# Node.js

## Introduction to Node.js

## Just before Node...

## What are JavaScript Engines?

- They are compilers, they turn JavaScript into machine code
- Machine Code is code for micro-processors

## What JavaScript Engines are around?

- V8
- SpiderMonkey
- Chakra
- Nitro
- Lots of others…

## Why are we talking about this?

- Node.js is the JavaScript engine (V8)
  - Taken out of the browser, with a little bit extra
- It's more or less JavaScript, but as a back-end programming language
- It can do the same sort of things as Ruby can do

## Node

## A little bit of history...

- Written in 2009
- By Ryan Dahl, who was working at a company called Joyent

## What is Node good at?

- Streaming data
- "Soft" real-time
- Every developer knows a bit of JS
- "Tooling"
- Unopinionated and flexible
- It can be isomorphic
- Corporate backing
- Performant
- Good community - easy to find developers

## What is Node not so good at?

- "Tooling" - too much out there
- Unopinionated
- Still quite young
- Memory leaks
- Dependency injection
- Serving static files (like HTML)
- Providing structure
- Generic JS woes (callbacks etc.)
- Debugging
- Heavy applications
- Providing functionality out of the box
- Getting away from fancy buzzwords

## How to run it?

```
node
```

```
node filename.js
```

## How to make Node reusable?

- Modules
- Exports
    - `module.exports`
- Imports
    - `require`

## NPM: Node Package Manager

- NPM home page
- NPM official introduction
- Good visualisation here
- Dependency visualisation here

Kind of like Ruby Gems

The largest open-source ecosystem of code in history

Think of them as libraries, gems or packages

## It uses dependencies

A dependency is code that other code relies on to run

- An example of a depency tree

## A Node Project Overview

- `node_modules`
- `package.json`
- `package-lock.json`

## Let's make a new Node project

```
mkdir our-first-node-project
cd our-first-node-project
mkdir src
touch src/index.js
npm init
```

## How to add dependencies

```
npm install --save package
npm install --save-dev package
```

## Let's add some scripts

```
"scripts": {
  "start" : "node app.js"
}
```

## Express

## What is Express?

- Express adds routing to our applications

## Adding Express

```
npm install --save express
```

## Using Express: Set up Server

```
const app = express();

const port = 3000;

app.listen(port, () => {
  console.log(`Server running. Visit http://localhost:${port}`);
});
```

## Using Express: Routes

```
const app = express();

app.get("/", (req, res) => {
  console.log("GET request to /");
});

app.get("/home", (req, res) => {
  res.send("GET request to /home");
});

app.get("/about", (req, res) => {
  res.send("GET request to /about");
});
```

## Using Express: Dynamic Routes

```
const app = express();

app.get("/hello/:name", (req, res) => {
  const { name } = req.params;
  res.send(`Hello ${name}`);
});
```

## Adding HTML

We are going to be using "EJS"

It's very similar to ERB, but allows us to embed JS code in HTML

## Adding HTML

```
mkdir views
mkdir public
```

## Adding HTML

```
const path = require("path");

// ...

const app = express();
app.set("view engine", "ejs");
app.set("views", path.join(__dirname, "..", "views"));
app.use(express.static(path.join(__dirname, "..", "public")));
```

## Adding HTML

```
app.get("/", (req, res) => {
  res.render("pages/index");
});

app.get("/hello/:name", (req, res) => {
  res.render("pages/dynamic", { name: req.params.name });
});
```

## Adding a Database

## Adding Packages

```
npm install --save body-parser sequelize sequelize-cli sqlite3
```

## Initializing Sequelize

```
node_modules/.bin/sequelize init
```

## Configuring your Database

*config/config.json*

```json
{
  "development": {
    "dialect": "sqlite",
    "storage": "./database.sqlite3"
  },
  "test": {
    "dialect": "sqlite",
    "storage": ":memory"
  },
  "production": {
    "dialect": "sqlite",
    "storage": "./database.sqlite3"
  }
}
```

## eating Models

```
le_modules/.bin/sequelize model:generate --name User --attributes firstName:string,lastName:string,email:str
```

## Migrating

```
node_modules/.bin/sequelize db:migrate
```

## Creating a Seeds File

```
node_modules/.bin/sequelize seed:generate --name seed-user
```

## Seeding

```
node_modules/.bin/sequelize db:seed:all
```

## Database Routes

## Getting all Users

```
const db = require("../models");

app.get("/users", (req, res) => {
  db.User.findAll().then(users => res.render("users/index", { users }));
});
```

## Getting one User

```
app.get("/users/:id", (req, res) => {
  db.User.findByPk(parseInt(req.params.id, 10)).then(user =>
    res.render("users/show", { user })
  );
});
```

## Showing the New Form for a User

```
app.get("/users/new", (req, res) => {
  res.render("users/new");
});
```

## Handling the submission of a form

```
app.use(bodyParser.urlencoded({ extended: true }));
app.use(bodyParser.json());

// ...

app.post("/users", (req, res) => {
  const { firstName, lastName, email } = req.body;
  db.User.create({ firstName, lastName, email }).then(u =>
    res.redirect(`/users/${u.id}`)
  );
});
```

## Deleting a User

```
app.get("/users/:id/delete", (req, res) => {
  const id = parseInt(req.params.id, 10);
  db.User.findByPk(id)
    .then(user => user.destroy({ force: true }))
    .then(() => res.redirect("/users"));
});
```

## Editing a User

```
app.get("/users/:id/edit", (req, res) => {
  db.User.findByPk(parseInt(req.params.id, 10)).then(user =>
    res.render("users/edit", { user })
  );
});
```

## Updating a User

```
app.post("/users/:id", (req, res) => {
  const id = parseInt(req.params.id, 10);
  const { firstName, lastName, email } = req.body;
  db.User.update({ firstName, lastName, email }, { where: { id } }).then(() => {
    res.redirect(`/users/${id}`);
  });
});
```