Node js Development

By: Ahmed El-Mahdy.

FB: bit.ly/2Utbjkz

Lkd: bitlylink.com/ooLxn

Content of Course

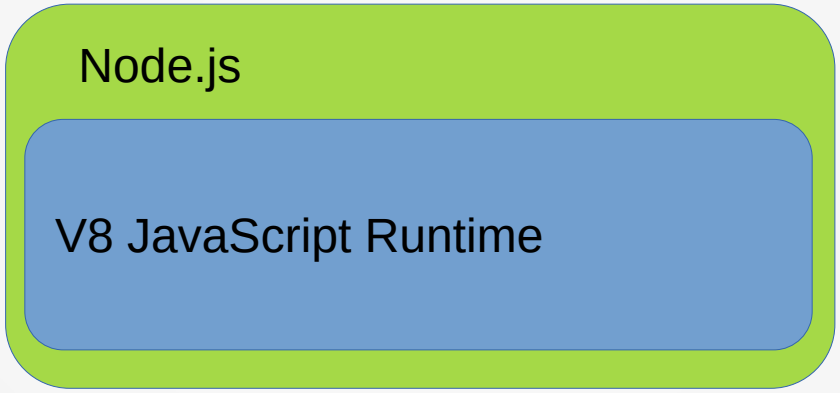
ITI

- Create server
- Read files
- Express
- Socket.io
- Database (out of scope)

What is NODE.JS?

ITI

- Allows to you build scalable network applications Using JavaScript in server-side.



It’s Fast because its mostly C code

Introduction

ITI

- Node.js is a platform built on Chrome's JavaScript runtime for easily building fast, scalable network applications.
- e.g. video streaming, SPAs, networking apps.
- Node.js uses an event-driven, non-blocking I/O model that makes it lightweight and efficient, perfect for DIRT Applications that run across distributed devices. (data-intensive real-time applications)
- It’s a command line tool, that runs JavaScript.
- Node.js = Runtime Environment + JavaScript Library.
- Node.js' package ecosystem, NPM, is the largest ecosystem of open source libraries in the world.

Why Node.js

ITI

- Asynchronous and Event Driven
- Very Fast
- Single Threaded but highly Scalable
- Node.js uses a single threaded model with event
- looping.
- Open source

WHO IS USING NODE.JS

ITI

Environment Setup

ITI

- The source code written in source file is simply javascript.
- The Node.js interpreter will be used to interpret and execute your javascript code.

Do & Don't With Node.js

ITI

You can :

- create an HTTP server.
- create a TCP server similar to HTTP server.
- create a Web Chat Application.
- creating online games, collaboration tools or anything which sends updates to the user in real-time.

You can’t :

- Node is a platform for writing JavaScript applications outside web browsers. This is not the JavaScript we are familiar with in web browsers. There is no DOM built into Node, nor any other browser capability.
- Node can’t run on GUI, but run on terminal/cmd

Installation on Ubuntu

ITI

Step 1: Add Node.js PPA

- **To add the repository, run the commands below**

```
sudo apt install curl
```

- **for the Latest release, add this PPA.**

```
curl -sL https://deb.nodesource.com/setup_10.x | sudo bash -
```

- **To install the LTS release, use this PPA.**

```
curl -sL https://deb.nodesource.com/setup_8.x | sudo bash -
```

Step 2: Install Node.js and NPM

- **To install, run the commands below**

```
sudo apt install nodejs
```

- **use the commands below to view the version number installed**

```
node -v
npm -v
```

Installation on Windows

ITI

Step 1:

Go to the site https://nodejs.org/en/download/ and download the necessary binary files.

Step 2:

Double click on the downloaded .msi file to start the installation.

Make sure to select NPM package manager on the Custom Setup screen, not the default of Node.js runtime. This way we'll install Node and NPM at the same time.

Let’s start with node.js

NODE .JS FACTS

ITI

- Node.js makes communication between client and server will happen in same language
- Node.js makes use of event-loops via JavaScript's call back functionality to implement the non-blocking I/O.
- There is no DOM implementation provided by Node.js
- Everything inside Node.js runs in a single-thread.

Node.js is "Event" based server

Core node.js

ITI

NODE .JS ECOSYSTEM:

- Node.js heavily relies on modules in order to load built-in APIs, third party modules or custom local module.
- Module is a self contained series of one or more .js files presented by an object.
- Modules is where we can encapsulate related functionality into a single file.
- Should be accessible from outside the file.

Core node.js

ITI

NODE GLOBAL:

- console– allows printing to allows printing to allows printing to allows printing to stdout
- require– function to load a module function to load a module

Core node.js

ITI

Module:

A reusable block of code whose existence doesn't accidently impact other code.

JS didn't have this before.

- **Modules allow Node to be extended.**
- **Modules act as libraries**
- **We can include a module with the global require function, require('module');**
- **We can install helping module**
- **Node provides core modules that can be included by their name:**

  File System –require('fs') / Http –require('http') /Utilities –require('util')
- **We can create our own custom module**

Core node.js

ITI

CommonJS Modules :

An agreed upon standard for now code modules should be structured

Require function:

The function takes a path of module and return module(function, object)

Let’s refresh our memory

Type of function:

- Simple function (statement function)
- First class function
  - return value
  -Pram function
  -Return function
- Expression function
  -result in value

Javascript Side

ITI

Prototypal Inheritance and Function Constructors

Function Constructors:

- Normal function used to construct objects.

Prototypal Inheritance

- Object inherit directly from other objects.

Call by reference and Call by value

IIFE (Immediately invoked function expressions)

Events & The Event Emitter ITI

- Event is something that has happened in our App that we can respond to.

- In node we actually talk about two different kinds of events.

  -First kind System events come from side node.js

- Coming from system like finished reading file – received data from net (c++, libuv)

