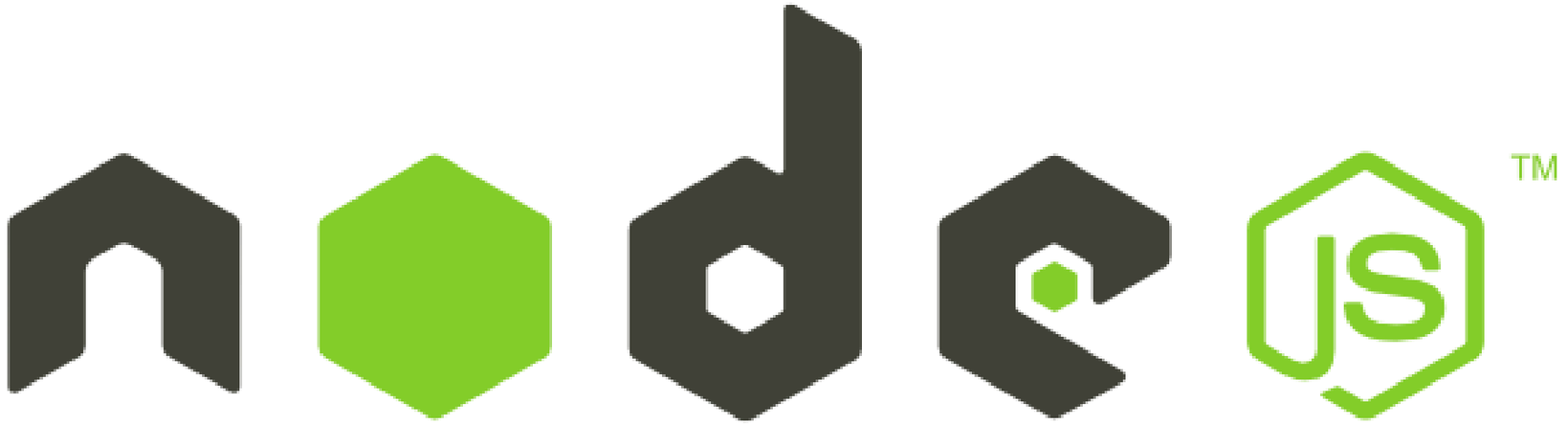


Introduction to Node.js

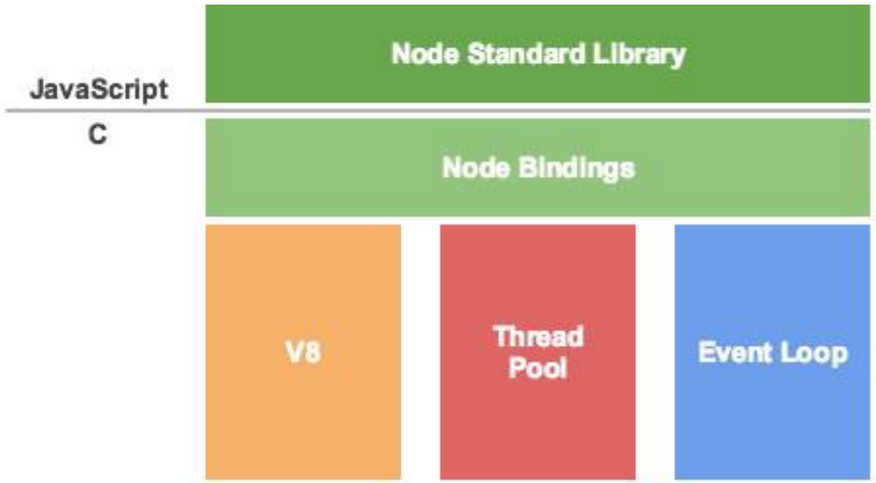
Frank Walsh
Diarmuid O'Connor

Agenda

- What is node.js
- Non Blocking and Blocking
- Event-based processes
- Callbacks in node
- Node Package Manager(NPM)
- Creating a node app
- Introduction to Express

