Web Programming

Node.js

200709006 - Affan Selim KAYA
200709011 - Onur Alp AKIN
190709009 - Emre ERZURUM

Overview

- In this presentation we will look on Node.js on 3 different parts which are:
- Part 1: Introduction to Node.js, V8 JS engine and Node.js benefits
- Part 2: Security on Node.js, Async I/O and Node.js examples
- Part 3: Applications and firmwares using Node.js and Node.js vs other backends

Part 1: Introduction, Installation, Demo

Introduction to Node.js (1/2)

- Node.js is an open-source and cross-platform JavaScript runtime environment.
- It is free to use for everyone.
- It is quite popular among devolopers and web programmers.
- Node.js has many pros but has cons too as every environment.
- Node.js is licensed under MIT.
- Node.js runs on Windows, Linux, Unix, Mac OS X and other platforms.

Introduction to Node.js (2/2)

- Node.js uses V8 JS engine.
- Node.js uses JavaScript on the servers.
- It written in C, C++ and JavaScript.
- Node.js is primarily used to build network programs such as Web servers.
- Node.js uses ‘.js’ extension on its files.

What is V8 JS Engine?

- V8 engine firstly released at 2 September 2008.
- Firstly developed by Chromium (Open-source Google Chrome) project developers (By leadership on Lars Bak)
- It compiles JS to native machine code (IA-32,x86-64,ARM)
- V8 is Google Chrome’s open source high-performance and fast JavaScript and WebAssembly engine
- It is written in C++
- V8 runs on Windows7 or later versions on Microsoft, macOs 10.12 ,and Linux systems that uses x64 base

Why V8 Engine Is Preferred?

- V8 translates js code to machine code and doesn’t use interpreter thus it’s fast
- V8 engine has a very big community
- V8 engine is the backbone of Google Chrome and other Chromium-based web browsers therefore it’s reliable
- V8 enables any C++ application to expose its own objects and functions to JavaScript code.

How V8 Engine Works?

- **When a developer runs a JS script on V8, the following steps are taken by the engine**
  1. The engine compiles and executes the JS code
  2. The engine handles the call stack
  3. The engine manages the memory heap
  4. The engine handles the garbage collection
  5. The engine provides all the data types, objects and functions
  6. The engine also provides the event loop (sometimes implemented by the browser as well)

Foundation of Node.js

- Originally, Node.js was founded by Ryan Dahl.
- It’s initial release is at 27 May 2009
- Node.js’s foundation’s goal is to provide an easy way to build scalable network programs
- In January 2010, a package manager was introduced for the Node.js environment called npm (will be explained later)
- In January 2012, Dahl stepped aside, promoted coworker and npm creator Isaac Schlueter to manage the project.

What is NPM?

- Npm is package manager of node.js
- The package manager does the heavylifting for programmers
- Makes it easier for programmers to publish and share Node.js packages
- Purpose of it is to simplify installation, updating, and uninstallation of packages

Reasons to Use Node.js (1/3)

- **Node.js can :**
  - create dynamic pages
  - easily handle file operations on server
  - collect form data
  - handle database operations very well
- **It is cross-platform**
  - Android, iOS, Windows, Linux etc.
- **Relatively easy to learn (it’s just js)**

Reasons to Use Node.js (2/3)

- **Node.js is :**
  - easy to tinker with
  - effortless to configure
- **Once written runs everywhere**
  - kinda like java
- **Node.js is utilized by more than 20 million sites on the internet.**

Reasons to Use Node.js (3/3)

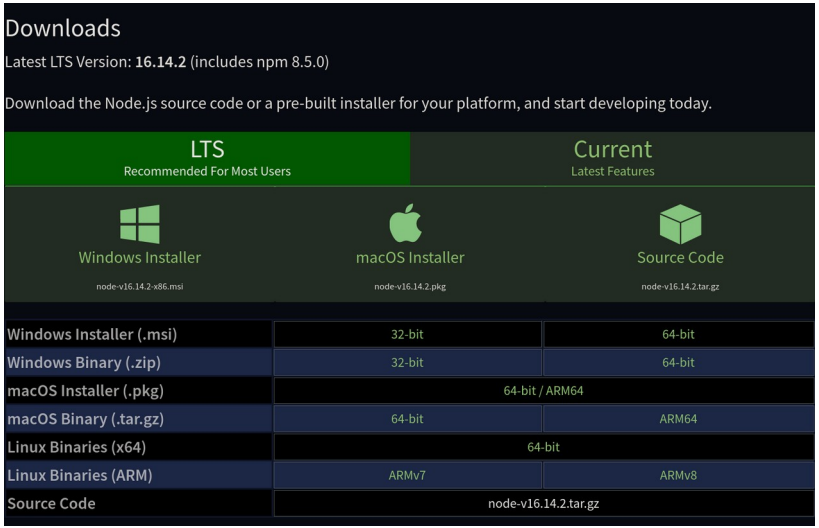
- **Node.js has:**
  - large community generating tens of thousands packages
  - lots of available free content on internet.
  - consistency with most databases
    - MySQL, sqlite3, MongoDB, NoSQL
- **Node.js can make front-end developers full-stack developers**

