Introduction to Node.js

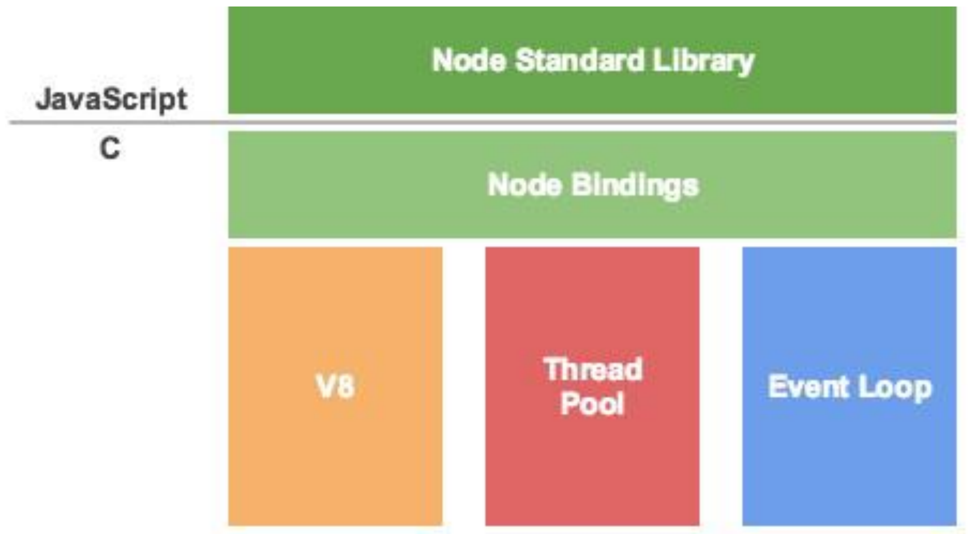
Frank Walsh
Diarmuid O'Connor

Agenda

- What is node.js
- The Dev Env for the Labs
- Event-based processes
- Callbacks in node
- 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…



NPM – the Package Manager

- Node has a small core API
- Most applications depend on third party modules
- Curated in online registry called the Node Package Manager system (NPM)
- NPM downloads and installs modules, placing them into a **node_modules** folder in your current folder.

NPM init

- You can use NPM to manage your node projects
- Run the following in the root folder of your app/project:

```
npm init
```

- This will ask you a bunch of questions, and then create a **package.json** for you.
- It attempts to make reasonable guesses about what you want things to be set to, and then writes a package.json file with the options you've selected.

NPX - the package runner

- Makes it easy to run a Node.js based executable that you would normally install via npm.
- Can use it at command line to execute packages, even if they are not previously installed.
- Very good for one-off commands/tests
- Comes with the latest versions of NPM
- The following example will execute the babel-node package to transpile and run index.js.

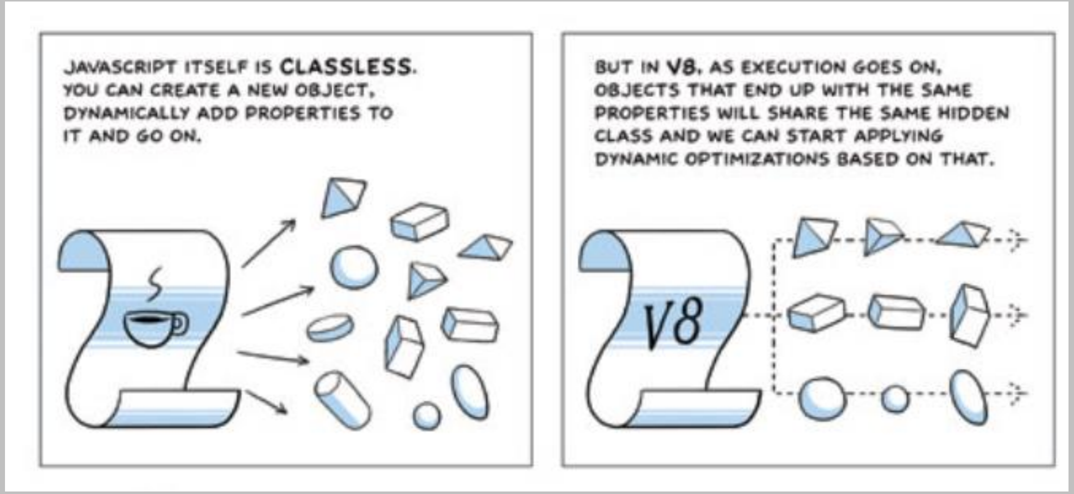
```
npx babel-node index.js
```

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

V8 JavaScript Engine


JAVASCRIPT ITSELF IS CLASSLESS. YOU CAN CREATE A NEW OBJECT, DYNAMICALLY ADD PROPERTIES TO IT AND GO ON.
BUT IN V8, AS EXECUTION GOES ON, OBJECTS THAT END UP WITH THE SAME PROPERTIES WILL SHARE THE SAME HIDDEN CLASS AND WE CAN START APPLYING DYNAMIC OPTIMIZATIONS BASED ON THAT.
V8

Node Development Environment

Development Environment Setup for Labs

Node.js:

We just talked about it

Babel:

Allow us to use up-to-date Javascript features, according to ECMAScript Standardisation

Nodemon:

monitor for any changes in your source and automatically restart your node app.

ESLint:

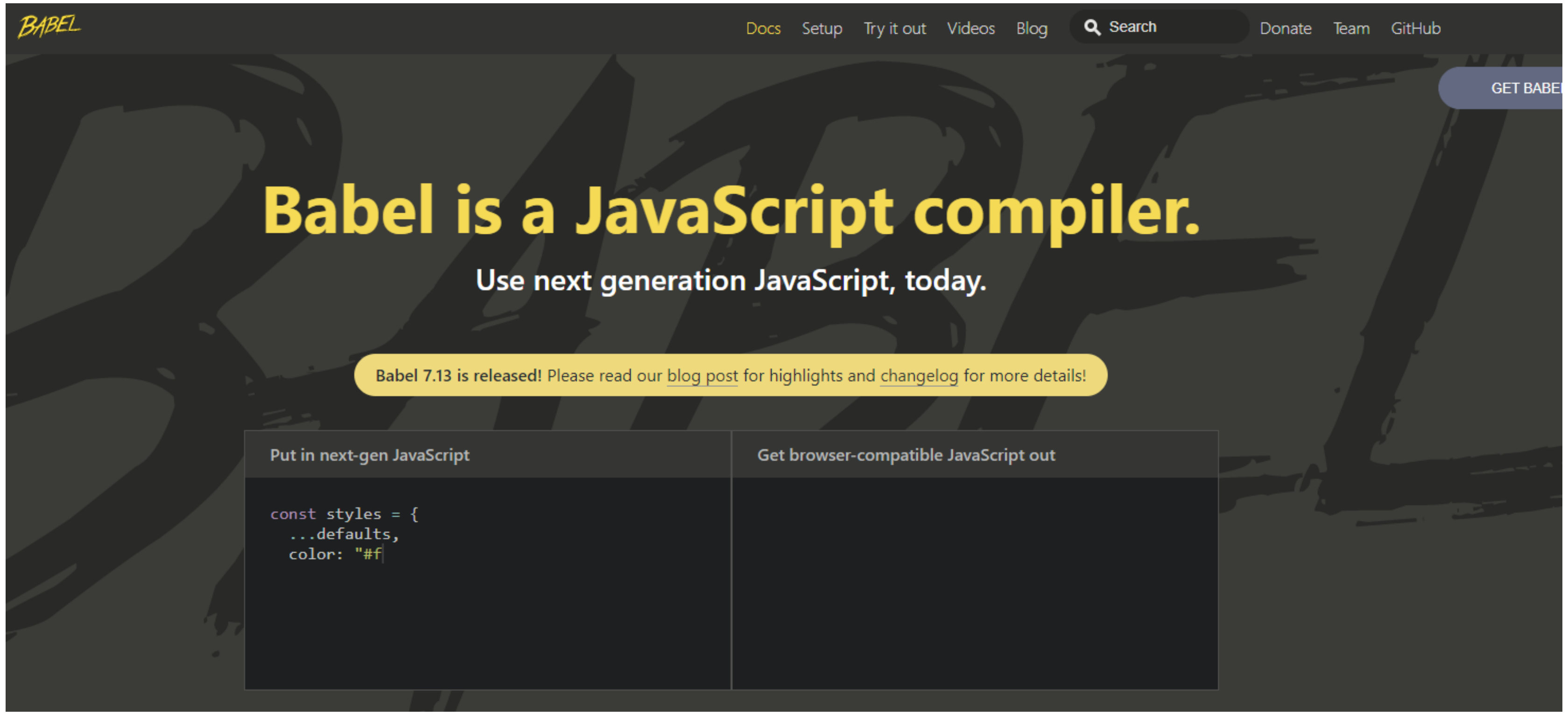
Find, report and fix problems in your javascript