  -Another side Custom event (Javascript) event emitter

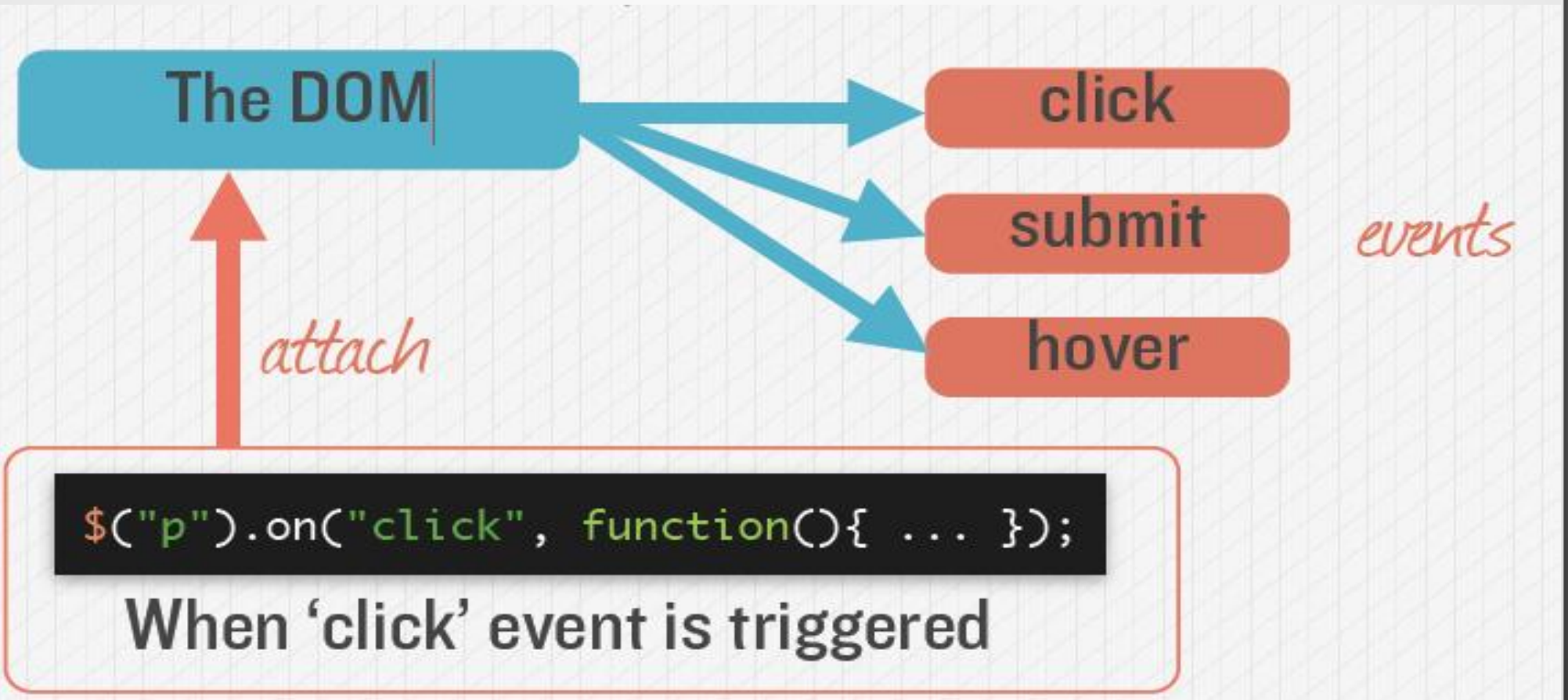
Events & The Event Emitter ITI

- All objects that emit events are instances of the EventEmitter class.
- These objects expose an eventEmitter.on() function that allows one or more Functions to be attached to named events emitted by the object.
- When the EventEmitter object emits an event, all of the Functions attached to that specific event are called synchronously
- Any object can become an EventEmitter through inheritance using:
  - -util.inherits() method.
  - -ES6 extends

EVENTS IN THE DOM

ITI

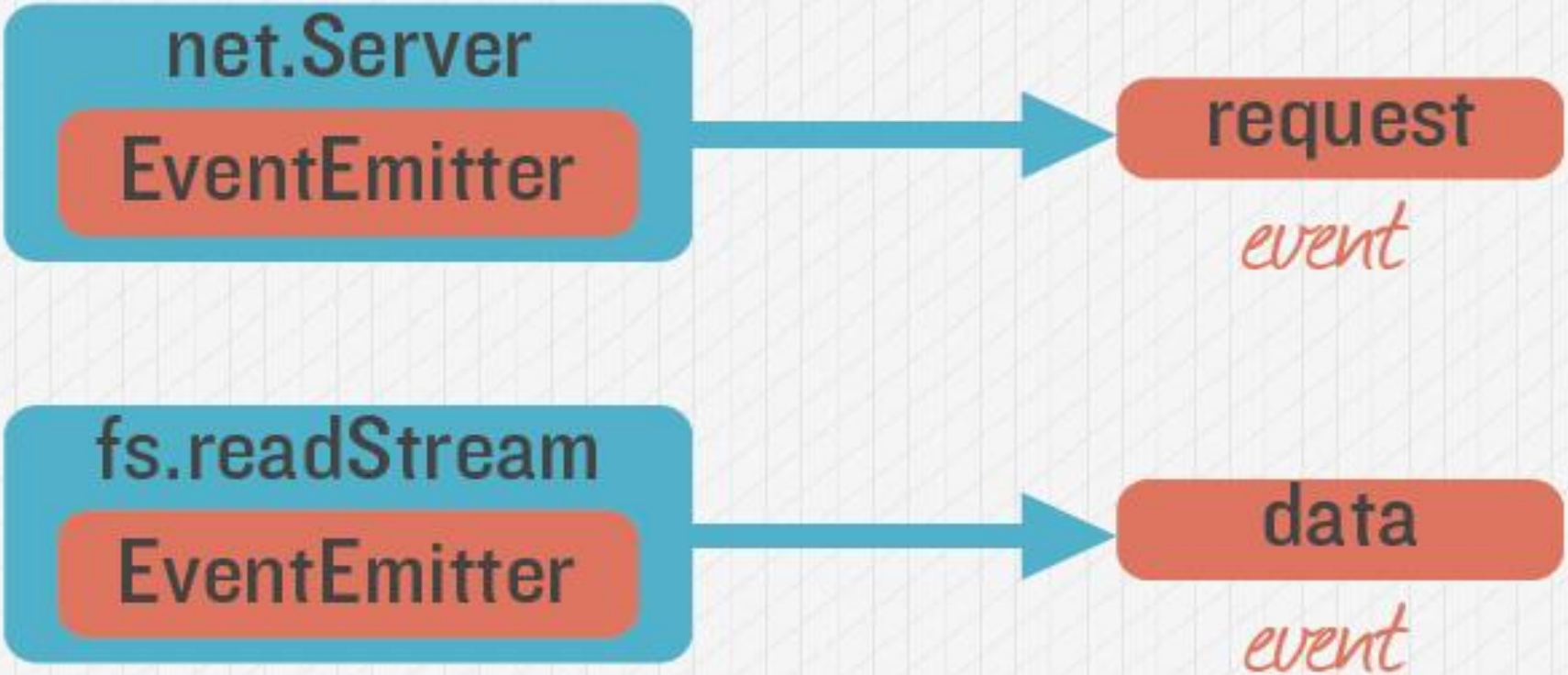
The DOM triggers events
you can listen for those events



EVENTS IN NODE

ITI

Many objects in Node emit events



CUSTOM EVENT EMITTERS

ITI

```
var EventEmitter = require('events').EventEmitter;
```

events

```
var logger = new EventEmitter();
```

error warn info

```
logger.on('error', function(message){
  console.log('ERR: ' + message);
});
```

listen for error event

```
logger.emit('error', 'Spilled Milk');
```

→ ERR: Spilled Milk

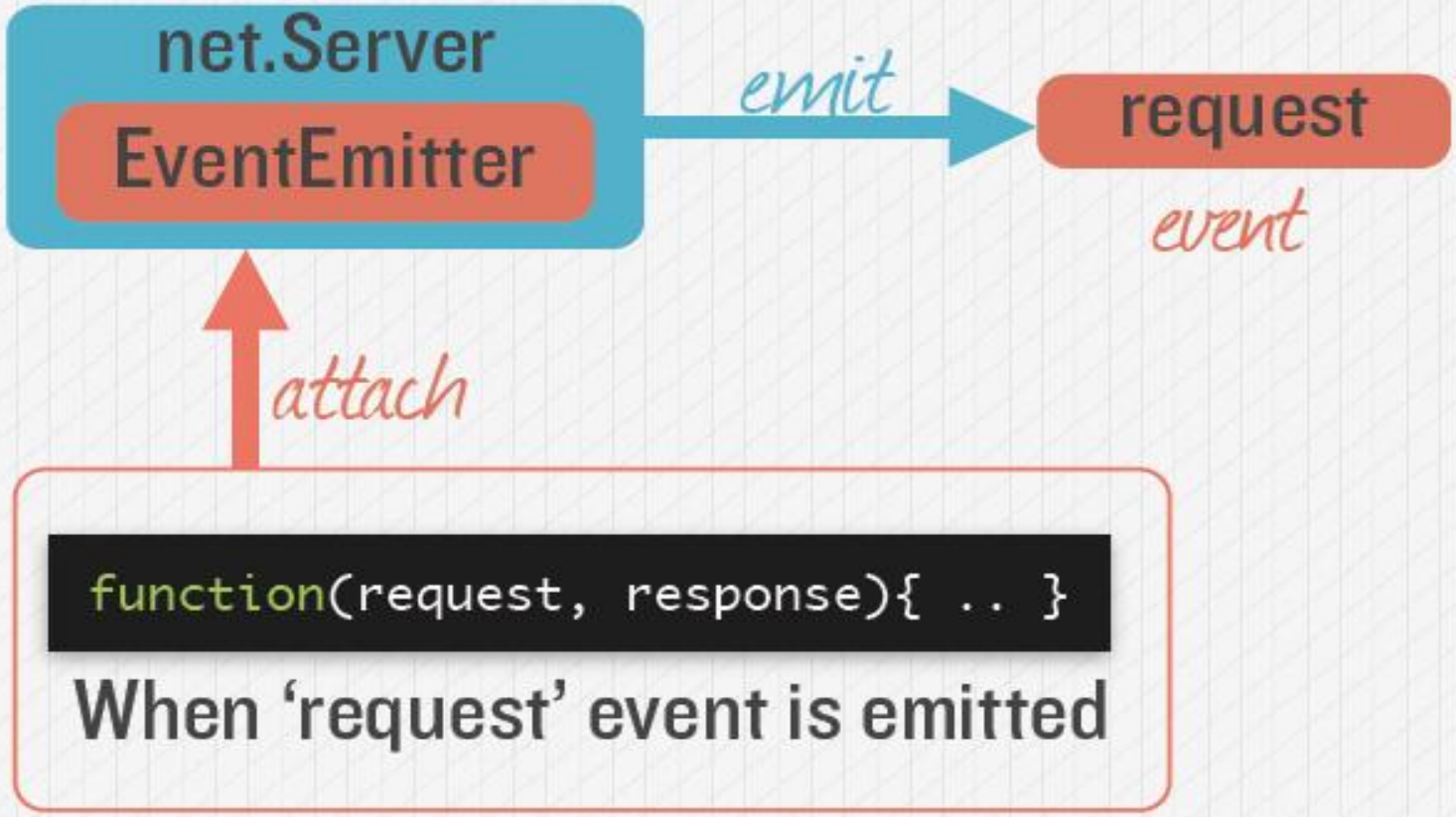
```
logger.emit('error', 'Eggs Cracked');
```