Reasons Not to Use Node.js

- **Node.js doesn't provide scalability**
- **If you’re doing CPU intensive tasks single thread won’t be enough**
  - High speed i/o operations where node.js shines the most.
- **Big node.js packages can have insufferable amount of dependencies**
- **Dealing with relational databases is relatively hard for node.js**
- **If you’re not careful you can have multiple callback functions for single callback function**

Installation of Node.js (1/5)

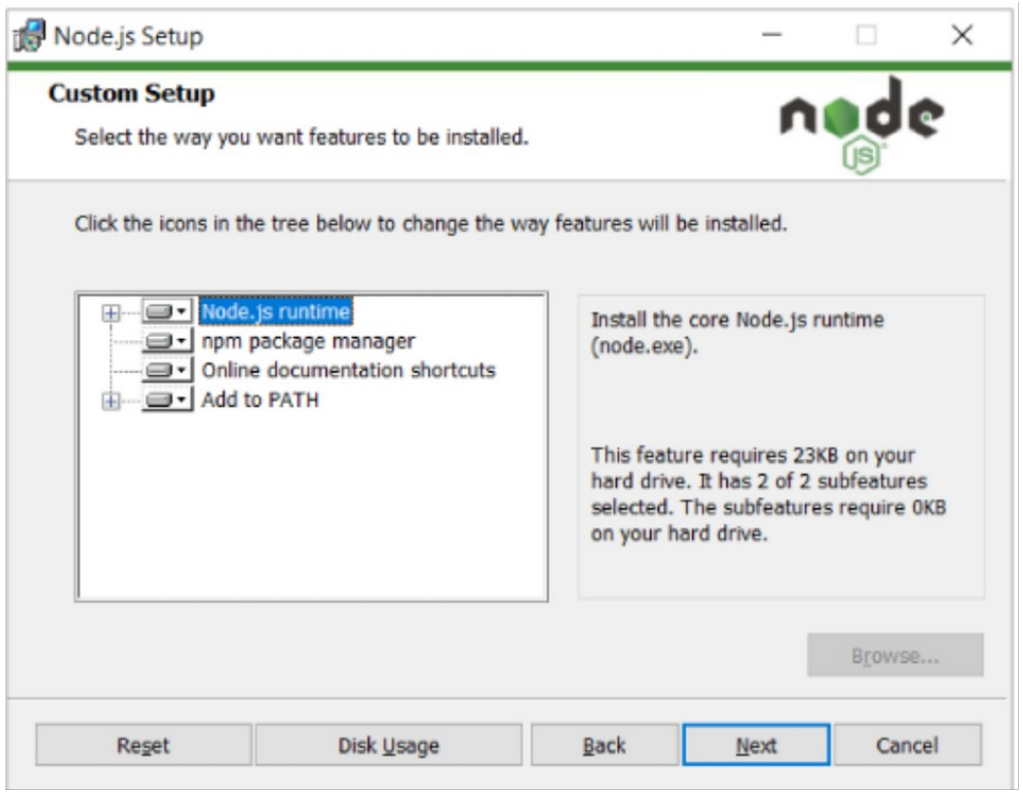
- Go to Node.js official website
  - https://nodejs.org
- Go to the download section
  - https://nodejs.org/en/download/
- You'll see LTS and Current versions
- Select which operating system you are using
- Download of desired verison.