What's Node: Basics

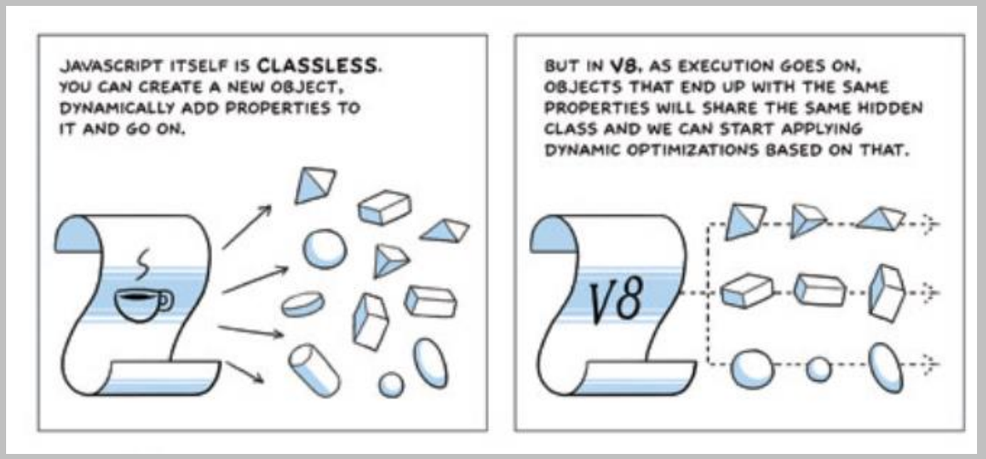
- A Javascript runtime. “Server side JS”
- The “.js” doesn’t mean that it’s written completely in JavaScript.
  - approx. 40% JS and 60% C++
- Ecosystem of packages (NPM)
- Official site: “Node's goal is to provide an easy way to build scalable network programs”.
- Single Threaded, Event based
  - Supports concurrency using events and callbacks…



What’s Node: V8.

- Embedded C++ component
- Javascript virtual machine.
- Very fast and platform independent
- Find out a bit about it's history here:

http://www.google.com/googlebooks/chrome/big_12.html



What is Node.js: Event-based

This Photo by Unknown Author is licensed under CC BY-NC

- Input/Output (io) is slow.
  - Reading/writing to data store, network access.
  - Read 4K randomly from SSD* 150,000 ns ~1GB/sec SSD
  - Round trip over network within same datacenter 500,000 ns
  - Send packet US->Netherlands->US 150,000,000 ns
- CPU operations are fast.
  - L1 cache reference 0.5 ns
  - L2 cache reference 7 ns

- I/O operations detrimental to highly concurrent apps (e.g. web applications)
- Solutions to deal with this are:
  - **Blocking code** combined with multiple threads of execution (e.g. Apache, IIS)
  - **Non-blocking**, **event-based code** in single thread (e.g. NGINX, Node.js)

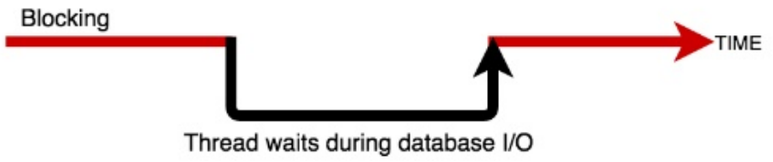
Source: https://gist.github.com/jboner/2841832

$1ns = 10^{-9}$ s (0.000000001s)

Blocking/Non-blocking Example

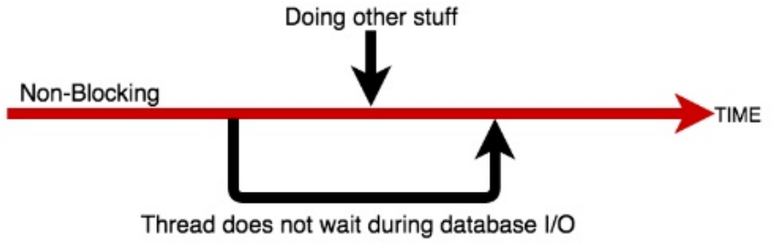
Blocking

1. Read from file and set equal to contents
2. Print Contents
3. Do other stuff...



Non-blocking

1) Read from File
    Whenever read is complete, print contents
2) Do other stuff...



