CREATIVE
CODE.IO

nodeJS

Introduction

Neal Riley

Beats Music - Pipeline Engineer

nodeJS

Course Format

DAY 1	DAY 2	DAY 3
Javascript	Node for Web	WebSockets
Running Node	Modules	RESTful APIs
Chat Server	Web Server	API Driven WebApp

PROJECTS

Installing NodeJS

- Download It!
    - http://nodejs.org/download/
- Package Manager
    - apt-get, yum, brew, etc.
- Install from Source
    - https://github.com/joyent/node/wiki/Installing-Node.js-via-package-manager

What is Node?

Javascript for Servers

What is Node?

nodeJS

Joyent®

Who is using Node?

PayPal™

UBER

@WalmartLabs

No Really… What is Node?

Google's V8 Engine

A Little Bit of Magic

- Client-Side
- window Global Object

- Server-Side
- process Global Object

How’s Everyone’s Javascript?

- Objects
- Functions
- Truth-y/False-y
- window, console
- setTimeout, setInterval

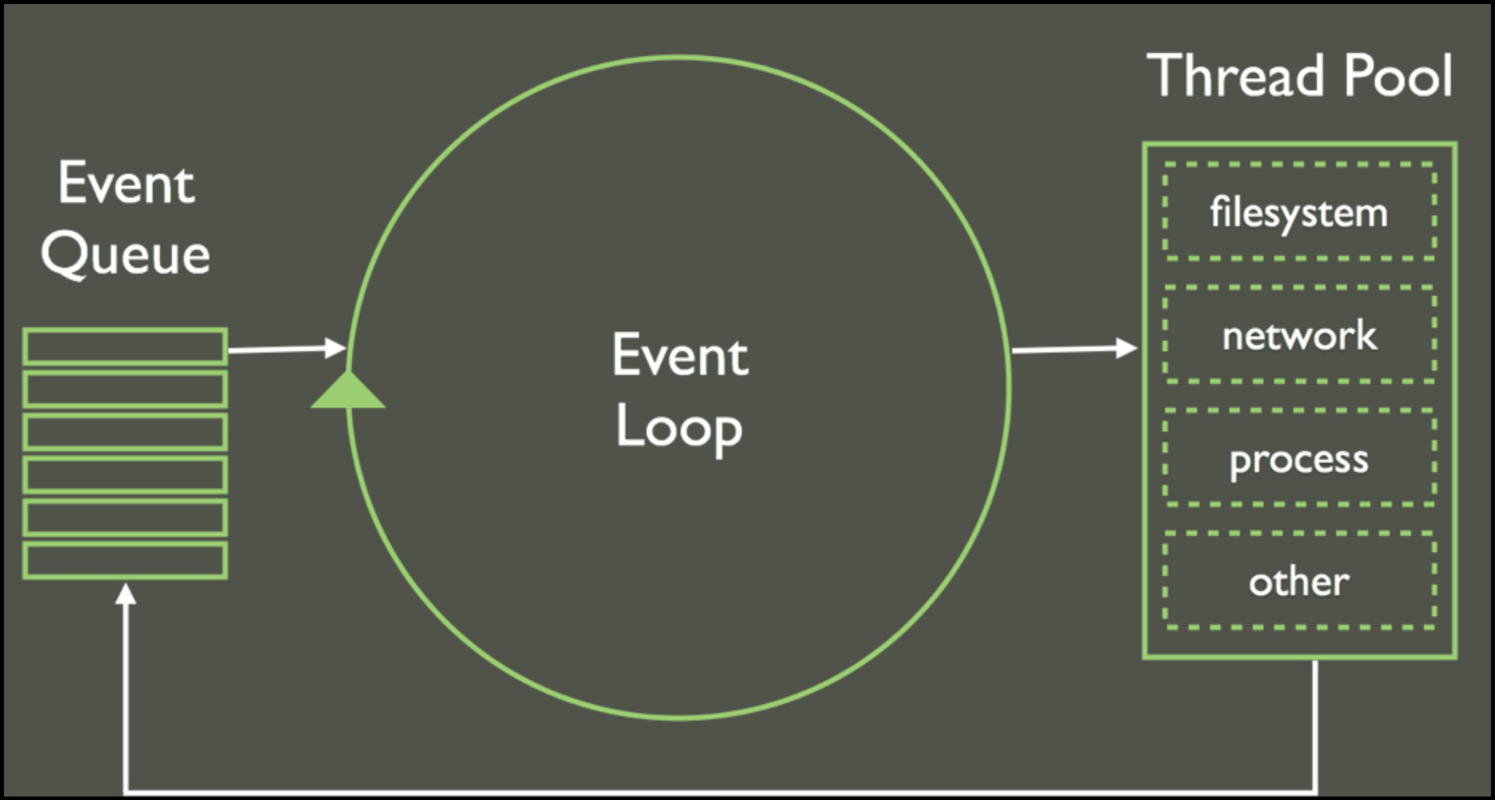
How’s Everyone’s Javascript?