Installation of Node.js (2/5)

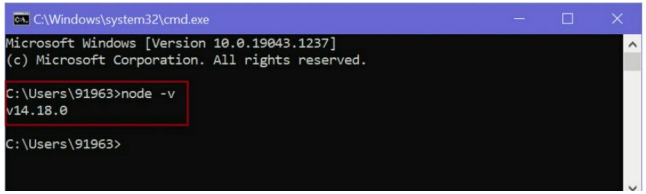
- **These instructions will be for Windows**
- **Open the downloaded installer**
- **On the opened screen, click next if you agree to the terms.**
- **Now select the installion folder.**
  - Default is just fine
- **The wizard will let you select components to install.**
- **Again, unless you have a specific need leave it default**
- **After that, software will setup everything automatically**

Installation of Node.js (3/5)



Installation of Node.js (4/5)

- **To verify installation:**
  - Open a command window or PowerShell and write
    - node -v
  - Hit enter
  - This should display the installed version of node.js
  - You can do the same thing for npm
    - npm –v
  - Now you are good to go



Installation of Node.js (5/5)

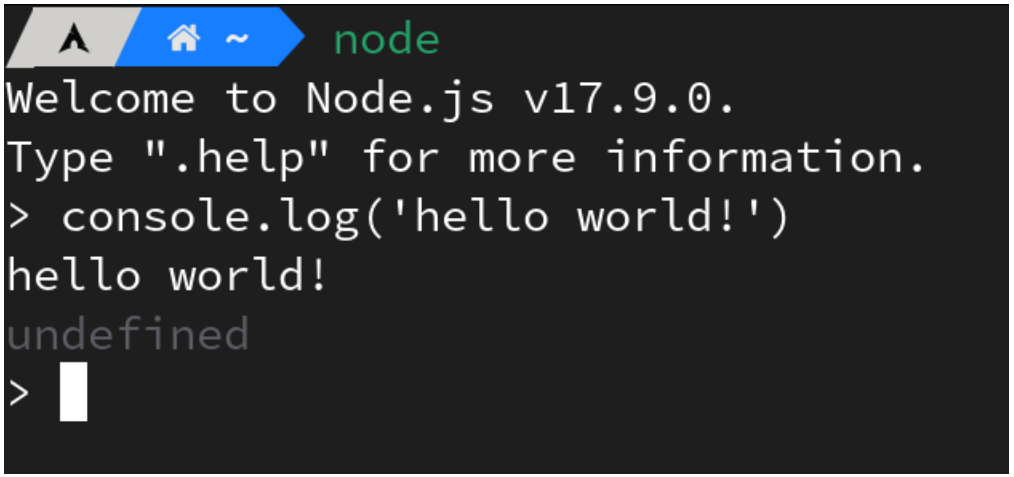
- **To upgrade node.js:**
  - The easiest way to would be downloading the latest version again
  - Download the installer and run it
  - The setup wizard will overwrite the old version, and replace it with new one.

Node.js Command Prompt (1/2)

- **The screen you see after you write `node` is called command prompt**
- **This is how we use node.js**
- **For example if you wanted to run `Hello World`**
- **You will need to either:**
  - Open interactive prompt and then write the command
  - Write it in a script file then run it

Node.js Command Prompt (2/2)

- **Below image is the former one (interactive)**



Creating a Node.js Package (1/2)

- **To install or manage dependencies you need to create a node package**
- **Even though you can create packages manually it’s not recommended**
- **Instead you should let npm handle it for you**
- **To create package you need to type**
    - npm init

Creating a Node.js Package (2/2)

- It will ask you questions then it will create your package.

```
/tmp/myNodePackage 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 init` for definitive documentation on these fields
and exactly what they do.

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

Press ^C at any time to quit.
package name: (mynodepackage)
version: (1.0.0)
description:
entry point: (index.js)
test command:
git repository:
keywords:
author:
license: (ISC)
About to write to /tmp/myNodePackage/package.json:
```

```
{
  "name": "mynodepackage",
  "version": "1.0.0",
  "description": "",
  "main": "index.js",
  "scripts": {
    "test": "echo \"Error: no test specified\" && exit 1"
  },
  "author": "",
  "license": "ISC"
}


Is this OK? (yes)
npm notice
npm notice New minor version of npm available! 8.5.5 -> 8.7.0
npm notice Changelog: https://github.com/npm/cli/releases/tag/v8.7.0
npm notice Run npm install -g npm@8.7.0 to update!
npm notice

/tmp/myNodePackage
```

Installation of a Node Package

- **To install new node packages you need to type**
  - npm install
- **It will install and check for vulnerabilities**

```
~/Dev/n/NodeDemo npm install express

added 57 packages, and audited 58 packages in 3s

7 packages are looking for funding
  run `npm fund` for details

found 0 vulnerabilities
```