→ ERR: Eggs Cracked

EVENTS IN NODE

ITI

Many objects in Node emit events



Asynchronous code & Non-Blocking code ITI

- Asynchronous code
- Non-blocking code
- Event loop
- Libuv
- Call Backs

Asynchronous code & Non-Blocking code ITI

Asynchronous code :

- More than one process running simultaneously.
- Node dose things asynchronously V8 Doesn't.

Synchronous:

- one process executing at a time .

*Javascript is synchronous think of it as only one line of code executing at a time

*Nodejs is a asynchronous

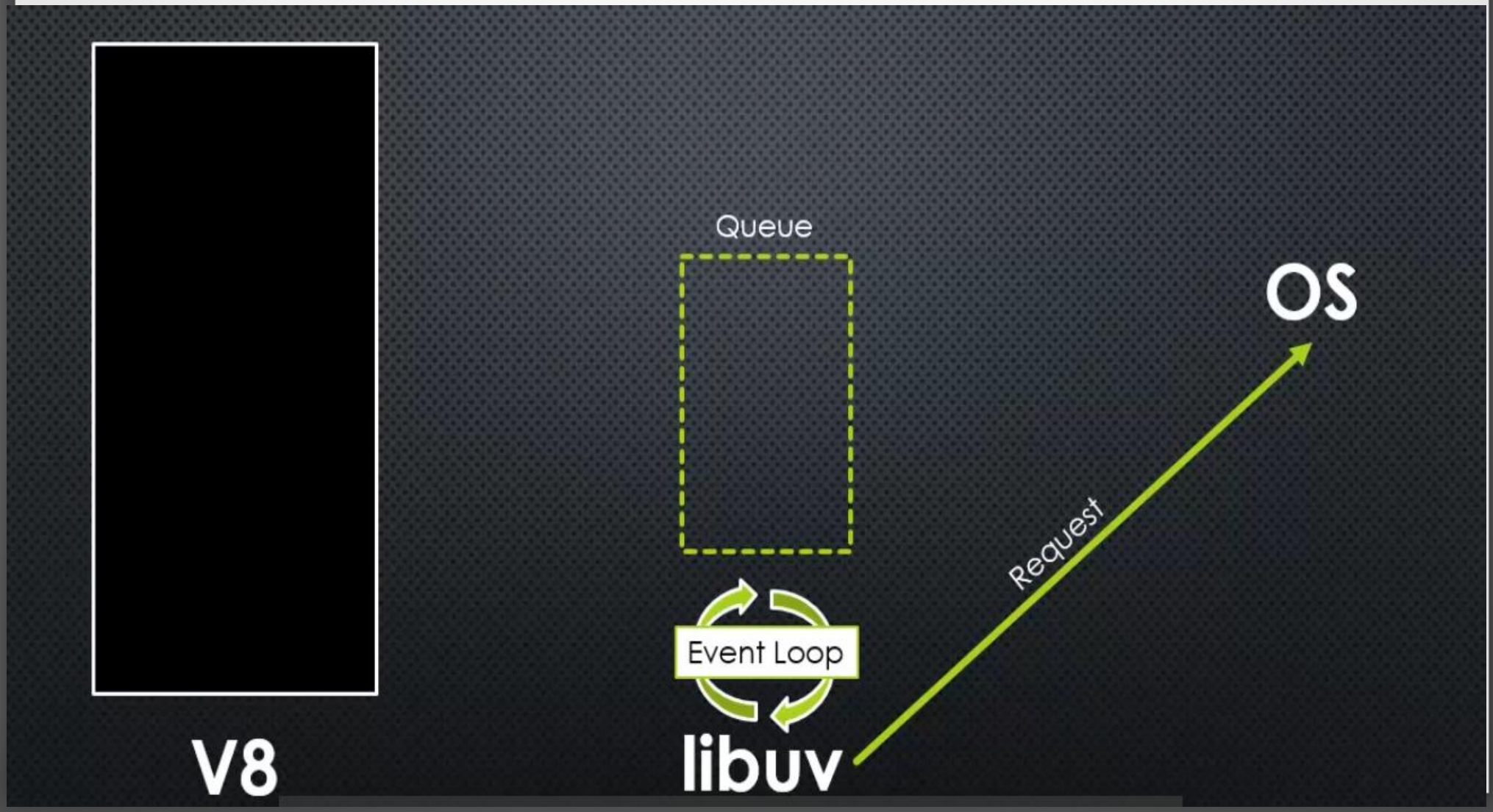
Asynchronous code & Non-Blocking code ITI

Call backs:

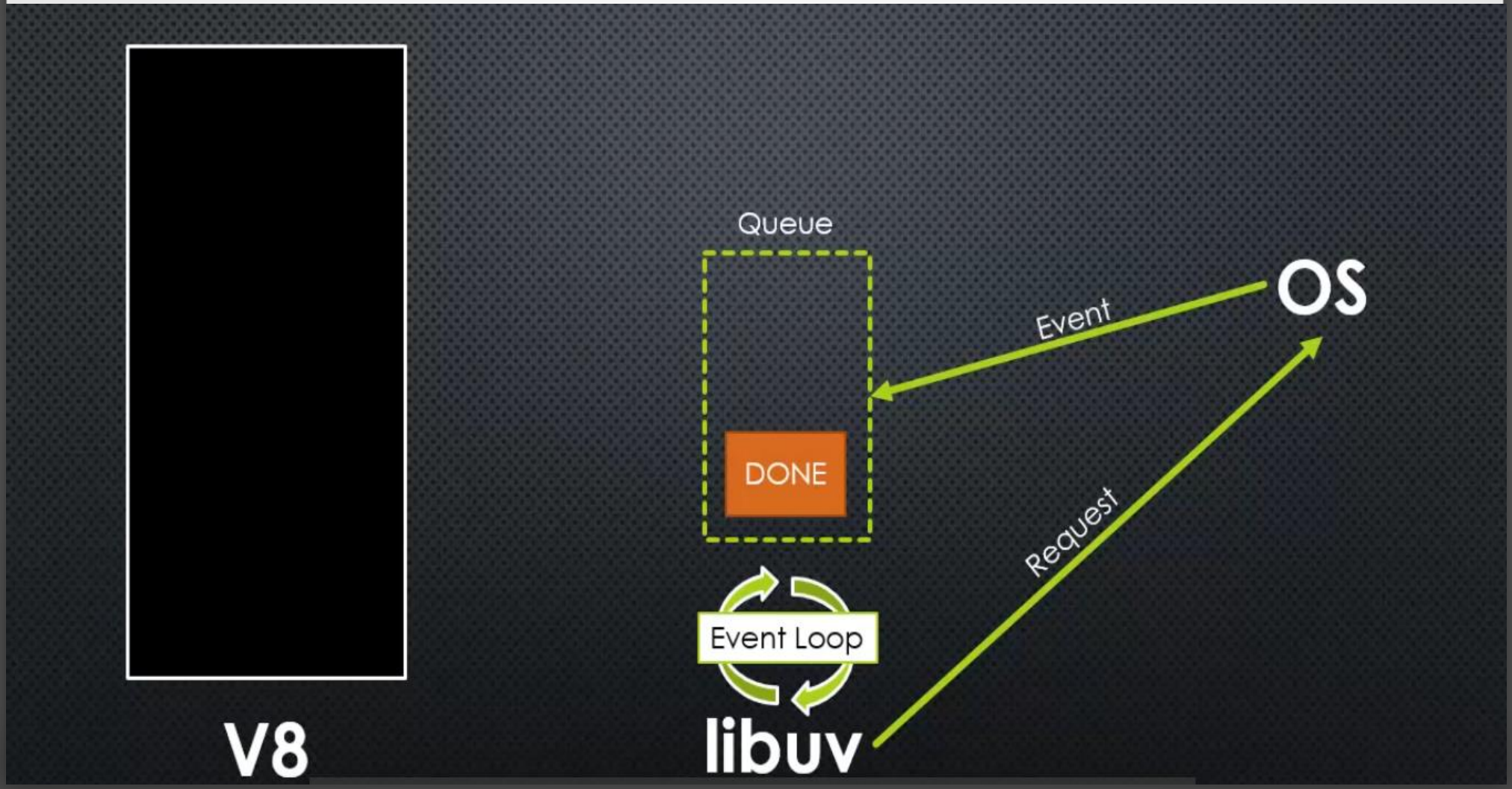
- A function passed to some other function which we assume
- Will be invoked at some point

*The function call backs invoking the function give it when it is done doing its work

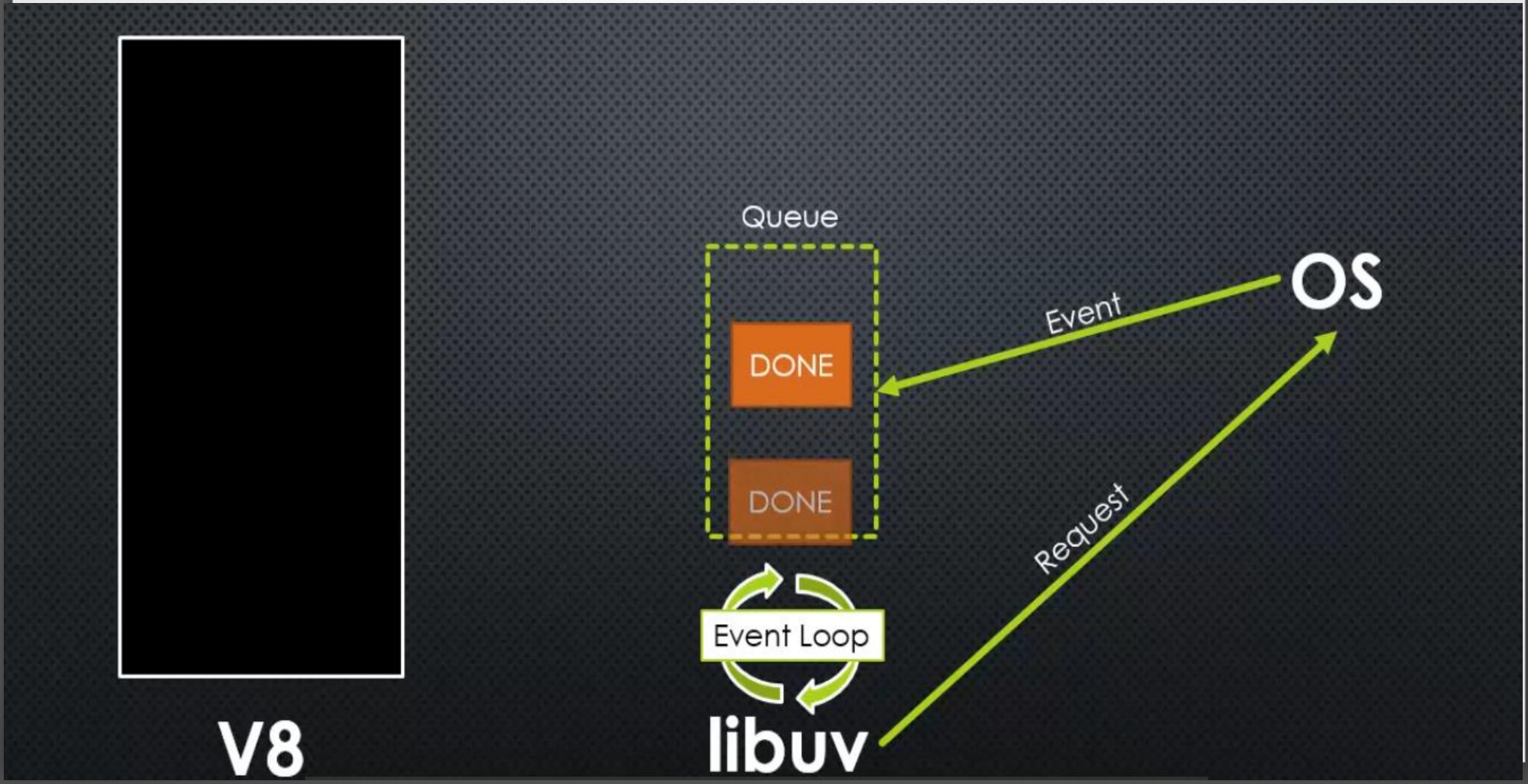
Asynchronous code & Non-Blocking code ITI



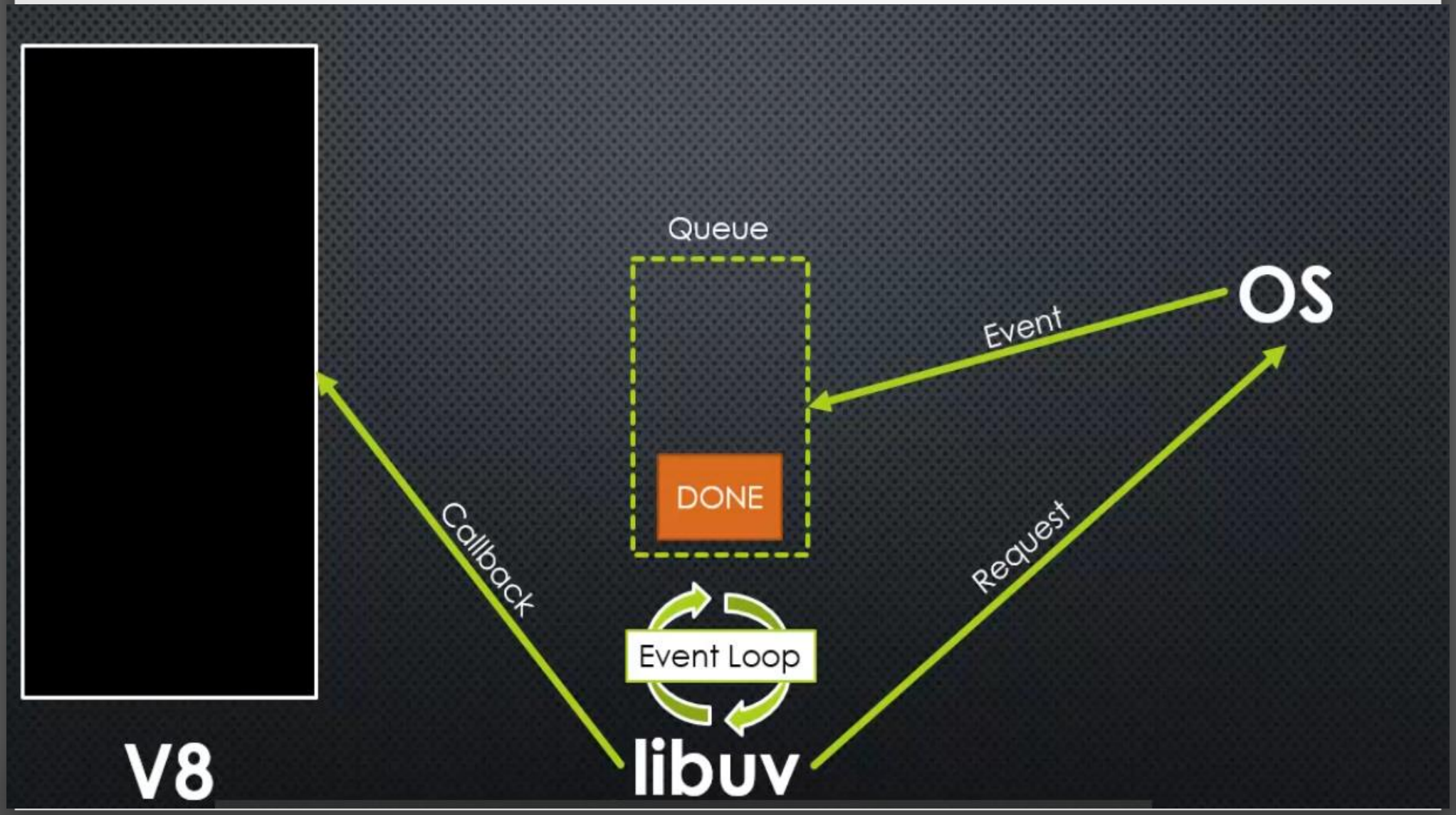
Asynchronous code & Non-Blocking code ITI