- Objects == Functions
- Function level scope
- Closures
- Callbacks
- Function Chaining

Event Loops

- A good example of methods that use the Event Loops in the browser and node are setTimeout and setInterval
- setTimeout and setInterval aren’t part of Javascript
- Many environments where Javascript runs provides an event loop, where methods like setTimeout and setInterval can function

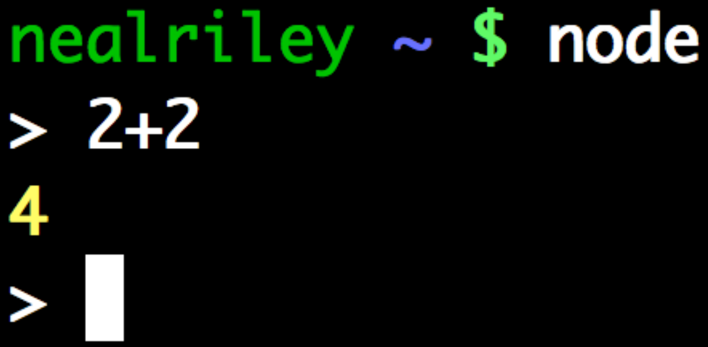
Event Loops


Event Queue
Event Loop
Thread Pool
filesystem
network
process
other

Command-Line Node

nodeJS is run from the command-line, you will need a terminal or shell to run it.

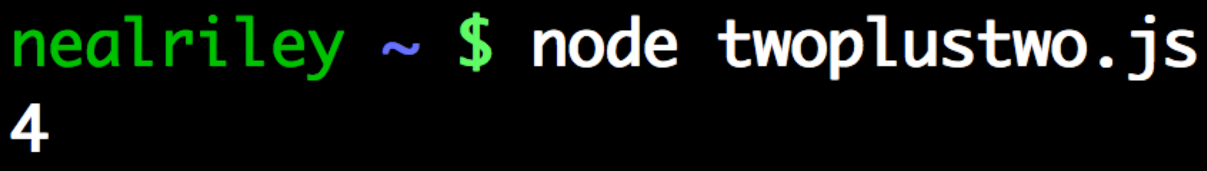
Running node is as easy as typing node:



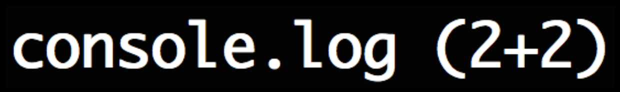
```
nealriley ~ $ node
> 2+2
4
> 
```

Node From a File

By adding a filename as node's first arguement, node will call a file instead of opening the node shell.



```
nealriley ~ $ node twoplustwo.js
4
```