Part 2: Async, Event Loop, Node.js Demos, Security

Node.js and Async Operations

- **Node.js does two concepts very well**
  - Asynchronous code execution
  - Non-blocking I/O
  - What is the output?

```
"use strict";

const main = () => {
    console.log('Tyger Tyger, burning bright,');
    console.log('In the forests of the night;');
    console.log('What immortal hand or eye,');
    console.log('Could frame thy fearful symmetry?');
};

main()
~
~
```

Output

- **When node.js is not using ‘async function’s it executes codes line by line**

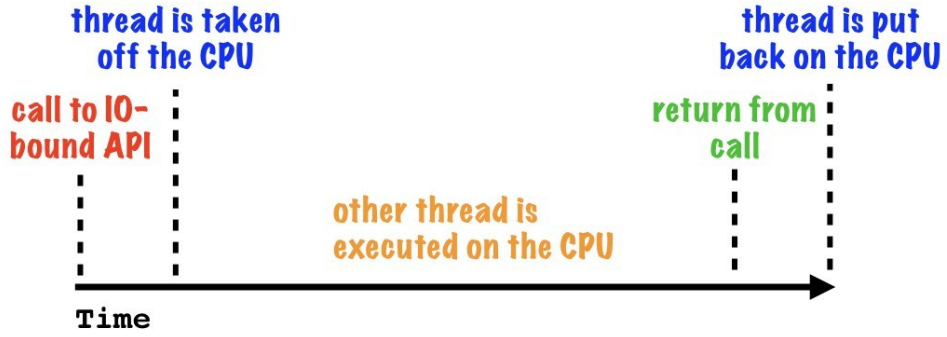
```
~ node standart.js
Tyger Tyger, burning bright,
In the forests of the night;
What immortal hand or eye,
Could frame thy fearful symmetry?
```

Asynchronous Code Execution

- **Asynchronous functions allow you to do execute many instructions in parallel**
- **It’s a trade-off between time and computation power**
- **For example**
  - If you write a server in node.js, your server can create files without making the clients wait

Non-blocking I/O

- And the concept of not waiting for output while giving input is what node.js all about
- They are not seperate concepts
- But rather async functions allow non-blocking i/o



Asynchronous Code Execution Demo

- When you use async functions you'll start to see glimpse of power of node.js

```
"use strict";

const doSomethingAsync = () => {
  return new Promise(resolve => {
    setTimeout(() => resolve('Some very heavy calculation is now finished'), 3000);
  })
}

const doSomething = async () => {
  console.log(await doSomethingAsync())
}

console.log('Before')
doSomething()
console.log('After')
```

- Now what do you think the output is?

Output

- **Pretty cool huh?**

```
~ node async.js
Before
After
Some very heavy calculation is now finished
```

Event Loop (1/2)

- But how can node do this?
- The answer is: event loop
- The event loop is one of the most important aspects to understand about Node.js
- The event loop works by reading code line by line than adding each instruction to call stack
- The event loop has a single thread

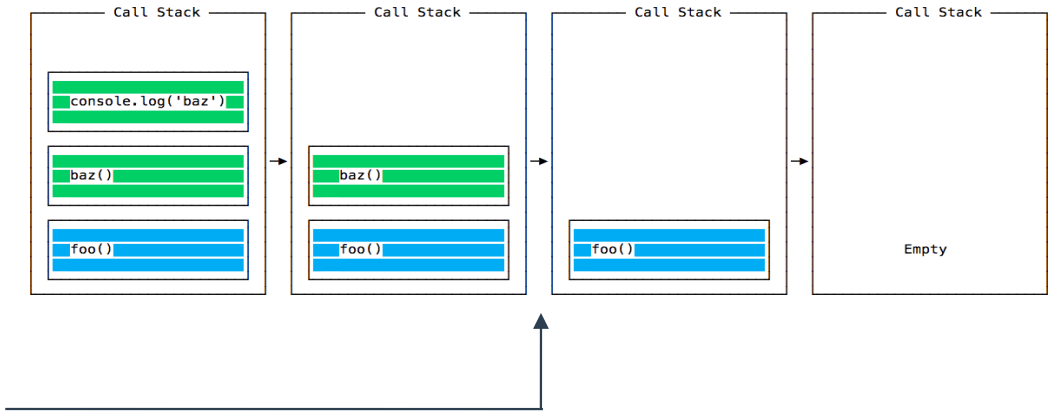
Event Loop (2/2)