Blocking/Non-blocking: JS

Blocking

```
import fs from 'fs';

const contents = fs.readFileSync('./readme.md', 'utf8');
console.log(contents);
console.log('Doing something else');
```

Console output

Hello World……
Doing something else

Non-blocking

callback

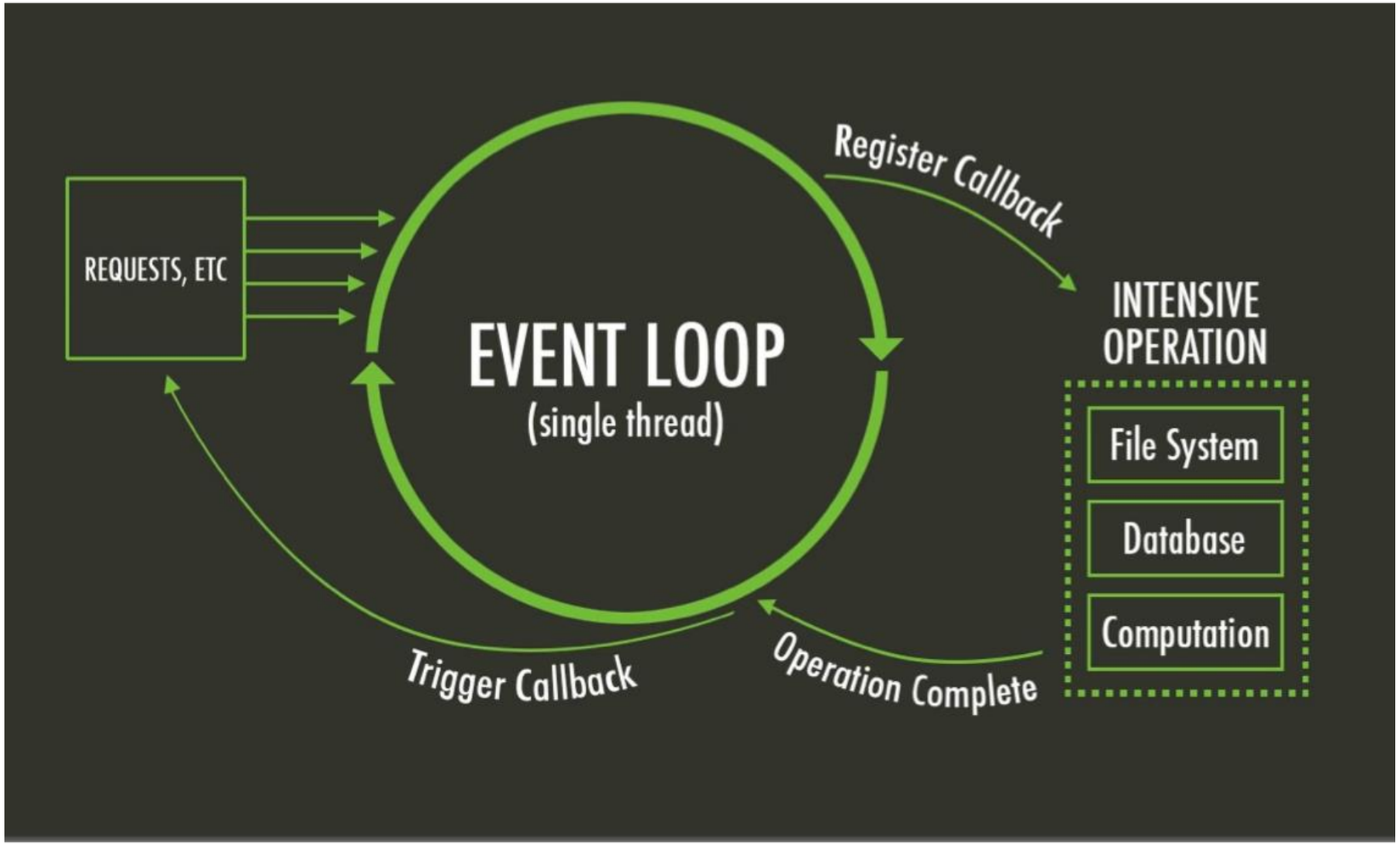
```
import fs from 'fs';
fs.readFile('./text.txt','uft8', (err, contents) => {
    console.log(contents);
});
console.log('Doing something else');
```

