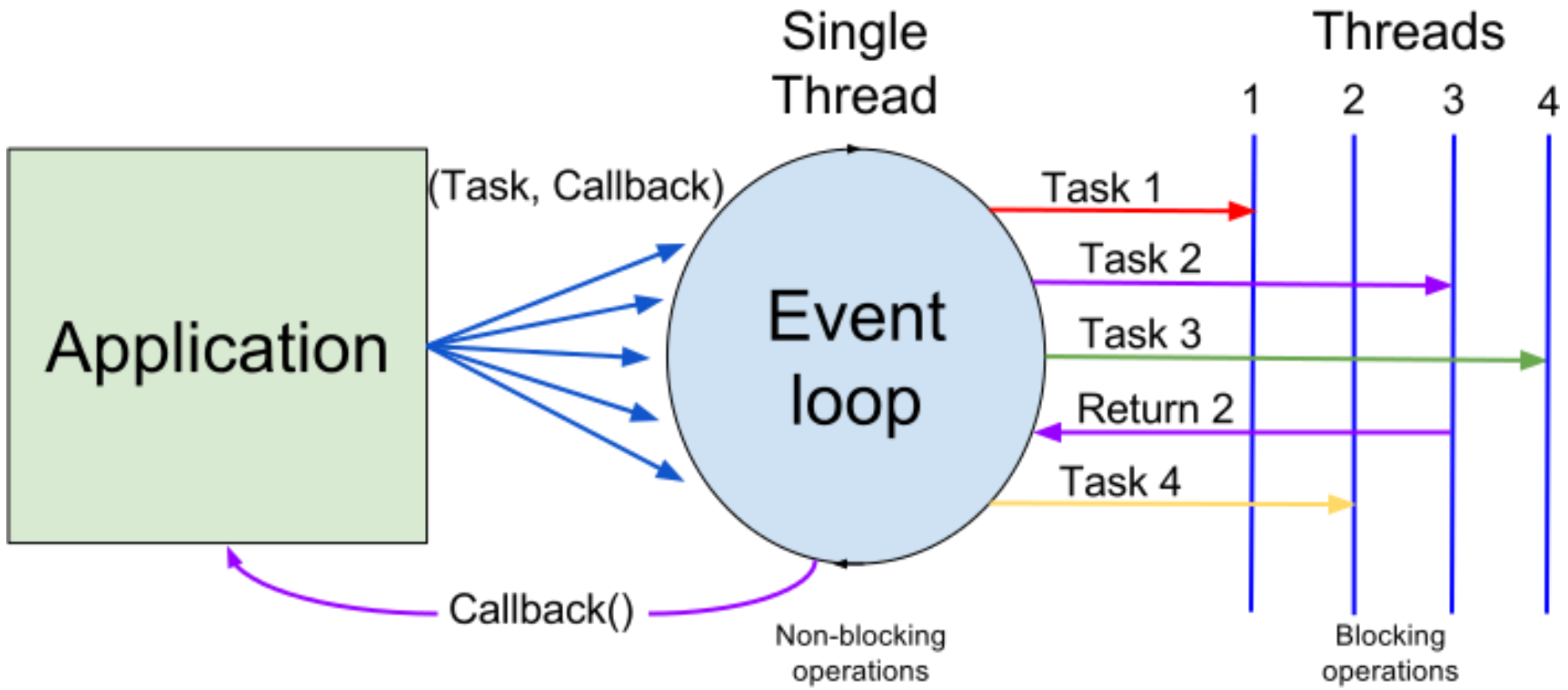


BLOCKING VS NON BLOCKING



SINGLE THREADED

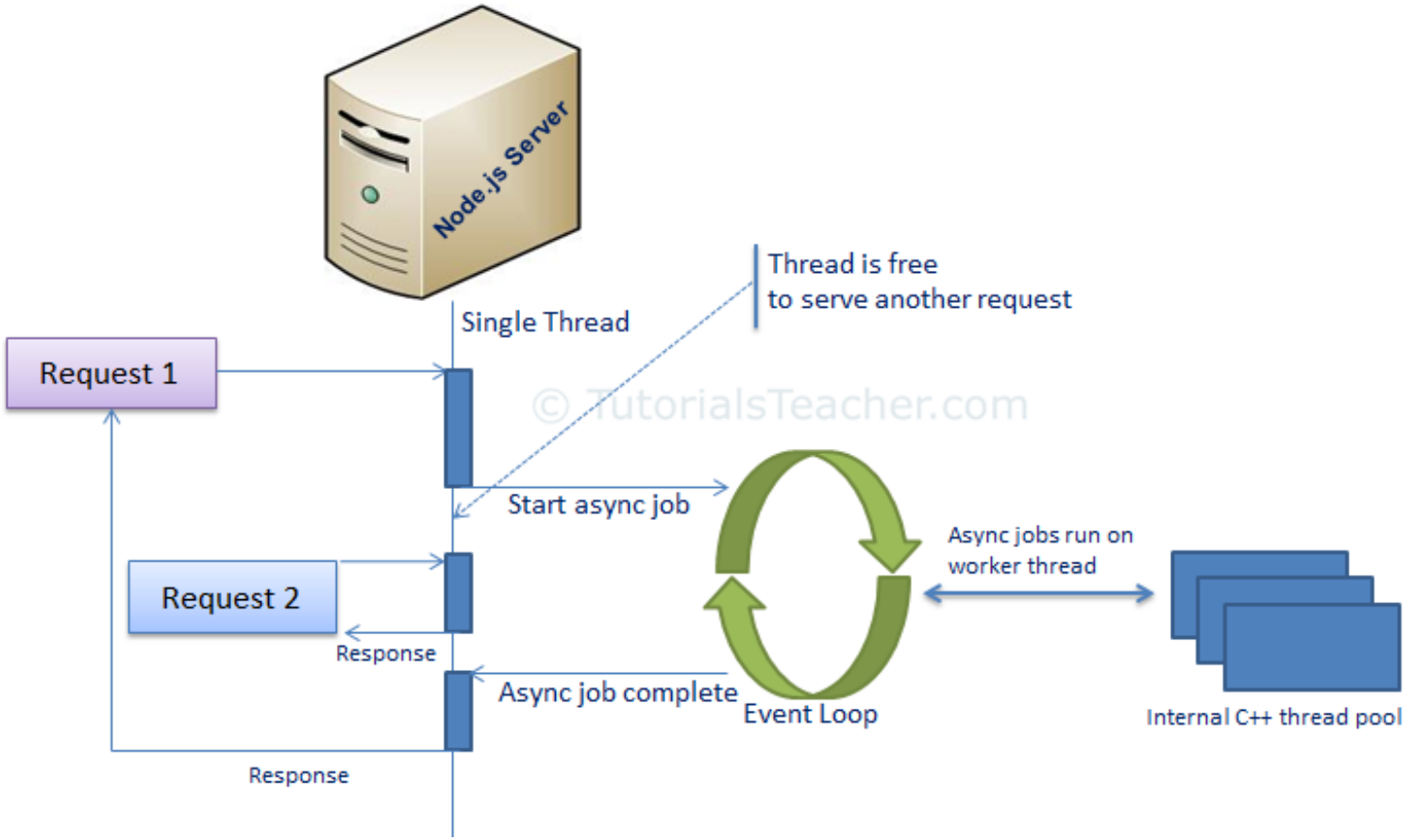




The main event loop is single-threaded but most of the I/O runs on separate threads