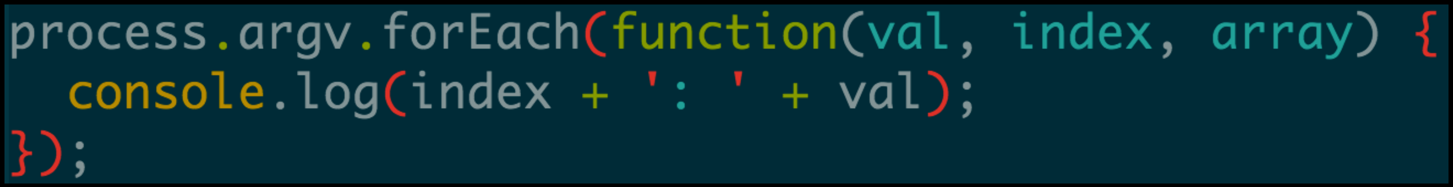


```
console.log (2+2)
```

Global Objects - Process

Node provides a few global objects that help your node application interact with its environment

One such object is **process**, which provides the **argv** object



```
process.argv.forEach(function(val, index, array) {
  console.log(index + ': ' + val);
});
```

Command-Line Arguments

You are able to pass arguments to your node application. Arguments are made available in the **process** global object

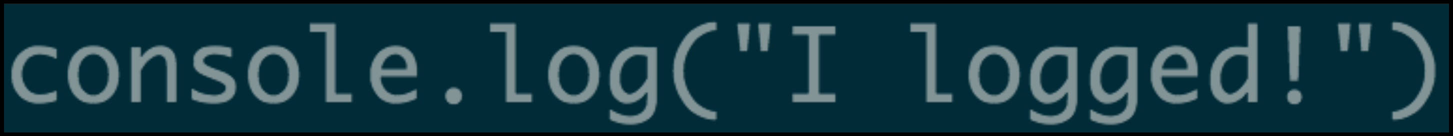
```
nealriley ~ $ node filename.js arg1 arg2
```

```
process.argv.forEach(function(val, index, array) {
  console.log(index + ': ' + val);
});
```

Global Objects - Console

Another object is console, which provides the “log” method, which prints text to the console’s stdout.

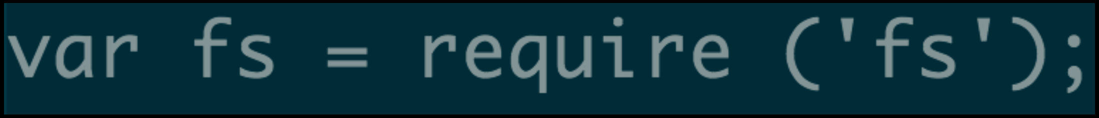
This is similar to how logging works in a web browser



```
console.log("I logged!")
```

Global Objects - require

- Node provides the “require” global function, which allows you to include modules provided by node or others



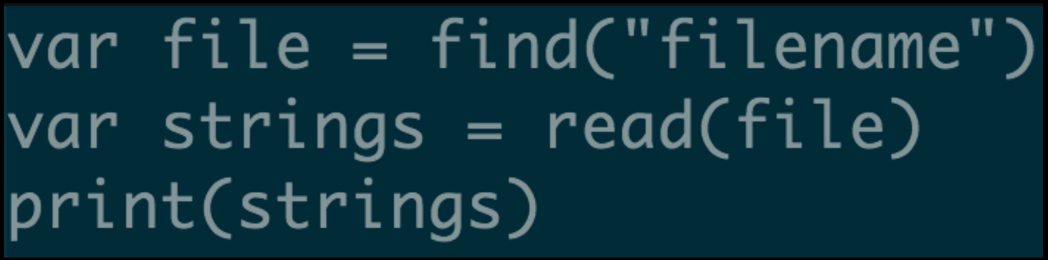
```
var fs = require ('fs');
```

Programming in Node is Weird

- Node is built at its core to be non-blocking
- Node libraries that require I/O follow two design patterns
    - Callback Method
    - Event Method