- Call stack can handle ramifications quite decent
- We will visualize call stack as an abstraction of CPU cycles

```
const bar = () => console.log('bar')

const baz = () => console.log('baz')

const foo = () => {
  console.log('foo')
  bar()
  baz()
}

foo()
```

- Let’s pick an example to understand

Call Stack of the Example



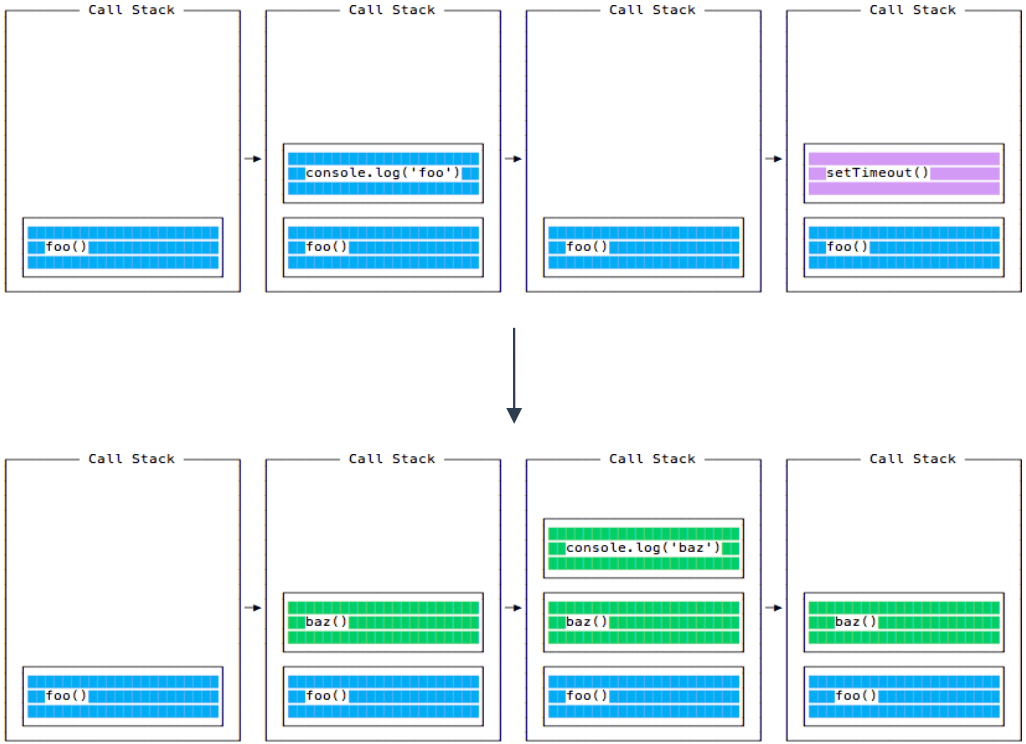
Event Loop of an Async Function

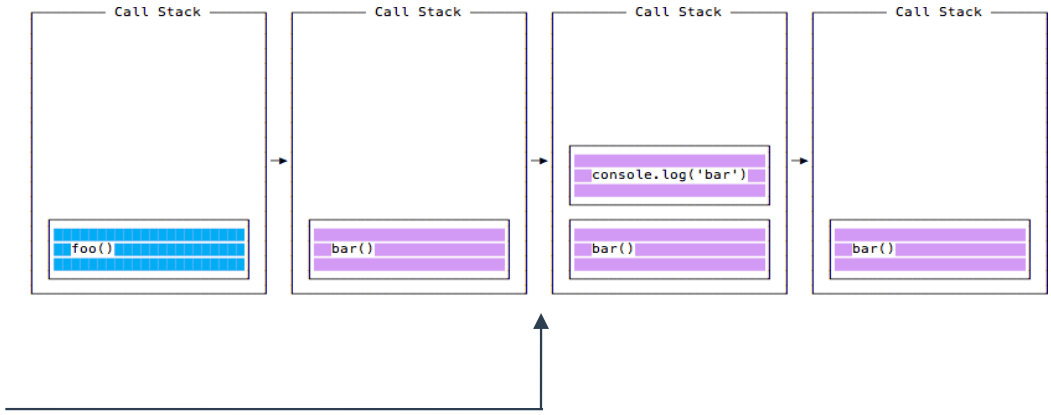
- Now we will see how async operation will look
- SetTimeout is an async function

```
const bar = () => console.log('bar')

const baz = () => console.log('baz')

const foo = () => {
  console.log('foo')
  setTimeout(bar, 0)
  baz()
}

foo()
```