Console output

Doing something else
Hello World ……

The Node Event Loop and Callbacks

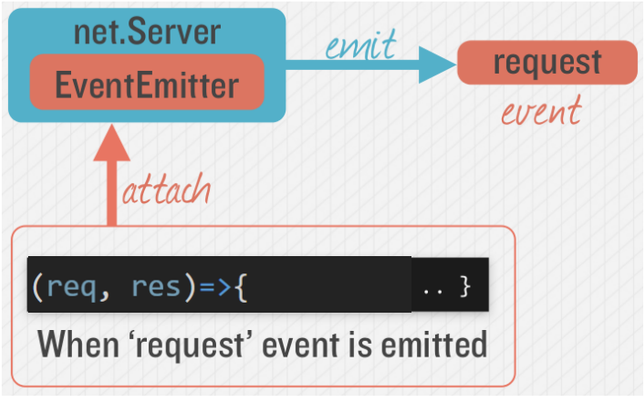
- A **Callback** is a function called at the completion of a given task. This prevents any blocking, and allows other code to be run in the meantime
- The Event Loop checks for known events, registers Callbacks and, triggers callback on completion of operation

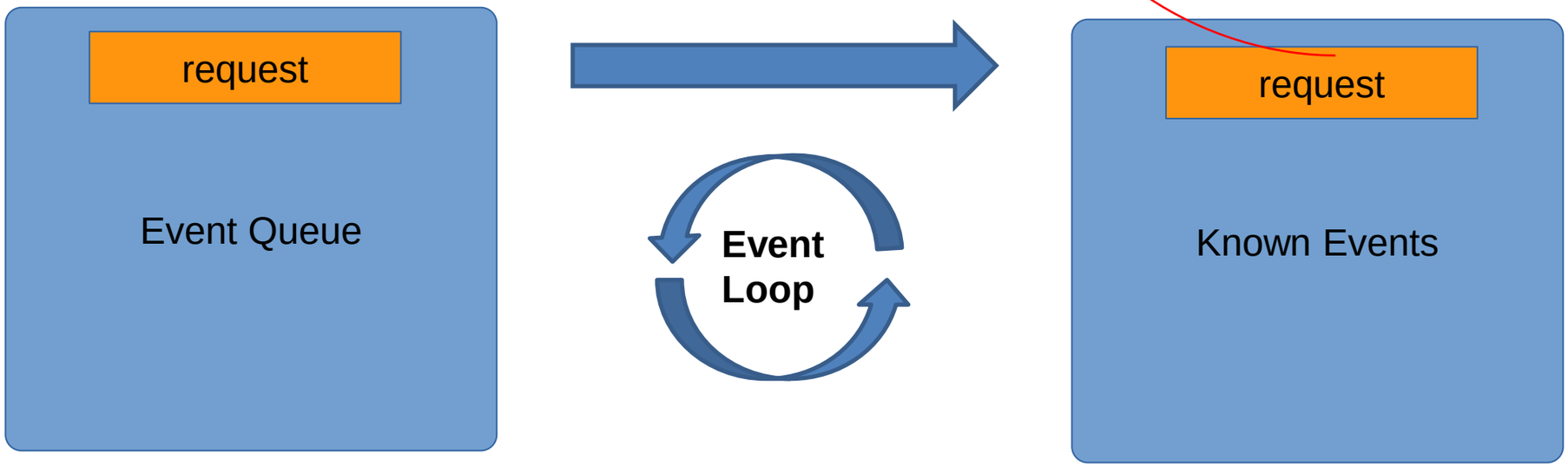


This Photo by Unknown Author is licensed under CC BY-SA

Node.js - Simple HTTP Server

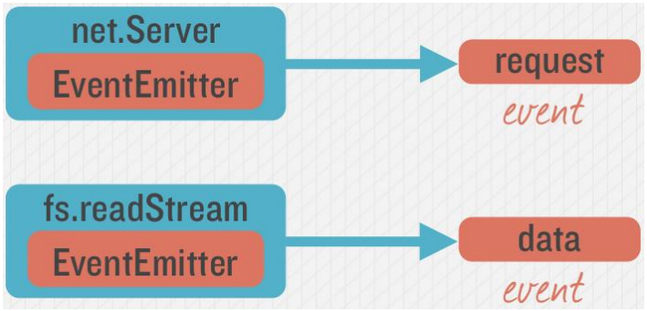
```
1   import http from 'http';
2   const port=8080;
3
4   var server = http.createServer((req, res)=>{
5       response.writeHead(200);
6       response.end("Hello World!");
7   });
8
9   server.listen(port);
10  console.log(`Server running at ${port}`);
```





Emitting Event in Node

Many objects can emit events in node.