“Blocking” Pattern

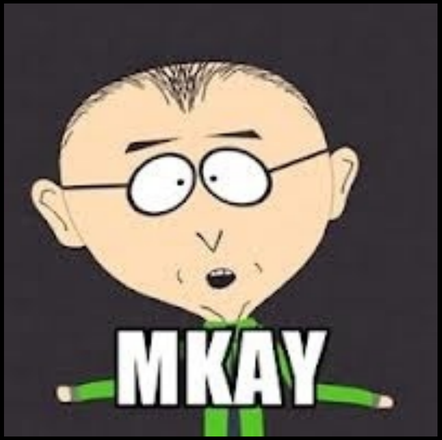
Consider the following example:



```
var file = find("filename")
var strings = read(file)
print(strings)
```

At each step in this code, we must wait until execution finished before moving onto the next step. This, in node, is referred to as “blocking”.

Blocking is Bad



Blocking is Bad

- Imagine what web servers do…
- If you blocked on every request, responses would take longer and longer to respond

Callback Method

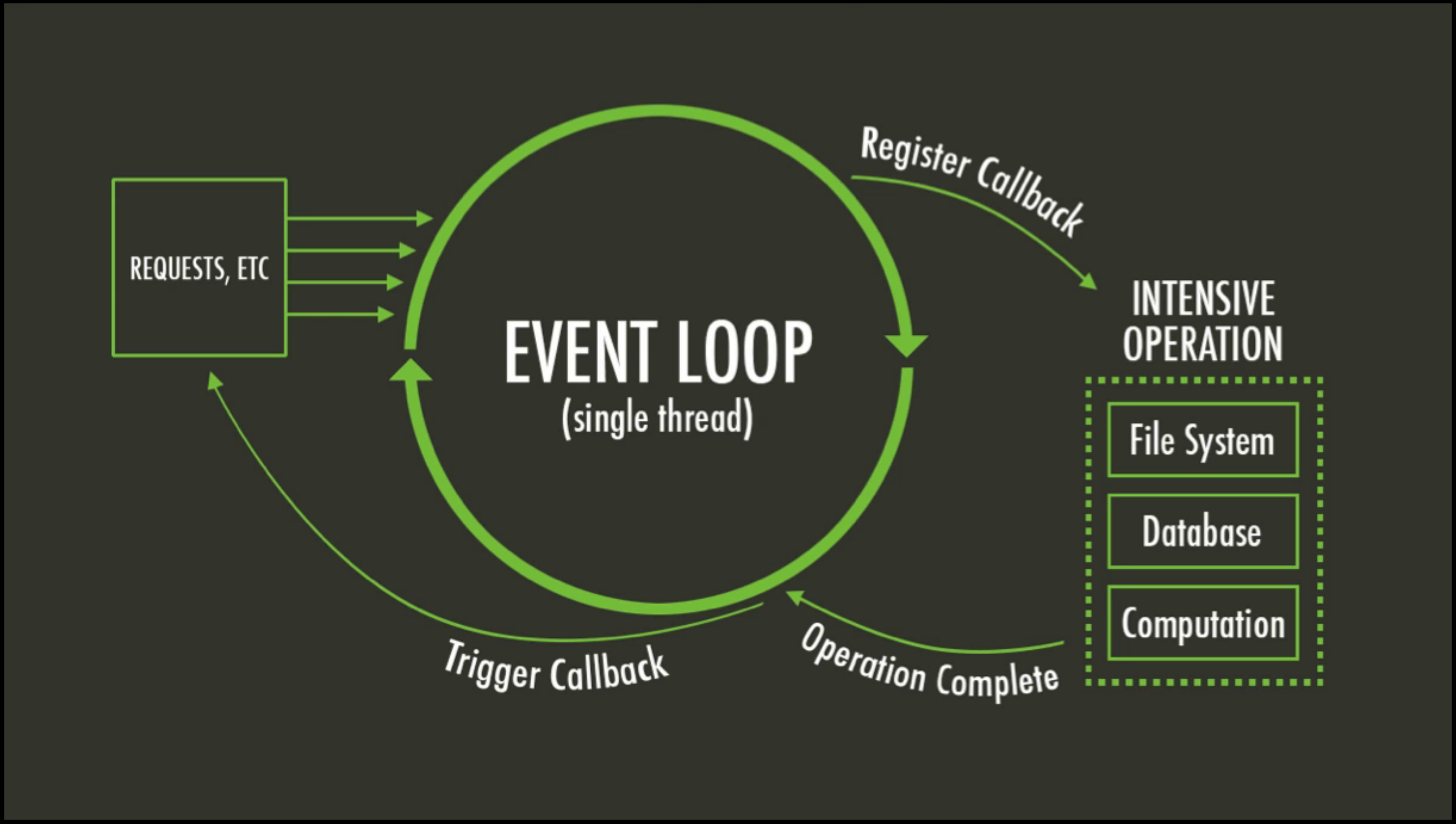
Looking at our previous blocking example:

```
var file = find("filename")
var strings = read(file)
print(strings)
```