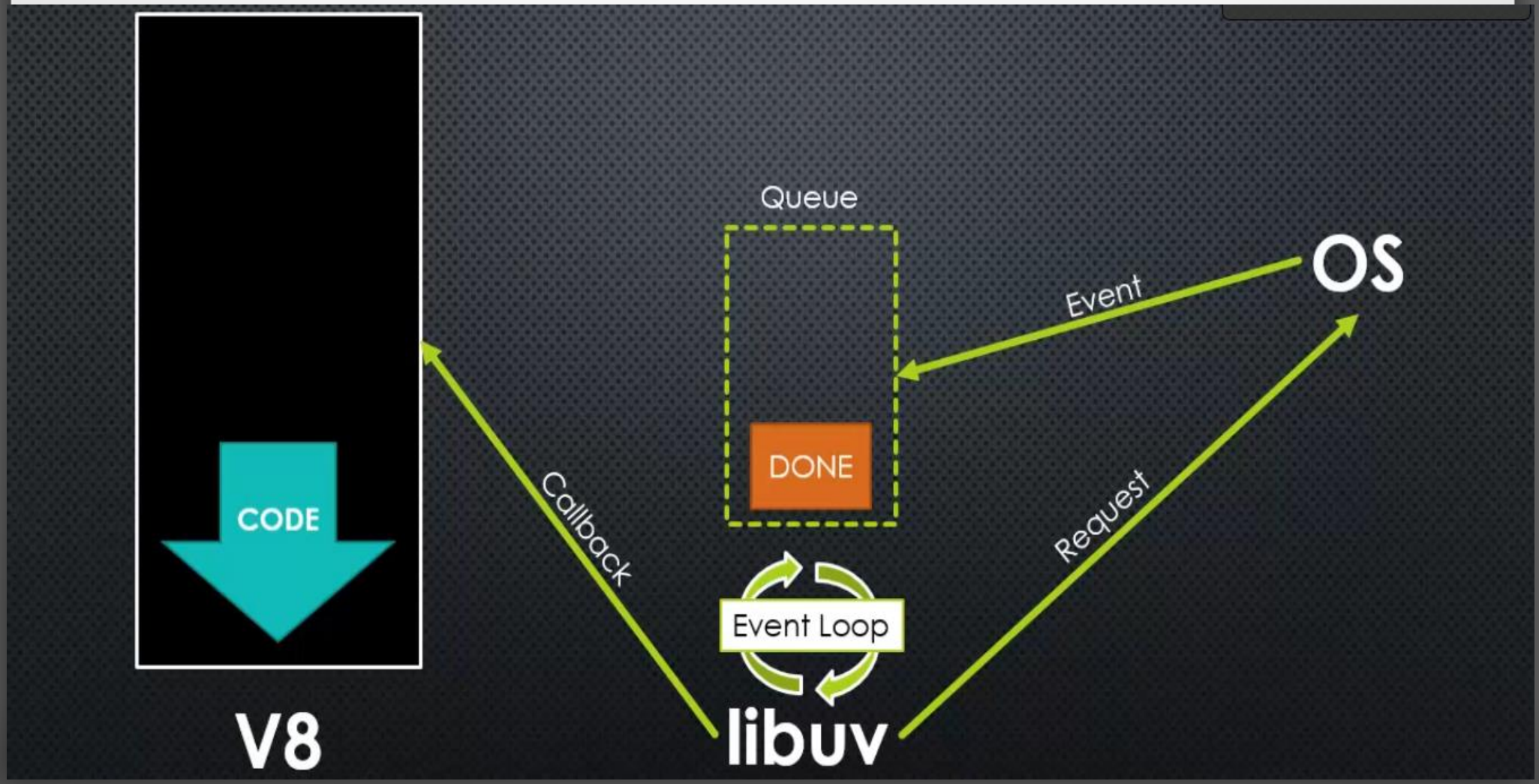
Asynchronous code & Non-Blocking code ITI



Asynchronous code & Non-Blocking code ITI



Asynchronous code & Non-Blocking code ITI



Asynchronous code & Non-Blocking code ITI

Non-Blocking:

- Doing other things without stopping your programing from running.
- Servers do nothing but I/O.
  -Scripts waiting on I/O requests degrades performance.
- To avoid blocking, Node makes use of the event driven nature of JS by attaching callbacks to I/O requests.
- Scripts waiting on I/O waste no space because they get popped off the stack when their non-I/O related code finishes executing.

*this made possible by Nodejs doing things asynchronous.
* Blocking does the opposite.

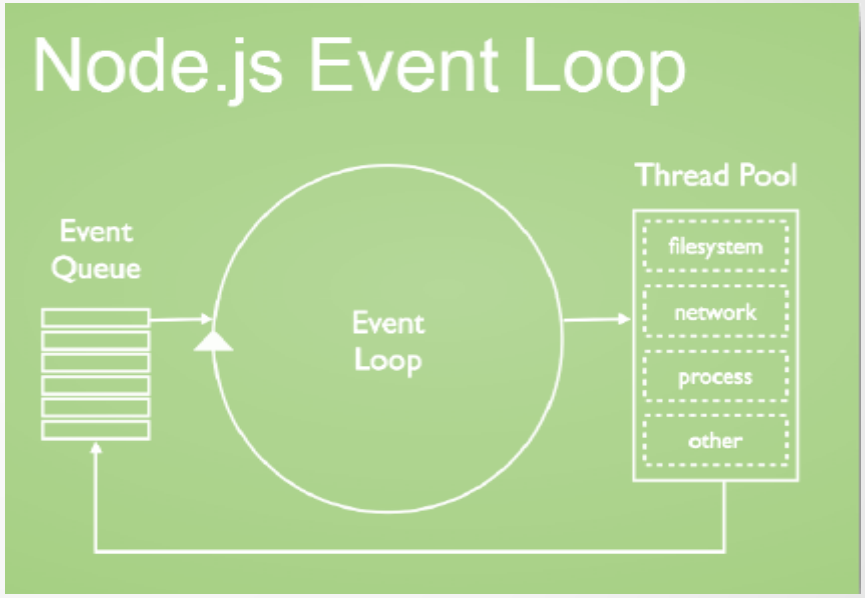
Asynchronous code & Non-Blocking code ITI

Event Loop:

- Event-loops are the core of event-driven programming, almost all the UI programs use event-loops to track the user event,

e.g. Clicks, Ajax Requests etc.