Testing:

Manual: Postman

Automated: Mocha, Should, Signon (more later…)

Babel



Babel is a JavaScript compiler.

Use next generation JavaScript, today.

Babel 7.13 is released! Please read our blog post for highlights and changelog for more details!

Put in next-gen JavaScript

```
const styles = {
  ...defaults,
  color: "#f
```

Get browser-compatible JavaScript out

Node.js and Babel

We’re using ES6+ syntax for front end development

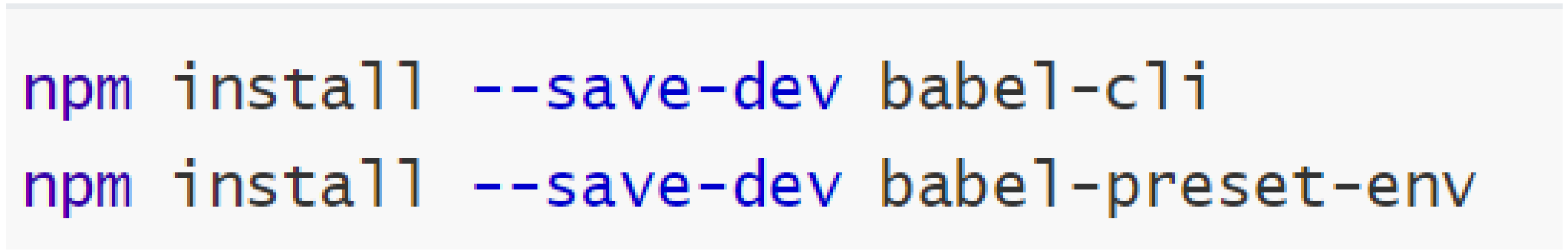
E.g. imports, spread operator, arrow functions, export default

Node.js does not support all of the latest and greatest ES6+ features

Solution:

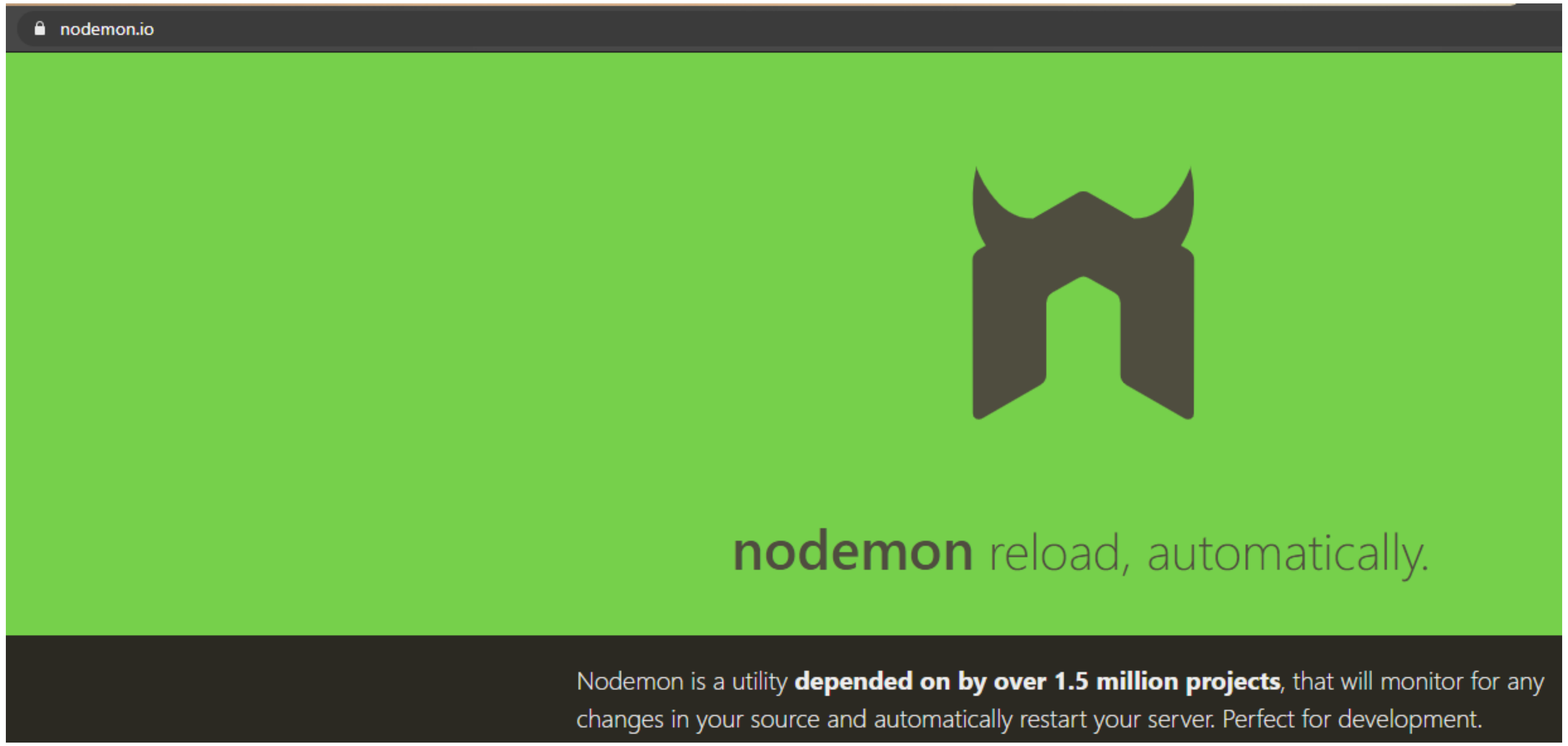
Use Babel to “Transpile” code from ES6+ to ES5 before we run it

We will install as **Development Dependency** for our project



```
npm install --save-dev babel-cli
npm install --save-dev babel-preset-env
```