We might write it like this:

```
var file = find("filename")
readStrings(file, result(strings) {
  print(strings)
})
```

Event Loops


REQUESTS, ETC
EVENT LOOP
(single thread)
Register Callback
INTENSIVE OPERATION
File System
Database
Computation
Operation Complete
Trigger Callback

Callback Method

- Code written in the Callback Method does not return a value directly
- Instead, a “callback” function is passed as an argument, which is called once the code completes, which contains any result inside it

What’s Good About Callbacks?

- Instead of waiting for a function to finish execution, you simply pass a function that executes once the code completes
- Because of this, node can continue executing the code in your application while “blocking” code operates in the background

What’s Bad About Callbacks?

- It takes some getting used to…
- “Callback Hell”
- While callbacks aren’t blocking, you must wait till the callback completes before values are returned

Event Emitter Model

- Also known as Pub/Sub (Publish and Subscribe)
- While also non-blocking, functions using the event model return an event emitter.
- When data is available for further use, and event is emitted which contains this data

Why use Event Emitters?

- I/O intensive functions which can return information in many stages are often better served by the Event Model, rather than the Callback Model
- Asynchronous interaction can be extended from node to clients by use of Sockets (which we will see more of later)

A Few Examples

Example:
Reading from a File

fs

fs.readFile

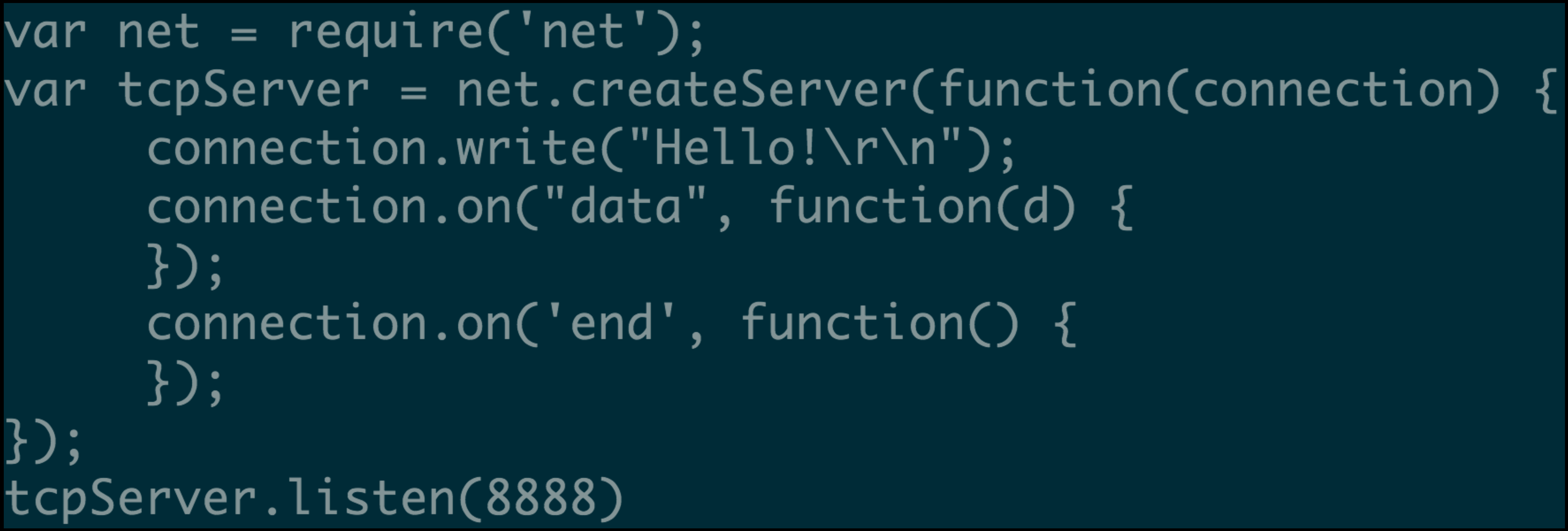
- filename
- function
    - error
    - data