- Instead of threads Node.js uses an event loop with a stack (EventQueue).



Stream, Buffers & Read files ITI

Buffer:

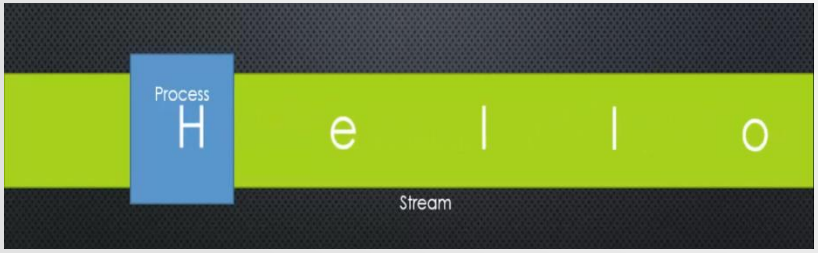
- **A temporary holding spot for data being moved from one place to another .**

***Intentionally limited in size coming and throw stream.**

Stream:

- **A sequence of data made available over time.**

***pieces of data that eventually combine into a whole.**



Stream, Buffers & Read files ITI

Read files:

Blocking code:

```
var con Read file from Filesystem, set equal to "contents"
console Print contents
console Do something else
```

Non-Blocking code:

```
fs.readFile('/etc/hosts', function(err, contents) {
  console.log(contents);
});
console.log('Doing something else');
```

Stream, Buffers & Read files ITI

Read files:

Blocking code:

Read file from File system, set equal to “contents”
-Print contents
-Do something else

Non-Blocking code:

Read file from Filesystem
whenever you’re complete, print the contents
This is a callback) (Do Something else

CALLBACK ALTERNATE SYNTAX

```
fs.readFile('/etc/hosts', function(err, contents) {
  console.log(contents);
});
```

Same as

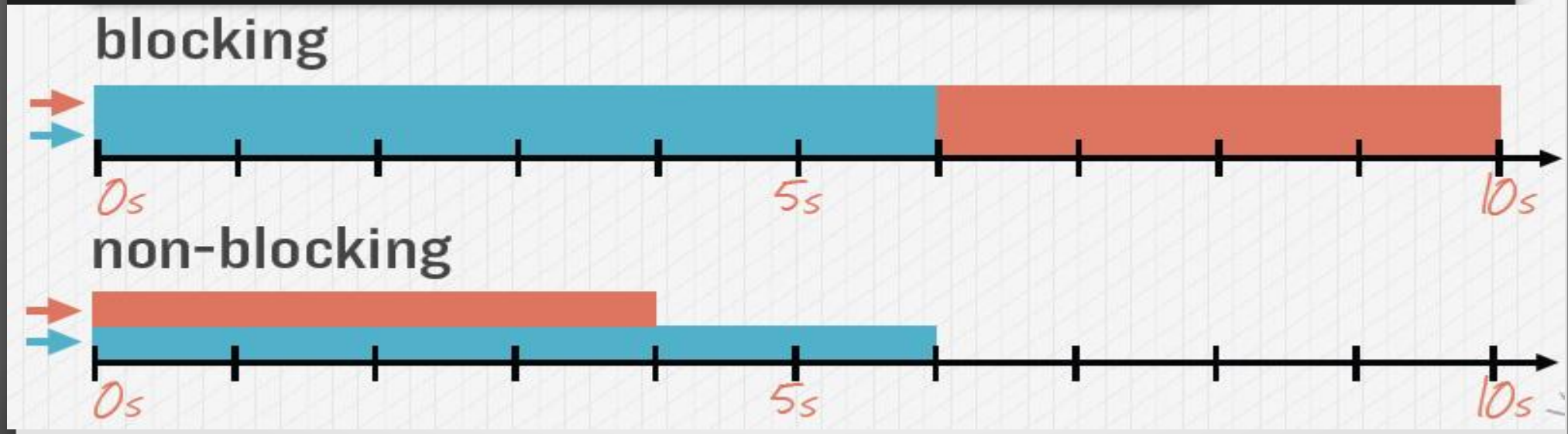
```
var callback = function(err, contents) {
  console.log(contents);
}

fs.readFile('/etc/hosts', callback);
```

Stream, Buffers & Read files ITI

BLOCKING VS NON-BLOCKING

```
var callback = function(err, contents) {
  console.log(contents);
}
fs.readFile('/etc/hosts', callback);
fs.readFile('/etc/inetcfg', callback);
```



Stream, Buffers & Read files ITI

```
var fs = require('fs')
var readeable = fs.createReadStream(__dirname + '/greet.text', {encoding: 'utf8',
highWaterMark: 16*1024})

var wrtieable = fs.createWriteStream(__dirname + '/copytext.text')
readeable.on('data', function(chunk){
    console.log(chunk)
    wrtieable.write(chunk)
})
```

Pipe:

Connecting two stream by writing to one stream what is being read from another.

***In node you pipe from a readable stream to writable stream**

```
var fs = require('fs')
var readeable = fs.createReadStream(__dirname + '/greet.text')

var wrtieable = fs.createWriteStream(__dirname + '/copytext.text')
readeable.pipe(wrtieable)
```

Create server

ITI

TCP/IP:

PROTOCOl:

- **A set of Rules Two Sides Agree On To Use When Communicating.**

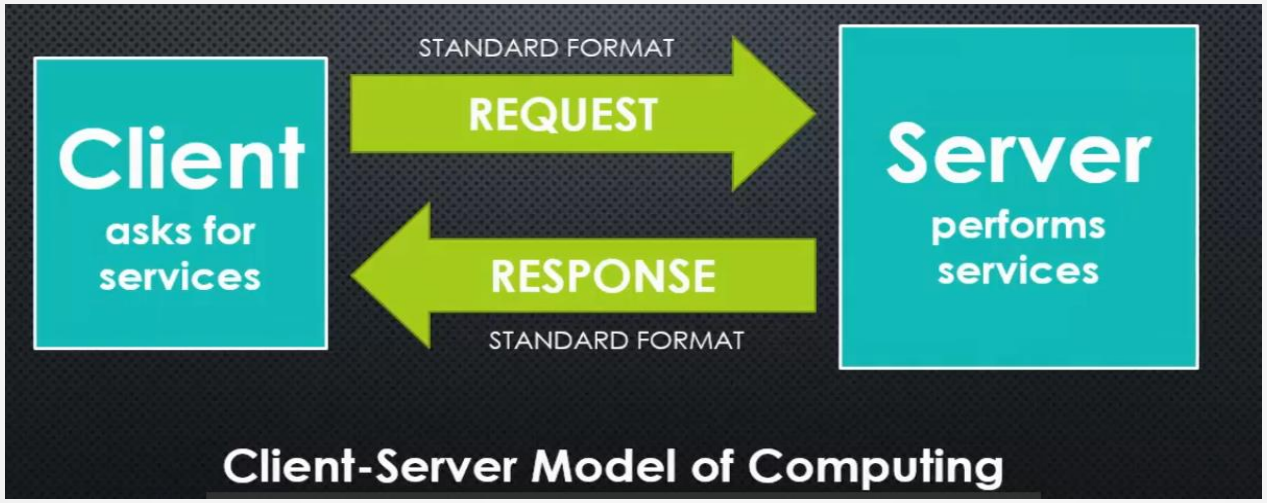
***Both the client server are programmed to understand and use that particular set of rules. It's similar to two people from different countries agreeing on a language to speak in.**