- This will be our example

Call Stack of the Async Example





Node.js Filesystem API (Demo)

- Node.js can create, delete, list files
- And can do many more

```
> fs.readFile('hello.txt', 'utf-8', function (err, data) {
...   if (err) throw err;
...   console.log(data);
... });
undefined
> hello world
```

Node.js OS API (Demo)

- **Node.js can handle information or status about operating system**

```
> os.networkInterfaces()
{
  lo: [
    {
      address: '127.0.0.1',
      netmask: '255.0.0.0',
      family: 'IPv4',
      mac: '00:00:00:00:00:00',
      internal: true,
      cidr: '127.0.0.1/8'
    },
    {
      address: '::1',
      netmask: 'ffff:ffff:ffff:ffff:ffff:ffff:ffff:ffff',
      family: 'IPv6',
      mac: '00:00:00:00:00:00',
      internal: true,
      cidr: '::1/128',
      scopeid: 0
    }
  ],
```

Node.js Web-Request API (Demo)

- Node.js can make requests too. You can either use third party library or standart module which is `https`

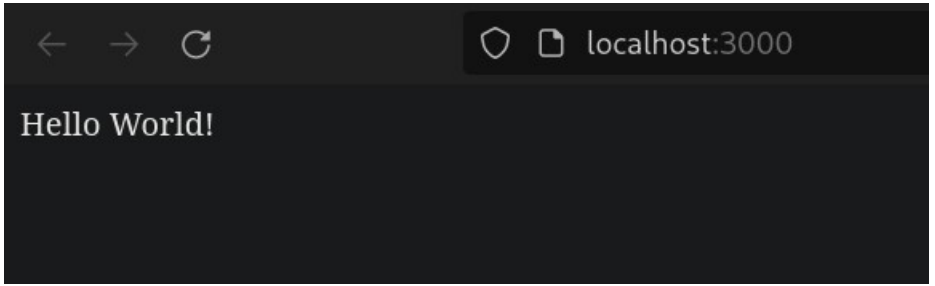
```
const https = require('https')
const options = {
  hostname: 'example.com',
  port: 443,
  path: '/todos',
  method: 'GET'
}

const req = https.request(options, res => {
  console.log(`statusCode: ${res.statusCode}`)

  res.on('data', d => {
    process.stdout.write(d)
  })
})

req.on('error', error => {
  console.error(error)
})

req.end()
```

```
<body>
<div>
    <h1>Example Domain</h1>
    <p>This domain is for use in illustrative examples in documents. You may use this
    domain in literature without prior coordination or asking for permission.</p>
    <p><a href="https://www.iana.org/domains/example">More information...</a></p>
</div>
</body>
```

Node.js Web-Server (Demo)

- You can create servers pretty fast with express

```
const express = require('express')
const app = express()
const port = 3000

app.get('/', (req, res) => res.send('Hello World!'))

app.listen(port, () => console.log(`Example app listening on http://localhost:${port}`))
```



Node.js Dependencies

- When you install npm packages your code depends on that package now
- And that installed package has dependencies too
- This ramification until root is called dependency tree
- When a dependent package has a vulnerability, now your package is also vulnerable

Insecure Code

- And when popular node packages are vulnerable, lots of apps will be exploitable
- You can imagine how it’s susceptible to havoc
- If a hospital running node.js is vulnerable then people could get hurt

Prevention

- **And to prevent that npm released it’s own security linter `audit` in 2018**
- **You can check your code for vulnerabilities by typing**
    - npm audit
- **In order to avoid insecure code you should educate yourself in secure code practices**

Npm Audit

- This is the preview of npm audit command
- It will list known vulnerabilities
- To fix issues you can type
  - npm audit fix

Dependency Chaos (1/2)

- Real world example of harm caused by dependency would be `left-pad`
- Left-pad is written by Azer Koçulu
- Left-pad is a very simple node-package with 11 lines of code
- Left-pad can be used to add characters to the beginning of a string
- The author of this package decided to delete it over a dispute with npm

Dependency Chaos (2/2)

- After the deletion programmers around the world got error message

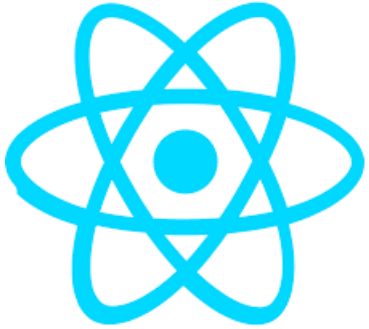
```
npm ERR! 404 'left-pad' is not in the npm registry.
```

- It meant that the code they were trying to run, required a package called left-pad, but the npm registry didn’t have it.
- Some of the largest, most widely used npm packages were suddenly broken. It even affected React.
- It’s absence was felt globally
- This programmer was able to briefly break the internet just by deleting one package.