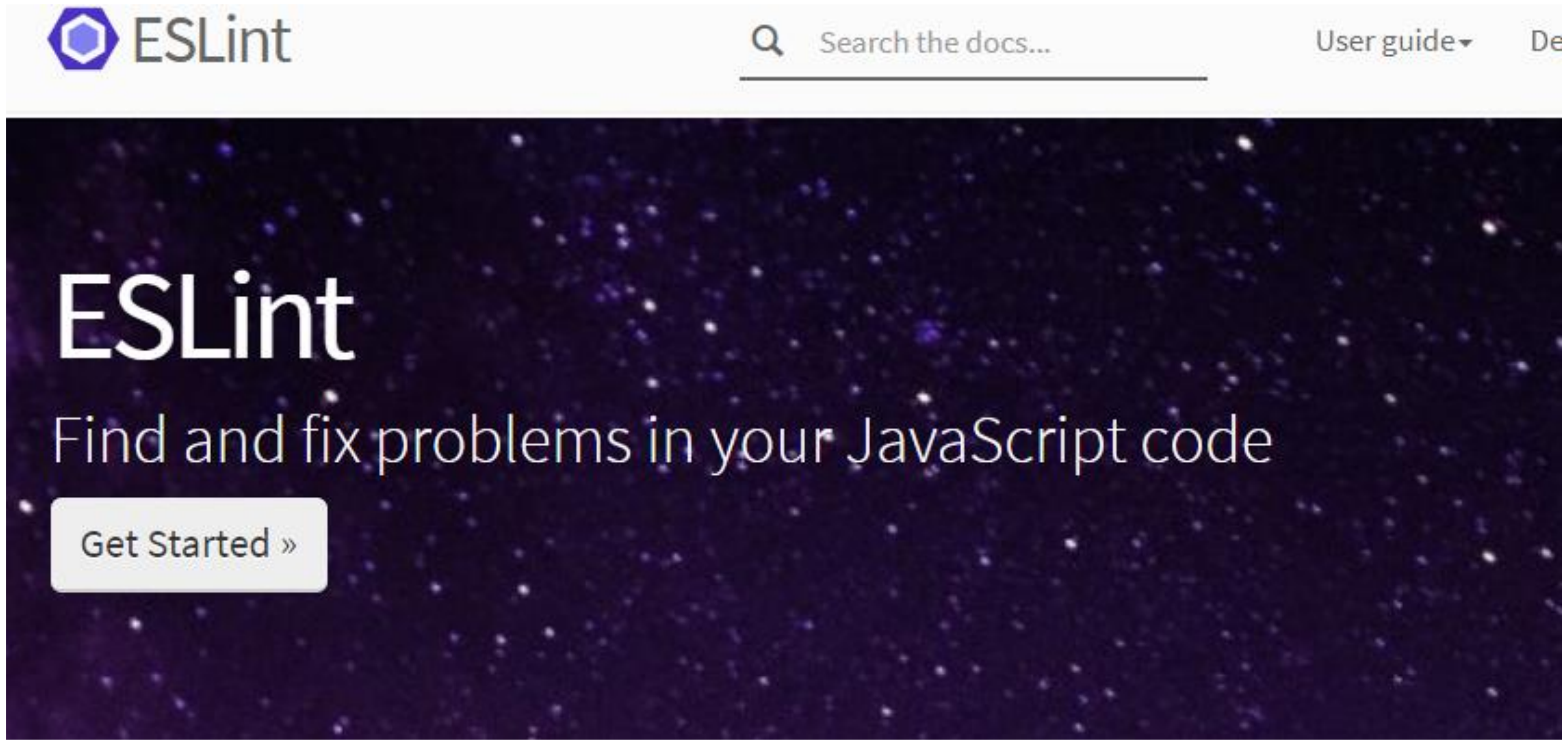
Nodemon



nodemon reload, automatically.

Nodemon is a utility **depended on by over 1.5 million projects**, that will monitor for any changes in your source and automatically restart your server. Perfect for development.

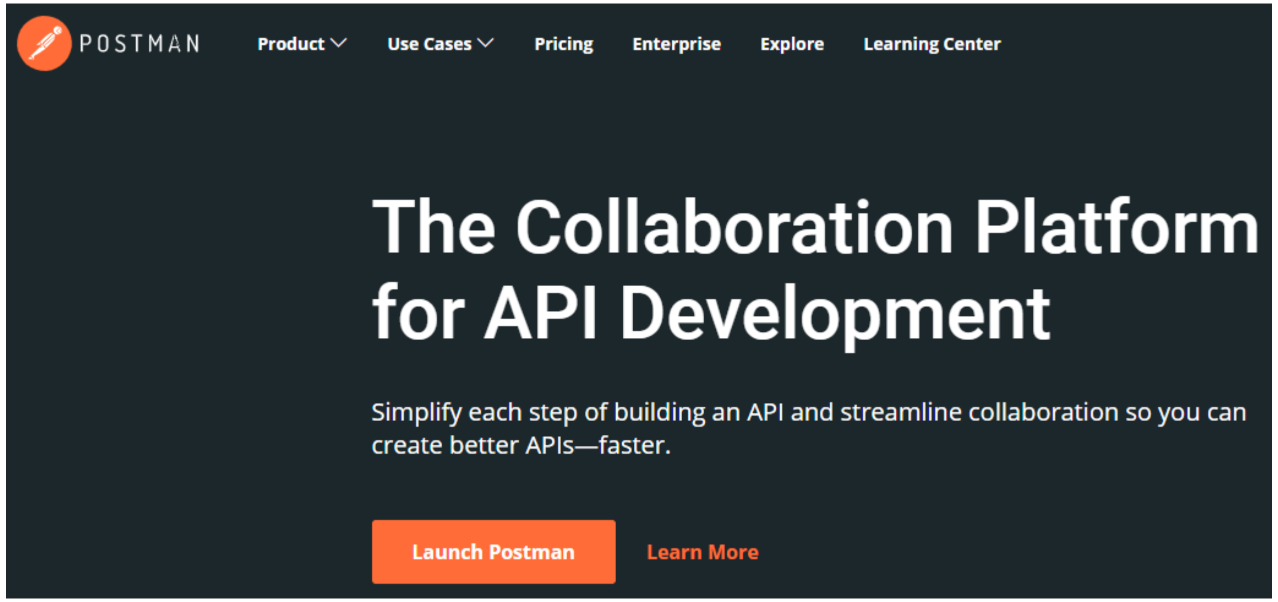
ESLint



ESLint

Find and fix problems in your JavaScript code

Testing



The Collaboration Platform for API Development

Simplify each step of building an API and streamline collaboration so you can create better APIs—faster.

simple, flexible, fun

Mocha is a feature-rich JavaScript test framework running on Node.js and in the browser,

What is Node.js: Event-based

- 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)

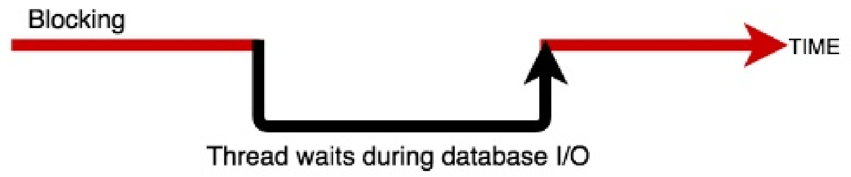
Source: https://gist.github.com/jboner/2841832