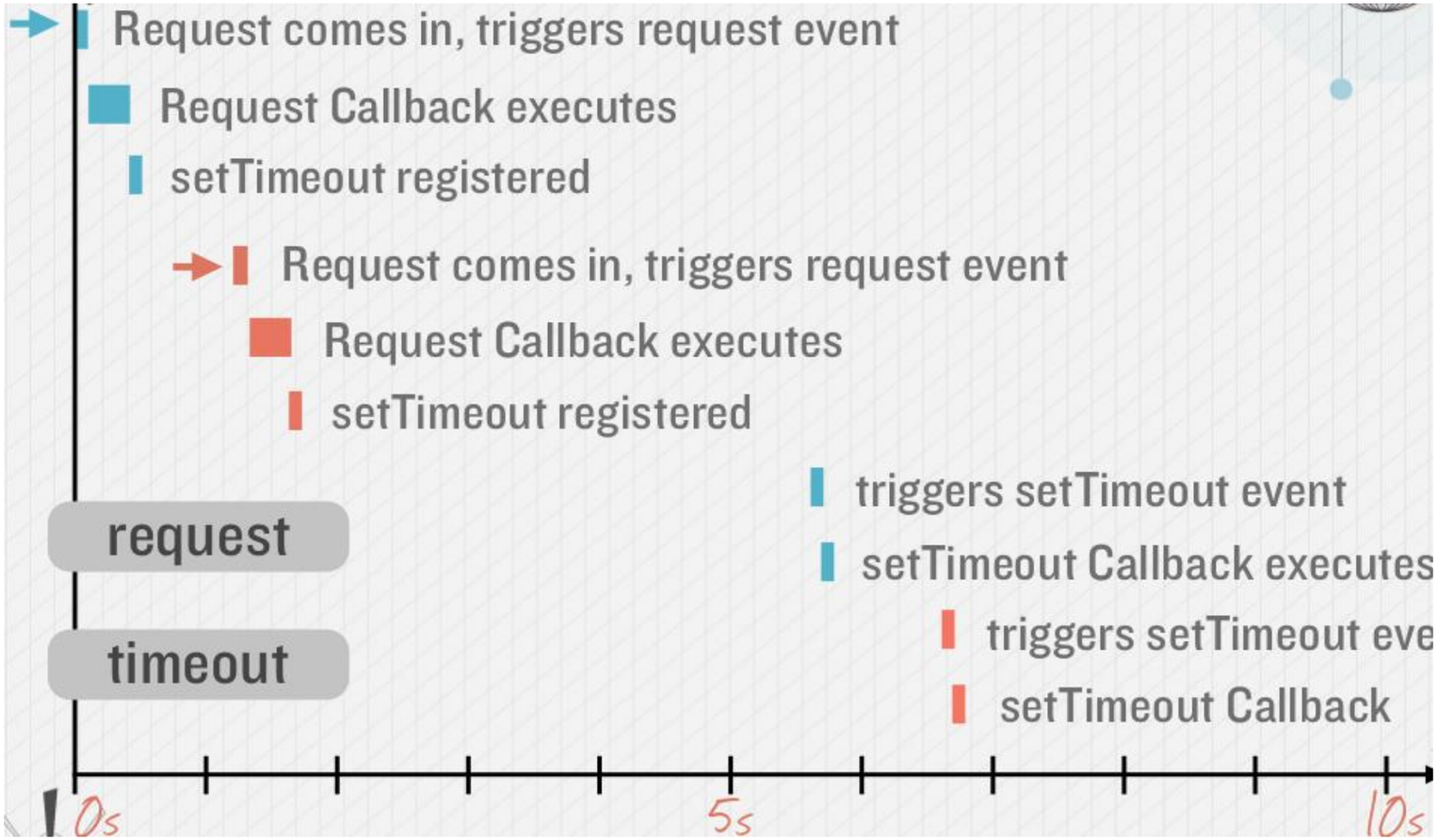
Example – Hello/Goodbye Callback

“Request” Callback

“Timeout” Callback

Callback Timeline, Non Blocking

Timing example: 2 requests to web application (indicated by red and blue in diagram)