Node for the Web

- Node comes bundled with two core libraries for communicating over a network

net **http**

TCP Server



```
var net = require('net');
var tcpServer = net.createServer(function(connection) {
     connection.write("Hello!\r\n");
     connection.on("data", function(d) {
     });
     connection.on('end', function() {
     });
});
tcpServer.listen(8888)
```

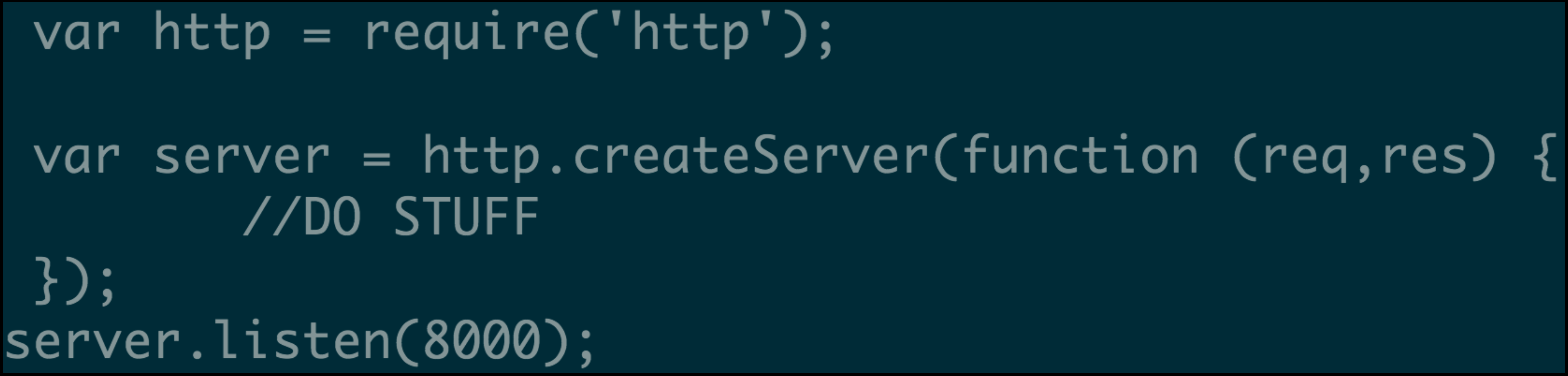
Streams

PROJECT
Notes with Friends

Notes with Friends

- Requirements
    - One Sender , One Receiver (Find a friend or two)
    - TCP Server, using the net library
    - Send a message to your friend
    - createServer , createClient, console.log

HTTP Server



```
 var http = require('http');

 var server = http.createServer(function (req,res) {
        //DO STUFF
 });
server.listen(8000);
```

PROJECT
Notes with Friends With Web

Notes with Friends With Web

- This time, we will post your friend’s message to a web server
- We will chain together a TCP Server and HTTP Server