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

Blocking/Non-blocking Example

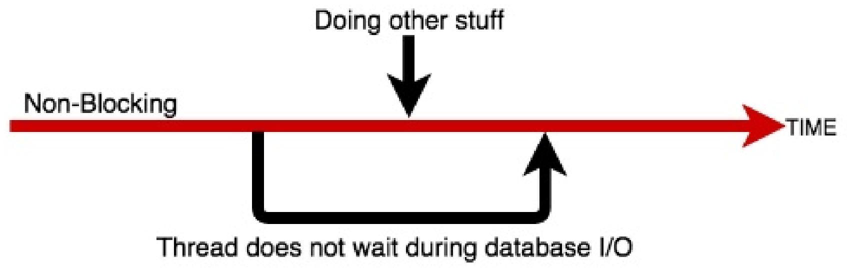
Blocking

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



Non-blocking

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



Blocking/Non-blocking example: 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
```

callback

Non-blocking

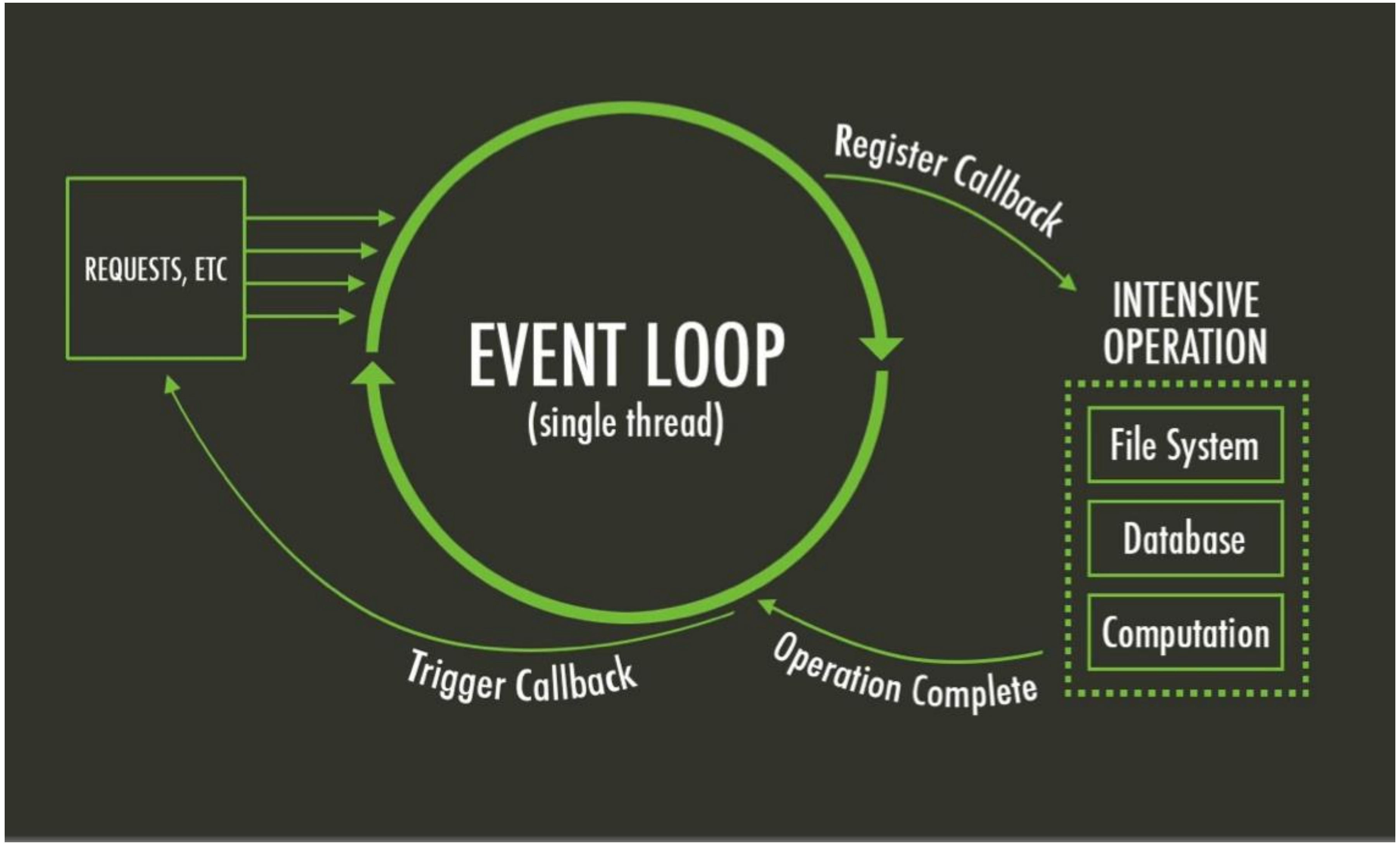
```
import fs from 'fs’;
fs.readFile('./text.txt’,’uft8’, (err, contents) => {
    console.log(contents);
});
console.log('Doing something else');
```

Console output

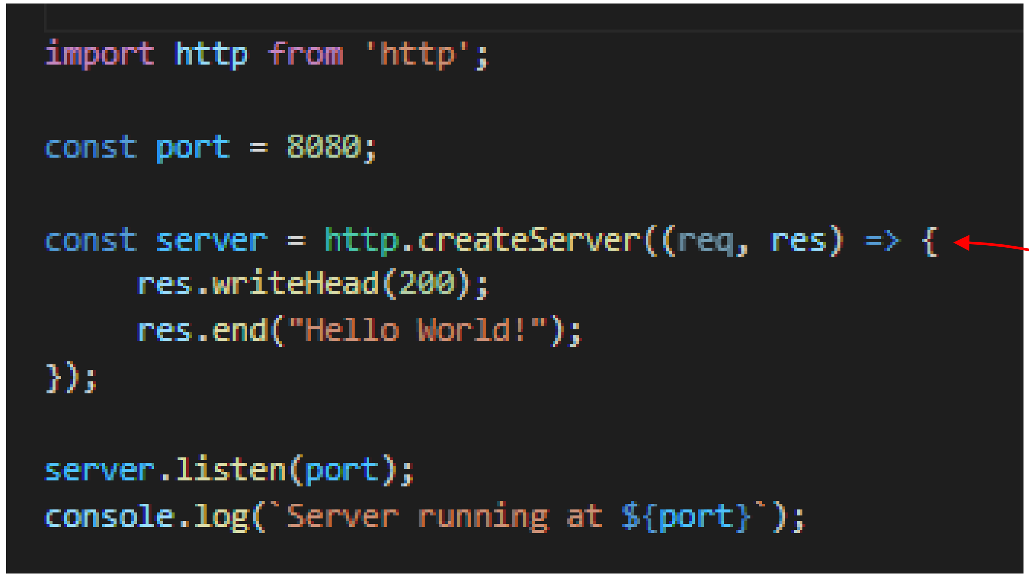