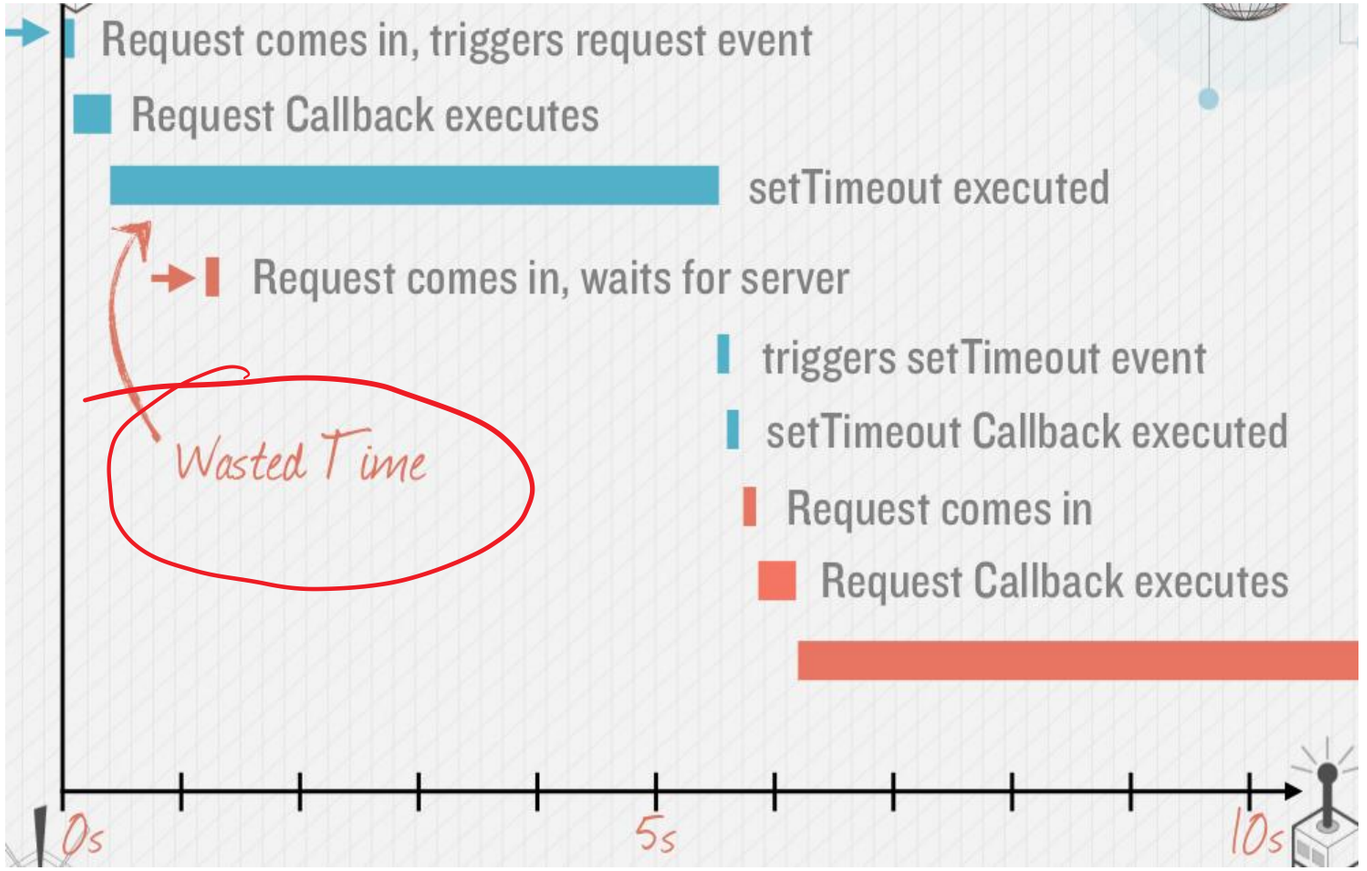
Avoid Blocking Calls in Node.js apps

- setTimeout in previous slide is an example of an asynchronous, non-blocking call.
- Avoid potential blocking/ synchronous calls
- **Activity likely to be blocking should be called asynchronously.**

Examples:

- Calls to 3rd party Web Services
- Database queries
- Computationally expensive operations (image file processing)

What if setTimeout() blocked…



Node “Error First” Callbacks

The “error-first” callback (or “node-style callback”) is a standard convention for many Node.js callbacks.