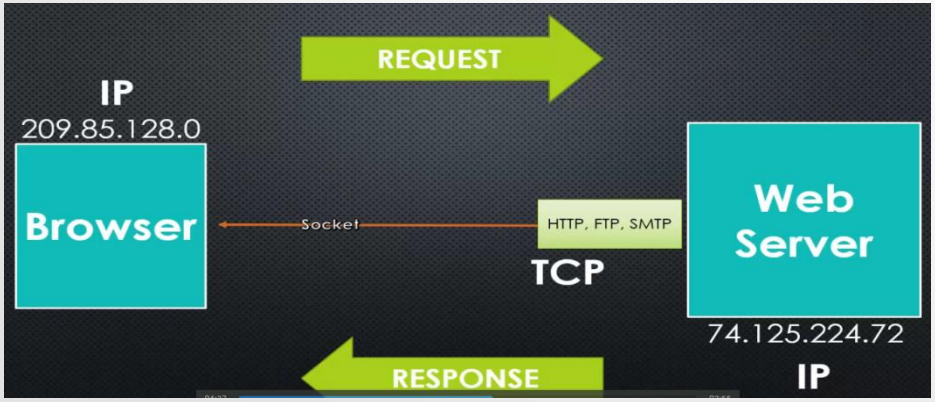
Create server

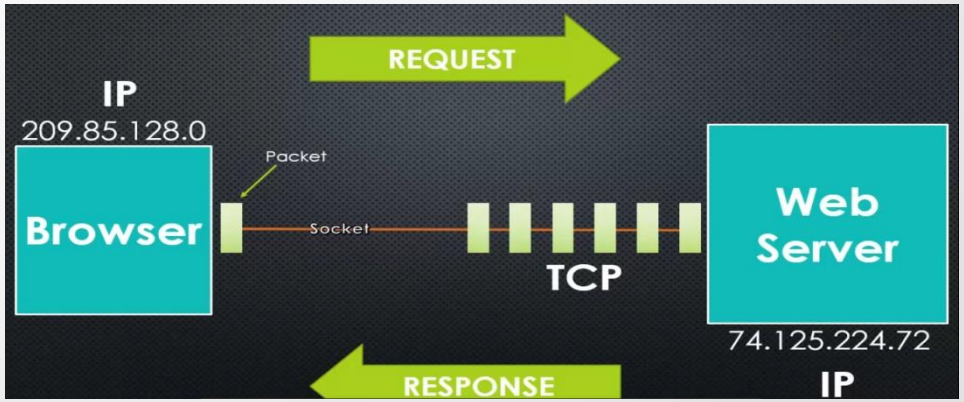
ITI

TCP/IP:

PROTOCOl:







Create server

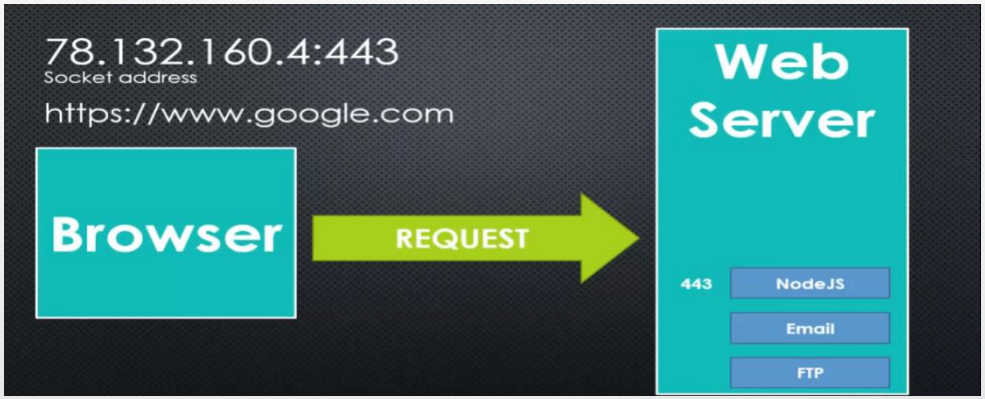
ITI

TCP/IP:

PROT:

- **Once a computer receives a packet' how it knows what program to send it to.**

*** When a program is setup on the operating system to receive packets from a particular port' it is said that the program is "listening" to the port.**



Create server

ITI

TCP/IP:

HTTP:

- **A set of rules (and a format) for Data being transferred on the web.**

***Stands for "hyper texttransfer protocol". It's a fromat (of vairous) definning transferred via TCP/IP**

```
CONNECT www.google.com:443 HTTP/1.1
Host: www.google.com
Connection: keep-alive
```

Create server

ITI

TCP/IP:

HTTP Respond:

```
HTTP/1.1 200 OK
Content-Length: 44
Content-Type: text/html

<html><head>...</head></html>
```

Create server

ITI

TCP/IP:

MIME Type:

- **A standard for specifying the type of data being sent.**

***Stands for “Multipurpose internet mail extension”.**
Examples: application/json, text/html, image/jpeg

Create server

ITI

Let’s Create a Server

Create server

ITI

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

http.createServer(function(request,response){

    console.log("request recieved");
    response.writeHead(200);//status code in header
    response.write("welcom to nodeJS world!!");//response body

    //to close the connection
    response.end();// so client knows it has recieved all data

});

//http.listen(3000,"127.0.0.1");
http.listen(3000);

// to ensure that server is running
console.log("listening on port 3000...");
```

Create server

ITI

HTTP ECHO SERVER

```
http.createServer(function(request, response){ ... });
```

But what is really going on here?

http://nodejs.org/api/

Create server

ITI

BREAKING IT DOWN

```
http.createServer(function(request, response){ ... });
```

http.createServer([requestListener])

Returns a new web server object.

The `requestListener` is a function which is automatically added to the `'request'` event.

Class: http.Server

This is an `EventEmitter` with the following events:

Event: 'request'

`function (request, response) { }`

Emitted each time there is a request.

Create server

ITI

ALTERNATE SYNTAX

```
http.createServer(function(request, response){ ... });
```

Same as

```
var server = http.createServer();
server.on('request', function(request, response){ ... });
```

This is how we add event listeners

Event: 'close'

`function () { }`

Emitted when the server closes.

```
server.on('close', function(){ ... });
```

Create server

ITI

API:

- A set of tools for Building a software application

*Stands for “application programming interface”. On the web the tools are usually made available via a ste of URLs which accepted and send only data via HTTP and TCP/IP.

Create server

ITI

ENDPOINT:

- One URL IN a Web API

*Sometimes that endpoint (URL) does multiple thing by making choices based on the HTTP request headers.

Create server

ITI

SERIALIZE:

- **Translating an object into a format that can be stored or transferred.**

***JSON, CSV, XML, and others are popular. “Deserialize” is the opposite (converting the format back into an object).**

Create server

ITI

ROUting:

- **Mapping HTTP Request to content.**

***whether actal files that server, or not .**

NPM

ITI

- NPM comes bundled with Node.js installation.
- NPM stands for Node Package Manager
- It is a package manager for Node.js
- It’s a “CLI” command line interface
- Libraries in Node.js are called packages.
- It allow us to install packages/modules, and publish our custom one and share them with other developers

www.npmjs.org

NPM COMMANDS

ITI

- Npm install pkg_nm - install a package
- Npm install -g pkg_nm - install a package globally
- Npm uninstall pkg_nm
- Npm update pkg_nm - update a package
- Npm list – show all packages installed in this application
- Npm -g list – show all packages installed globally on your PC

PACKAGE.JSON

ITI

- It’s a file that contains txt info about project’s loaded modules
- It can build or rebuild node-module folder
- Most important fields are name & version.

package.json

```
{
  "name": "myapp",
  "version": "1.0.0",
  "description": "this is a testing app",
  "main": "10_CustomModule2.js",
  "scripts": {
    "test": "echo \"Error: no test specified\" && exit 1"
  },
  "author": "NiveeNasr",
  "license": "ISC"
}
```

Administrator: Nodejs command prompt

```
E:\intake36\MyCourses_36\NodeJS\Demos>npm init
This utility will walk you through creating a package.json file.
It only covers the most common items, and tries to guess sensible defaults.

See `npm help json` for definitive documentation on these fields
and exactly what they do.

Use `npm install <pkg> --save` afterwards to install a package and
save it as a dependency in the package.json file.