```
Doing something else
Hello World ……
```

The Node Event Loop and Callbacks

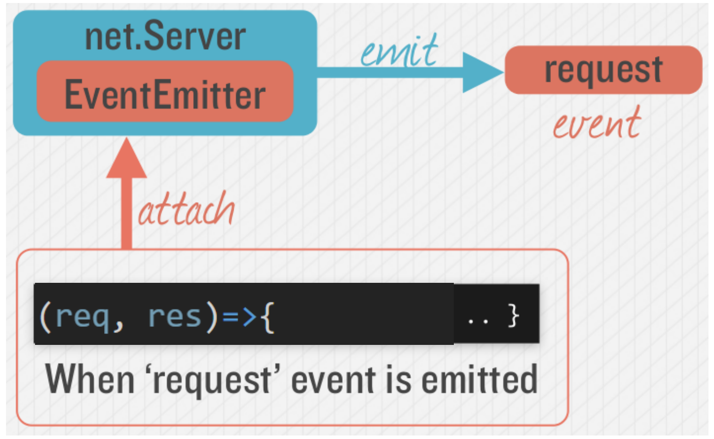
- A **Callback** is a function called at the completion of a given task. This prevents any blocking, and allows other code to be run in the meantime
- The Event Loop checks for known events, registers Callbacks and triggers callback on completion of operation
- More info here: https://developer.ibm.com/tutorials/learn-nodejs-the-event-loop/

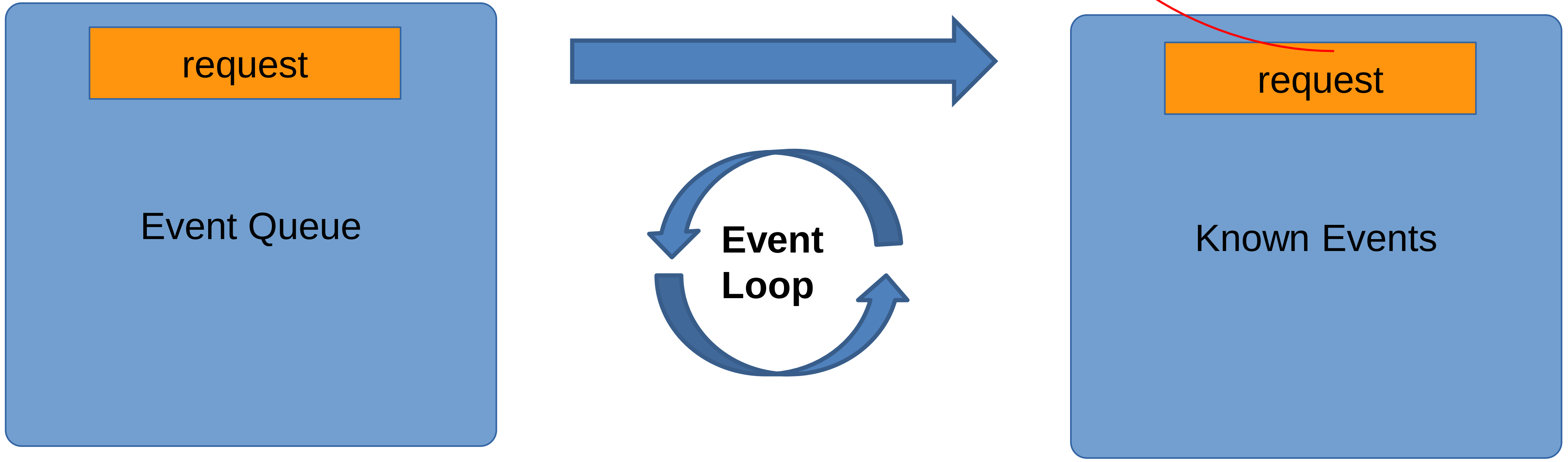


Node.js - Simple HTTP Server



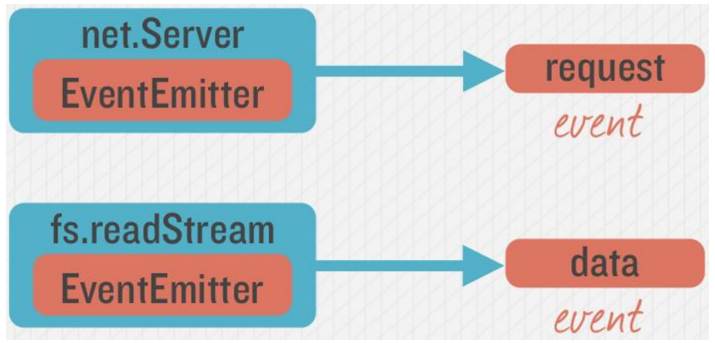
```
import http from 'http';