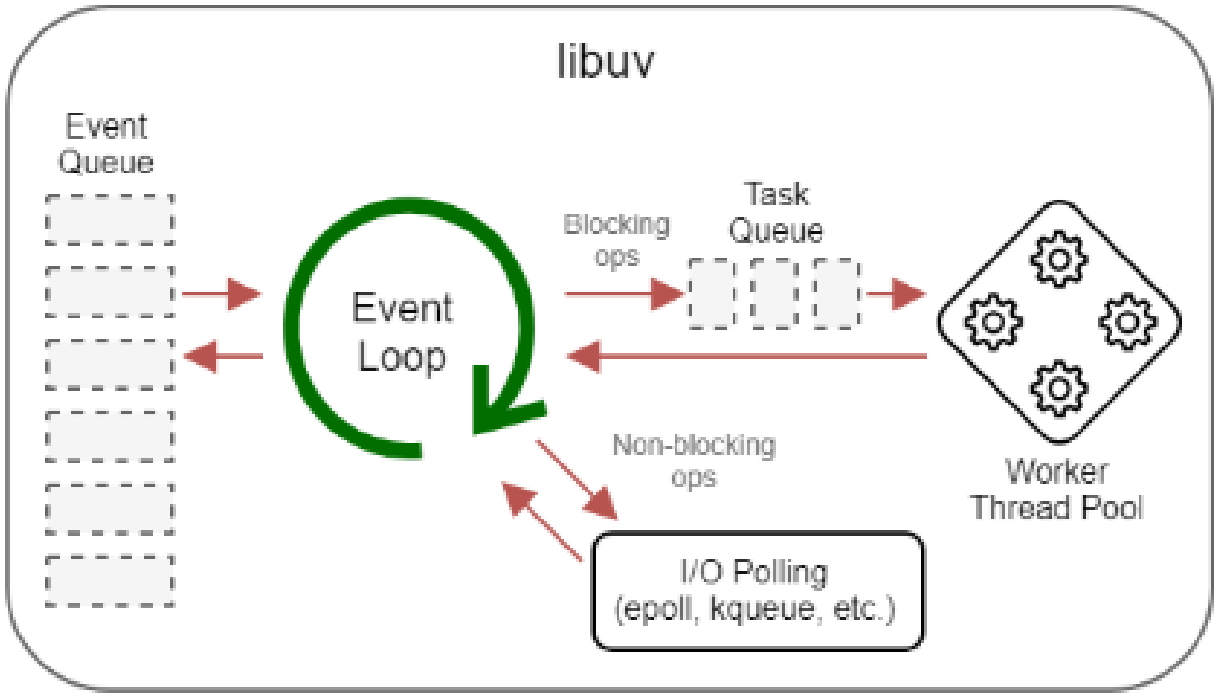
HIGH LEVEL THREADS

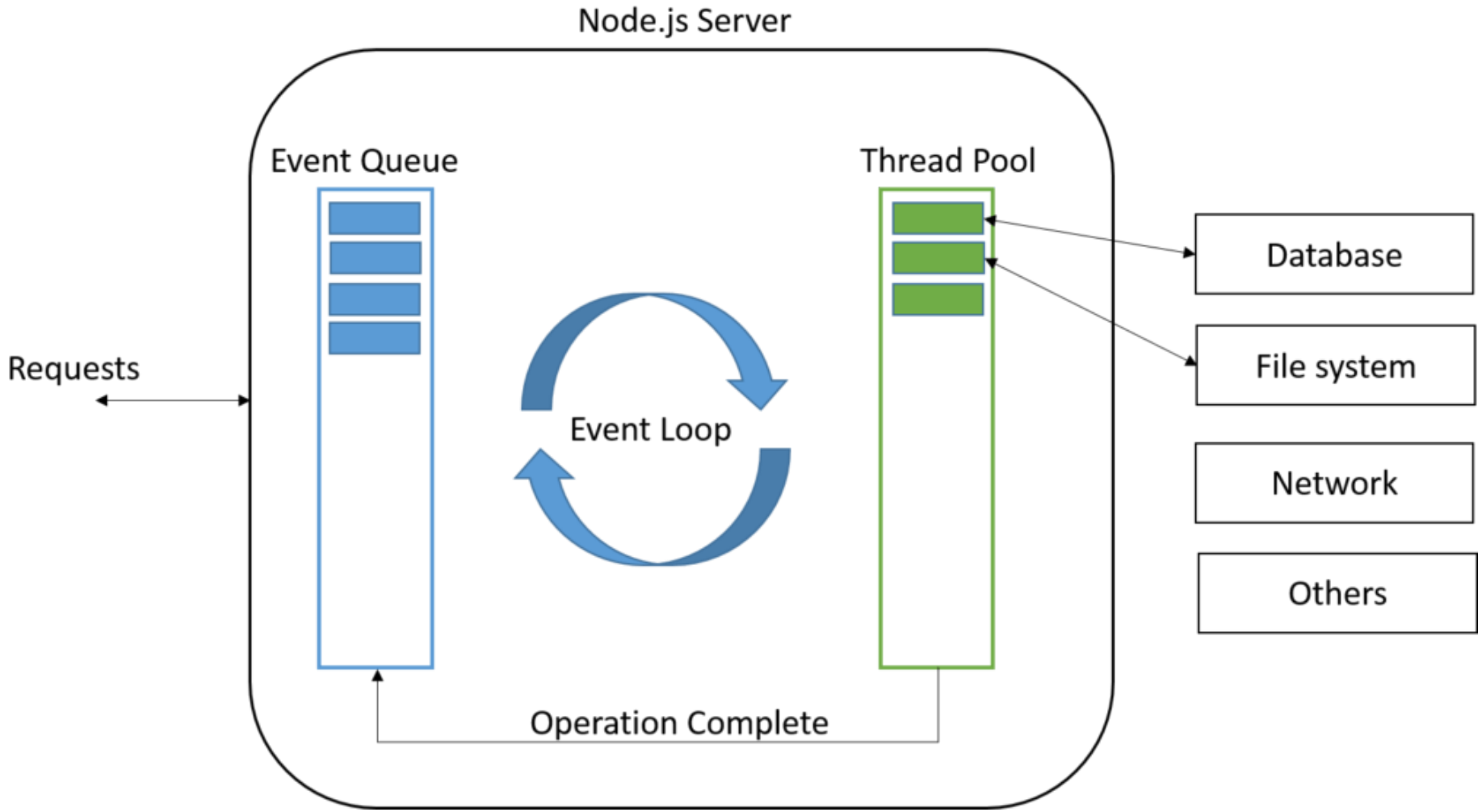
Threads run in processes; one process can have many threads in it, and as they are in same process, they share a memory space.



EVENT LOOP WORKER THREADS

Events are pushed to the main thread then worker threads process the request.





The event loop is implemented via **libuv** (C++).

EVENT EMITTER

Have custom events? Use the events module.

```
const emitter = require("events");
```

EVENT EMITTER

Who else uses EventEmitter class?

EVENT EMITTER
NOTES & COMMENTS

The EventEmitter will call all listeners synchronously in the order in which they were registered.

This is important to ensure the proper sequencing of events and to avoid race conditions or logic errors

EVENT EMITTER

```
const EventEmitter = require('events');


class MyEmitter extends EventEmitter {}


const myEmitter = new MyEmitter();
myEmitter.on('event', () => {
  console.log('an event occurred!');
});
myEmitter.emit('event');
```

Used to raise and handle custom events

EXAMPLE: EMITTER

```
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);
});
```

EVENT EMITTER
ONE TIME

To emit (send) a one-time event

myemitter.**once()**

Calling it again will do nothing…

EVENT EMITTER
UNREGISTER EVENTS

myemitter.off('event', listener)

myemitter.removeListener('event', listener)