Part 3: Node.js Apps, Top Node.js Apps, Node.js vs Other Backends

Node.js Usage Areas

- Node.js has many usage areas
- Most of them are web-apps
- Nearly all of node.js apps are related to web-apps even if they’re not directly web-apps



Real-Time Chats

- Most people use phone apps or social media apps
- Node.js can create chat rooms very easily because because of single-threaded asynchronous model of nodejs.
- It is easily scalable and is often used to build chatbots.

IoT (Internet of Things) (1/2)

- IoT applications often has multiple sensors
- They often send small chunks of data
- Massive amounts of data can be collected
- Inserting large quantities of data into a database usually create a bottleneck
- If system has bottleneck it will not function properly

IoT (Internet of Things) (2/2)

- Node.js is a good choice as it can handle these concurrent requests quickly.
- Node.js can receive data, then send it to the backend in a piecewise manner
- Piecewise operations ensure that data gets stored without breaking any system

Data Streaming

- Node.js is light and fast
- Thus it’s a great choice for applications that process many short messages
- Node.js is a perfect choice for applications require low latency
- For that reason companies like Netflix use Node.js for data streaming purposes.
- Also, Node.js provides a native streaming API.

SPAs (Complex Single-Page Applications)

- In SPAs, the entire application is loaded on a single page
- This means that there are several requests made in the background
- Things like validation code between client machine and server
- Node’s ability to process many requests at low response times makes great fit for modern web applications

REST API Based Applications

- Because of aforementioned reasons (fast, scalable and non-blocking asynchronous structure)
- Nodejs is really suitable for rest api development.
- Node.js also provides packages like Express.js that make building web applications even easier
- You can develop custom APIs in matter of hours

Top 5 Apps Using Node.js

- PayPal
- Netflix
- LinkedIn
- Uber
- eBay

Node.js in PayPal

- Paypal is a money transaction application so it need’s to be reliable
- PayPal was built on Java
- But recently it recently migrated to Node.js
- As of 2015, the service has over 184 million active users
- And it uses Node.js for front-end of the website

Node.js in Netflix

- As mentioned before Netflix uses Node.js
- The whole user interface on Netflix.com is built with Node.js
- Node.js delivers a blazing-fast, modular, and lightweight application
- With the usage of Node.js Netflix’s web app load time has been reduced by 70%
- Now proven effective, the company decided to move the entire data access layer to Node.js recently

Node.js in LinkedIn

- **Node.js powers the server-side of LinkedIn’s mobile app**
- **The new mobile app is with better performance:**
  - Consume lower memory
  - Has improved resource utilization
  - is 20 times faster than the previous ruby based version
  - cut down the number of servers needed

Node.js in Uber

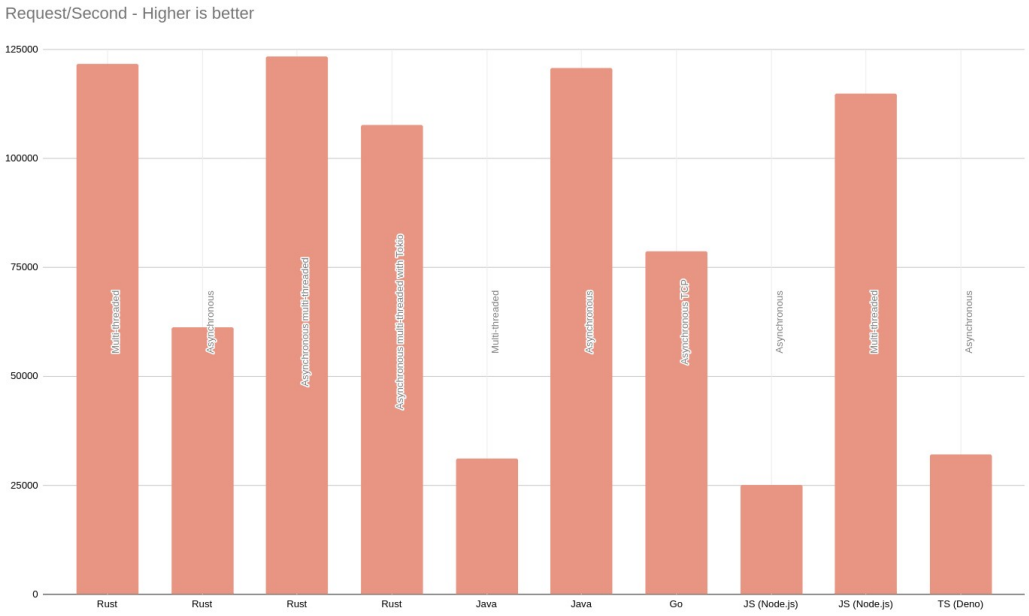
- **Uber was one of the first companies that adopted Node.js**
- **With the usage of Node.js in Uber's app:**
  - Performance was improved drastically
  - Information was processed quickly
  - Speed was boosted up
  - Any errors could be addressed right away