const port = 8080;

const server = http.createServer((req, res) => {
      res.writeHead(200);
      res.end("Hello World!");
});

server.listen(port);
console.log(`Server running at ${port}`);
```



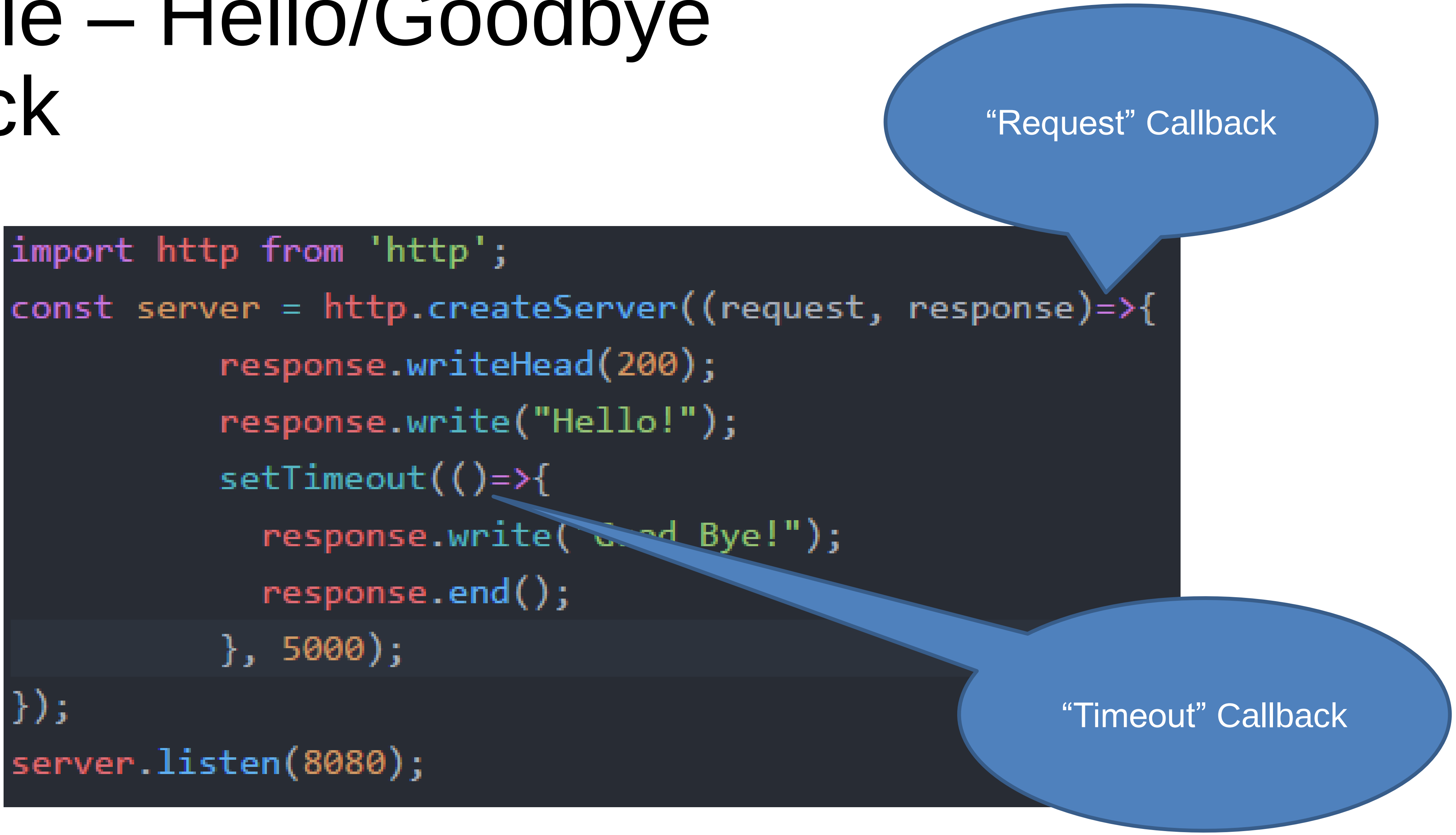


Emitting Event in Node

Many objects can emit events in node.



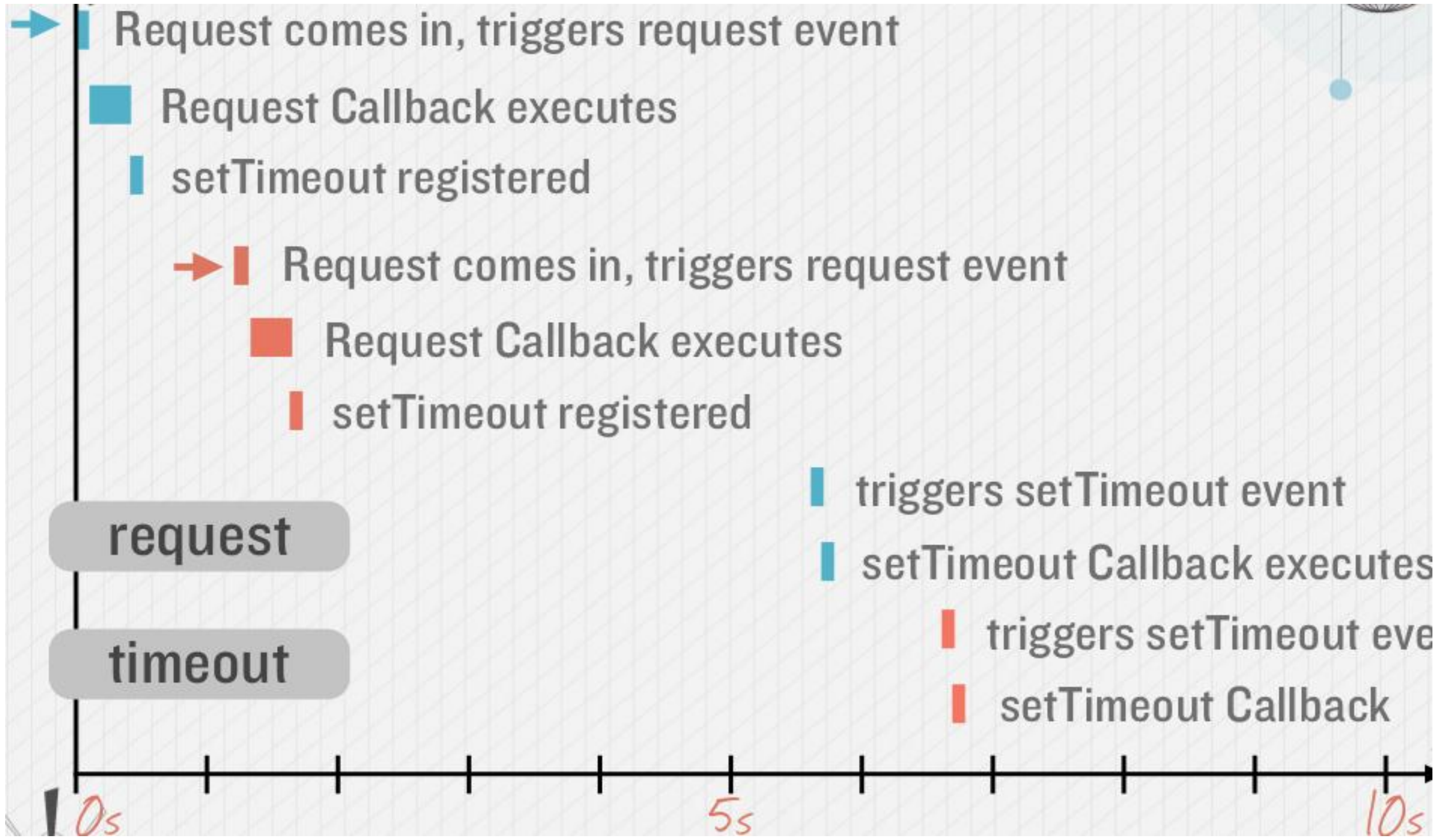
Example – Hello/Goodbye Callback