Press ^C at any time to quit.
name: (Demos) myApp
Sorry, name can no longer contain capital letters.
name: (Demos) myapp
version: (1.0.0)
description: this is a testing app
entry point: (10_Scope.js) 10_CustomModule2.js
test command:
git repository:
keywords:
author: NiveeNasr
license: (ISC)
About to write to E:\intake36\MyCourses_36\NodeJS\Demos\package.json:

{
  "name": "myapp",
  "version": "1.0.0",
  "description": "this is a testing app",
  "main": "10_CustomModule2.js",
  "scripts": {
    "test": "echo \"Error: no test specified\" && exit 1"
  },
  "author": "NiveeNasr",
  "license": "ISC"
}


Is this ok? (yes) y

E:\intake36\MyCourses_36\NodeJS\Demos>
```

SEMANTIC VERSION ING AND LICENSE ITI

- License –how much you have control over the code & how others can use your code within their applications

- Version –
  version [Major].[Minor].[Patch]

SEMANTIC VERSION ING AND LICENSE ITI

- Npm init - initialize a package.jsonfile
- Npm install - install packages listed in package.json
- Npm install pkg_nm -- save - install packages and add it to dependencies in package.json file
- Npm install pkg_nm--save -dev - install packages and add it to devdependencies in package.json file
- Npm uninstall pkg_nm -- save –dev

Heavily used module

ITI

- Other frameworks that will make it easier using node

  -Express –to make things simpler e.g. syntax, DB connections .

  -Ejs–HTML template system (view engines)

  -Socket.IO –to create real-time apps

  -Nodemon –to monitor Node.js and push change automatically

  -Redis–in memory DB

  -Mongo DB

Express

ITI

- Express is a web application framework for building web apps with node as our backend system.
- Express has function names after http verbs
- We can create routes for http request methods
- Middleware is the essential building blocks of express

Create Server By Express

ITI

```
var express = require('express');
var app = express();



app.get('/', function(req, res){
    res.send('<html><head></head><body><h1>Hello</h1></body></html>')
})
app.listen(3000)
```

MiddleWare Express

ITI

```
app.use('/', function(req, res, next){
    console.log('middleware route')
    next();
})
```

```
app.use('/assets', express.static(__dirname + './public'));

app.get('/', function(req, res){
    res.send(
        "<html><head><link type=text/css href=assets/style.css rel = stylesheet />\
        </head><body><h1>Hello</h1></body></html>")
})
```

BodyParser Express

ITI

```
//body parser for post request
var bodyParser = require('body-parser')
var urlcodeParser = bodyParser.urlencoded({extended: false})
var jsonParser = bodyParser.json()
```

```
//test post request

app.post('/person',urlcodeParser ,function(req, res){
    res.send('done')
    console.log(req.body.firstname)
    console.log(req.body.lastname)
})

app.post('/personjson',jsonParser ,function(req, res){
    res.send('tahnk')
    console.log(req.body.firstname)
    console.log(req.body.lastname)
})
```

QueryString

ITI

```
<body>
    <h1>Person <%= ID %></h1>
    <h2>QSTR: <%= QSTR %></h2>
</body>
```

Ejs

ITI

- Often used view/templating engine with Express
- Express expect to have all its templates in view directory
- Ejsstands for Embedded javaScript
- ".ejs" should pass to render() with requiredparameters as an object

```
<!DOCTYPE html>
<html>
<head>
    <title>Untitled Document</title>
</head>
<body>
    <%-include("head.ejs")%>
    <form method="post">
        <input name="name" type="text" />
        <input type="submit"> </form>
    <hr/>
    <ul>
       <%for (var i=0; i<names.length; i++){%>
           <li>
               <%= names[i] %>
           </li>
       <%}%>
    </ul>
</body>
</html>
```

```
app.set("view engine", "ejs");
app.get("/", function (req, res) {
    res.render("one.ejs", {
        name: "ali", names: names
    })
});
```

Ejs

ITI

```
// engine
app.set('view engine', 'ejs')


app.get('/person/:id', function(req, res){
    res.render('person', {'ID': req.params.id})
})

app.get('/', function(req, res){
    res.render('index')
})
```

Application Settings

ITI

- set()
- get()
- enable()
- disable()

```
//application Settings
app.set("view engine","jade");
app.set("views","templates");

app.enable("view cache");

app.enable("case sensitive routing");//disabled by default (/hello !=/Hello)
app.enable("strict routing");//enabled by default (/hello =/hello/)

app.disable("x-powered-by");//enabled by default
```

MIDDLEWARE ITI

- Functions show user something goes between the browsers and what we do with data from browsers

- Middleware category
  -Third party
  -Custom middleware using app.use() register a piece of middleware
  -Built-in
  -Routing function
  -Parameter based

- Requests flow move from up to down and move through middleware as they go down

MIDDLEWARE

ITI

```
//3rd party Middleware
app.use(bodyParser.urlencoded({
  extended: true
}));

//custom Middleware
app.use(function(res,req,next){

    next();
});

//routeFunction Middleware
app.get()

//built-in Middleware
app.use(express.static("./public"));
```

Socket.IO

ITI

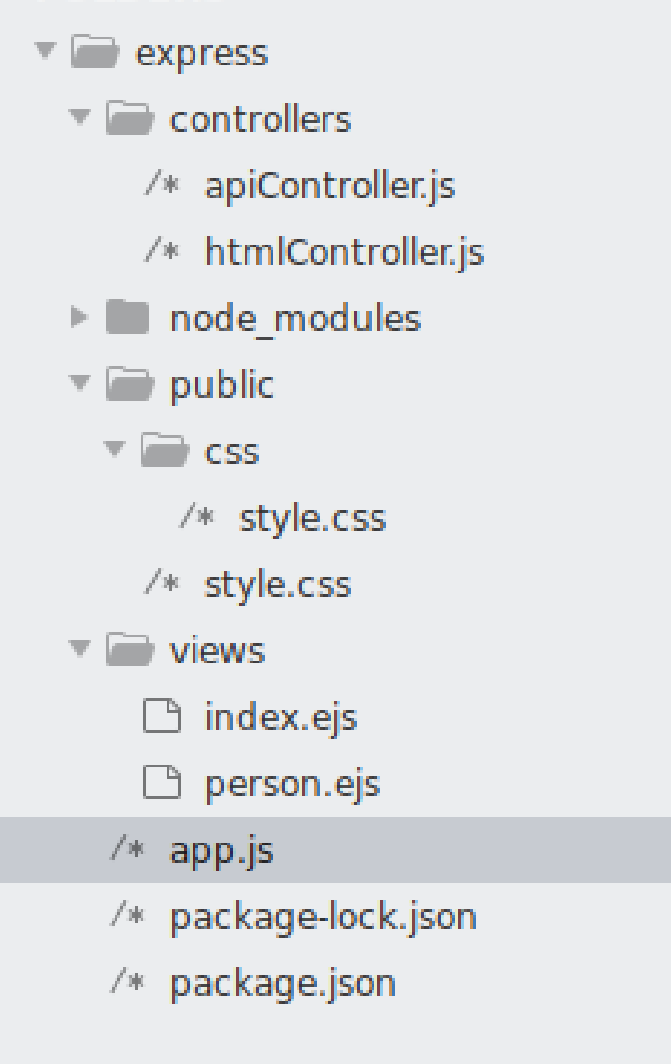
- Websocketis bidirectional communication between browser and server
- Socket-iois front-end and back-end solution designed for nodejs

Structer App

ITI

```
.
├── app.js
├── bin
│   └── www
├── package.json
├── public
│   ├── images
│   ├── javascripts
│   └── stylesheets
│       └── style.css
├── routes
│   ├── index.js
│   └── users.js
└── views
    ├── error.jade
    ├── index.jade
    └── layout.jade

7 directories, 9 files
```



https://github.com/elmahdy-intake37/node-projects?fbclid=IwAR0UwzhbwX6__FyrUbYmmJl-quTs_xgzHHcugsvhogbZwzHomx0TiaG6eKga