Node.js in eBay

- **Node.js is part of the eBay' tech stack**
- **After the switch to Node.js, eBay reported:**
  - Improved speed
  - Improved simplicity
  - Improved scalability
  - Improved transparency and control

Node.js vs Other Fierce Backends

- **Even though Node.js is popular, it’s not the only option for speed**



Node.js vs Go (1/2)

- Go is a statically-typed programming language introduced by Google
- It is open-source and has syntax similar to C
- In terms of raw speed Go is slightly better than Node.js
- Go doesn’t need an interpreter, it’s a compiled language
- So go has the same level of performance low-level languages like C++

Node.js vs Go (2/2)

- **In terms of IO operations, Go is similar to Node.js.**
- **Both has a garbage collector**
  - Helps preventing memory leaks
  - Ensure stability
- **Node.js is only little behind Go in performance.**
- **Single-threaded Node.js boosts efficiency**
- **V8 Javascript engine discards the need of an interpreter**
- **In conclusion very few separates Node.js and Go in real-life performance**
- **Although in real life, the raw performance would not be the only parameter we would compare the two**

Node.js vs Python

- **Event-driven architecture allows Node.js servers to process more requests**
- **Performance of Python is not really its strong point**
  - especially with Django
- **Python require more resources to work at acceptable speeds**
- **Both has many diverse libraries**
- **But Python is ahead by little in terms of libraries**

Node.js vs PHP

- Unlike node PHP is synchronous
- PHP was developed much earlier
- PHP has slow page-load time
- PHP blocks a process till it’s been entirely calculated so has the blocking i/o
- One upside of PHP is that it’s an old language meaning it has much more refined content for learning

Node.js vs Java

- Java is slower than native languages since it uses compilers
- It’s garbage collector requires lots of memory space
- Node.js never buffer, outputting the data in chunks and making runtime faster.
- Just-In-Time compilers may improve Java’s performance a bit

Thank You For Listening

- https://nodejs.org/en/docs/
- https://en.m.wikipedia.org/wiki/Npm_(software)
- https://en.m.wikipedia.org/wiki/Node.js
- https://en.m.wikipedia.org/wiki/Asynchronous_I/O
- https://developer.mozilla.org/en-US/docs/Learn/Server-side/Express_Nodejs/Introduction
- https://www.simform.com/bl-frameworks/
- https://stackabuse.com/executing-shell-commands-with-node-js/
- https://www.geeksforgeeks.org/how-to-open-node-js-command-prompt/amp/
- https://github.com/goldbergyoni/nodebestpractices
- https://github.com/amkurian/NodeJs-Code-Snippets
- https://snyk.io/blog/peacenotwar-malicious-npm-node-ipc-package-vulnerability/
- https://jfrog.com/blog/large-scale-npm-attack-targets-azure-developers-with-malicious-packages/
- https://www.cisecurity.org/advisory/a-vulnerability-in-an-npm-package-could-allow-for-remote-code-execution_2021-136
- https://docs.npmjs.com/cli/v8/commands/npm-audit
- https://qz.com/646467/how-one-programmer-broke-the-internet-by-deleting-a-tiny-piece-of-code/amp/
- https://www.mindinventory.com/blog/nodejs

https://www.dropbox.com/s/5ek1ywkz2sh50xu/3.mp4

https://drive.google.com/file/d/1dD0_iuyCaqZKVGltopjq7XB0Dc8xplPy/view

200709006 - Affan Selim KAYA
200709011 - Onur Alp AKIN
190709009 - Emre ERZURUM