```
import http from 'http';
const server = http.createServer((request, response)=>{
          response.writeHead(200);
          response.write("Hello!");
          setTimeout(()=>{
            response.write("Good Bye!");
            response.end();
          }, 5000);
});
server.listen(8080);
```

Callback Timeline, Non Blocking

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



Request comes in, triggers request event
Request Callback executes
setTimeout registered
Request comes in, triggers request event
Request Callback executes
setTimeout registered
triggers setTimeout event
setTimeout Callback executes
triggers setTimeout eve
setTimeout Callback
request
timeout
0s
5s
10s

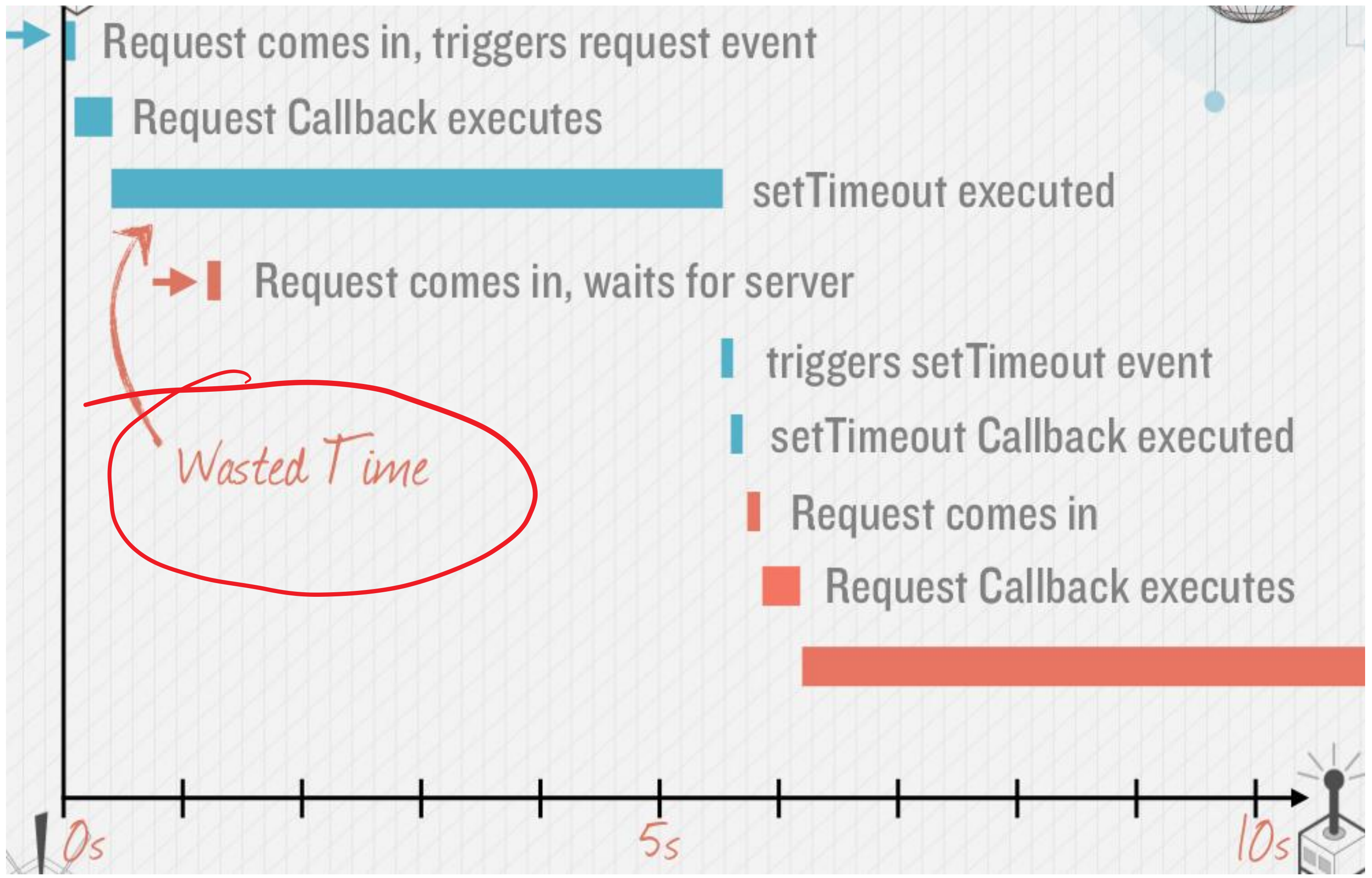
Avoid Blocking Calls in Node.js apps

- 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)

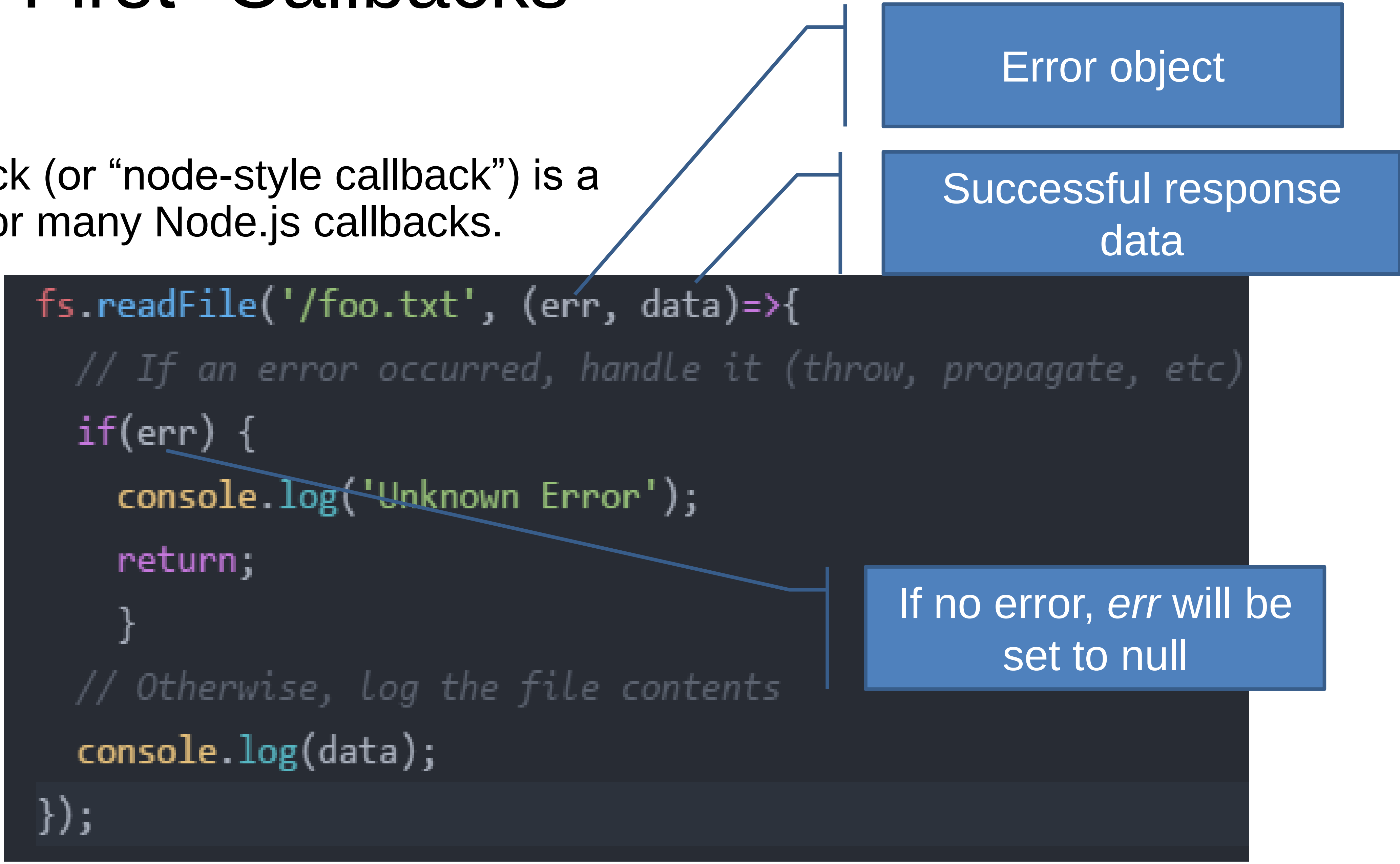
What if setTimeout() blocked…



Request comes in, triggers request event
Request Callback executes
setTimeout executed
Request comes in, waits for server
triggers setTimeout event
setTimeout Callback executed
Request comes in
Request Callback executes
Wasted Time
0s
5s
10s

Node “Error First” Callbacks

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



```
fs.readFile('/foo.txt', (err, data)=>{
  // If an error occurred, handle it (throw, propagate, etc)
  if(err) {
    console.log('Unknown Error');
    return;
    }
  // Otherwise, log the file contents
  console.log(data);
});
```

Node Modules

Node Modules

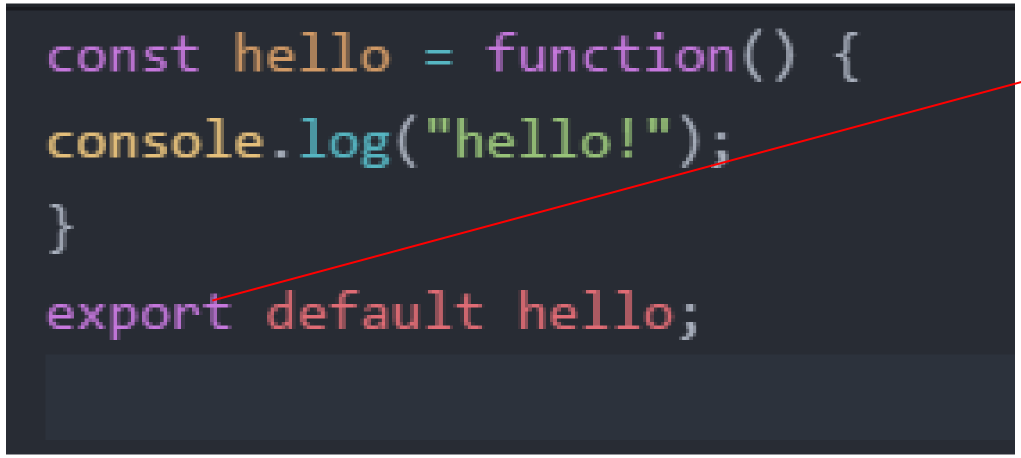
- To install NPM modules, navigate to the application folder and run “npm install”. For example :
- **npm install express --save**
- This installs into a “**node_module**” folder in the current folder.
- The **--save** bit updates your **package.json** with the dependency
- To use the module in your code, use:
- **import express from 'express';**
- This loads express from local **node_modules** folder.

NPM Common Commands

Common npm commands:
– **npm init** *initialize a package.json file*
– **npm install <package name> -g** *install a package, if –g option is given package will be installed globally,* ***--save*** *and* ***--save-dev*** *will add package to your dependencies*
– **npm install** *install packages listed in package.json*
– **npm ls –g** *listed local packages (without –g) or global packages (with –g)*
– **npm update <package name>** *update a package*

Creating your own Node Modules

- We want to create the following module called **custom_hello.js:**



```
const hello = function() {
console.log("hello!");
}
export default hello;
```

Export defines what import returns

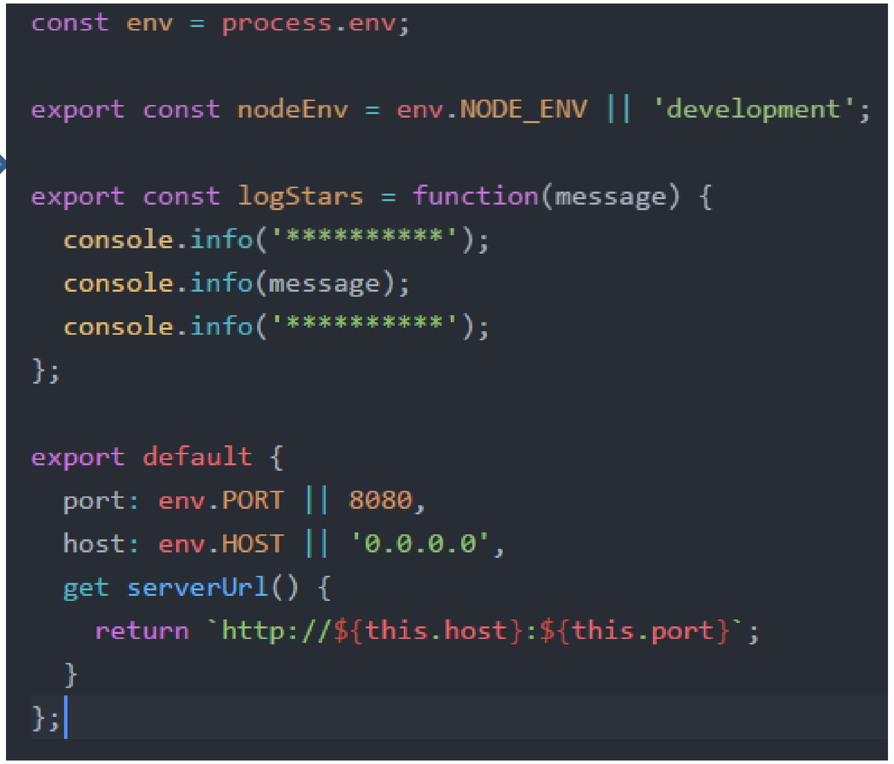
- To access in our application, **index.js:**

```
import hello from './custom_hello';
hello();
```

Creating your own Node Modules

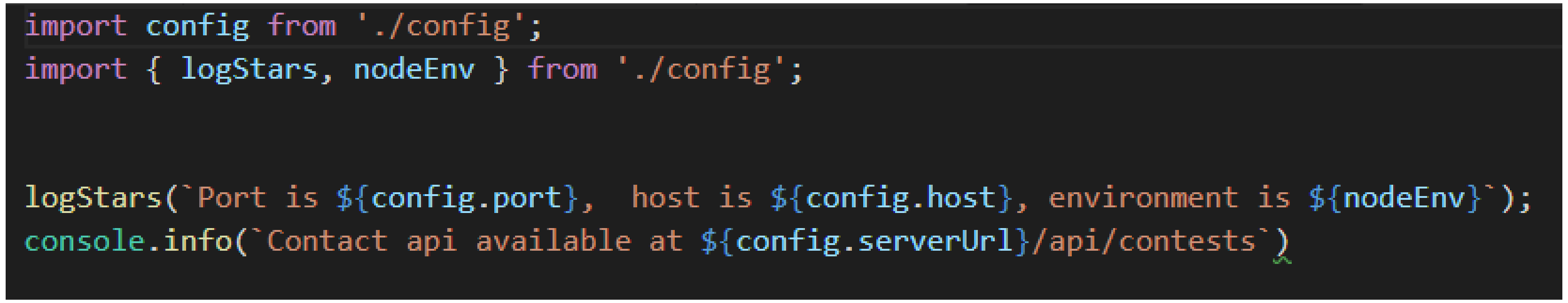
- Exporting Multiple Properties

Config.js



```
const env = process.env;

export const nodeEnv = env.NODE_ENV || 'development';

export const logStars = function(message) {
  console.info('**********');
  console.info(message);
  console.info('**********');
};

export default {
  port: env.PORT || 8080,
  host: env.HOST || '0.0.0.0',
  get serverUrl() {
    return `http://${this.host}:${this.port}`;
  }
};
```

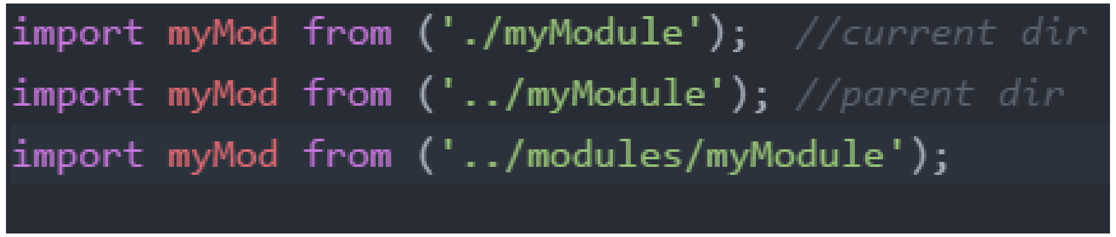
- Accessing in other scripts



```
import config from './config';
import { logStars, nodeEnv } from './config';


logStars(`Port is ${config.port},  host is ${config.host}, environment is ${nodeEnv}`);
console.info(`Contact api available at ${config.serverUrl}/api/contests`)
```

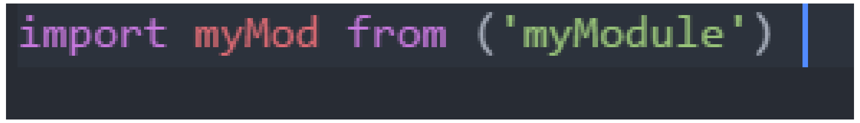
The import search

- Import searches for modules based on path specified:



```
import myMod from ('./myModule');  //current dir
import myMod from ('../myModule'); //parent dir
import myMod from ('../modules/myModule');
```

- Just providing the module name will search in **node_modules** folder



```
import myMod from ('myModule')
```