Welcome to the

CoGrammar

Introduction to Node.js and setting up an ExpressJS server

The session will start shortly...

Questions? Drop them in the chat. We'll have dedicated moderators answering questions.

CoGrammar

Full Stack Web Development Session Housekeeping

- The use of disrespectful language is prohibited in the questions, this is a supportive, learning environment for all - please engage accordingly. **(Fundamental British Values: Mutual Respect and Tolerance)**
- No question is daft or silly - **ask them!**
- There are **Q&A sessions** midway and at the end of the session, should you wish to ask any follow-up questions. Moderators are going to be answering questions as the session progresses as well.
- If you have any questions outside of this lecture, or that are not answered during this lecture, please do submit these for upcoming Academic Sessions. You can submit these questions here: Questions

Full Stack Web Development Session Housekeeping cont.

- For all **non-academic questions**, please submit a query: www.hyperiondev.com/support

- Report a **safeguarding** incident: www.hyperiondev.com/safeguardreporting

- We would love your **feedback** on lectures: [Feedback on Lectures]()

Safeguarding & Welfare

We are committed to all our students and staff feeling safe and happy; we want to make sure there is always someone you can turn to if you are worried about anything.

If you are feeling upset or unsafe, are worried about a friend, student or family member, or you feel like something isn't right, speak to our safeguarding team:

Ian Wyles
Designated Safeguarding Lead

Simone Botes

Nurhaan Snyman

Rafiq Manan

Ronald Munodawafa

Tevin Pitts

Scan to report a safeguarding concern

or email the Designated Safeguarding Lead:
Ian Wyles
safeguarding@hyperiondev.com

CoGrammar HyperionDev

SKILLS FOR LIFE

SKILLS BOOTCAMPS

Department for Education

CoGrammar

NodeJS and Express

January 2025

Polls

Please have a look at the poll notification and select an option.

Which of the following is used to handle asynchronous I/O in Node.js?

A. Promises
B. Callbacks
C. Event Emitters
D. Both Promises and Callbacks

CoGrammar

Polls

Please have a look at the poll notification and select an option.

Which of the following is used in Node.js to manage project dependencies and packages?

A. npm
B. git
C. webpack
D. gulp

CoGrammar

Lesson Objectives

- ❖ Explain the purpose of Express.js as a Node.js framework
- ❖ Explain the importance of npm (Node Package Manager) and its role in managing dependencies.
- ❖ Set up a basic Node.js application and install Express using npm.
- ❖ Identify the key features and components of Node.js and Express.

CoGrammar

NodeJS

Definition and Key Features

- Node.js is an open-source, cross-platform runtime environment that allows JavaScript to be run on the server side.
- **Key Features:**
  - **Asynchronous & Non-blocking I/O:** Efficient handling of I/O operations.
  - **Single-threaded Event Loop:** Handles multiple requests concurrently without multiple threads.
  - **Fast Execution:** V8 engine compiles JavaScript into machine code.
  - **npm (Node Package Manager):** Access to a vast ecosystem of libraries and tools.

CoGrammar

NodeJS

Use cases

Common Applications:

- ❖ **Web Servers & APIs:** Build fast, scalable web servers or RESTful APIs.
- ❖ **Real-time Applications:** Ideal for chat apps, live updates, etc.
- ❖ **Microservices:** Suitable for modular, lightweight service architectures.

Express.js

Definition and Use Cases

- Express is a minimal and flexible Node.js web application framework that provides a robust set of features for web applications.
- Express.js' main features include:
  - **Routing:** defines routes for handling different HTTP methods (GET, POST, PUT, DELETE).
  - **Middleware:** functions having access to request and response objects in the application.

CoGrammar

Express.js

Definition and Use Cases

- ➢ **Static File Serving:** built in middlewares in place for serving static files (HTML, CSS, JS, Images).
- ➢ **Creating APIs:** Easy creation of API endpoints for web applications. The endpoints can perform tasks such as interacting with a database e.t.c.

❖ Express.js' lightweight and unopinionated nature makes it popular among developers for building scalable web solutions

CoGrammar

Prerequisites for Express.js

- **Node.js:** make sure node.js is installed on your laptop
  - Confirm by running **node -v**
- **Code Editor:** preferably Visual Studio Code

CoGrammar

Configuring Node.js and Installing Express.js

CoGrammar

Installation and Configuration

- ❖ Create a folder where your application will live and change directory to it:
  - ➢ mkdir server
  - ➢ cd server
- ❖ Initialize your package.json file with the default settings:
  - ➢ npm init -y (The y is optional if you need to skip prompts)
- ❖ Install express.js:
  - ➢ npm install express

CoGrammar

Installation and Configuration

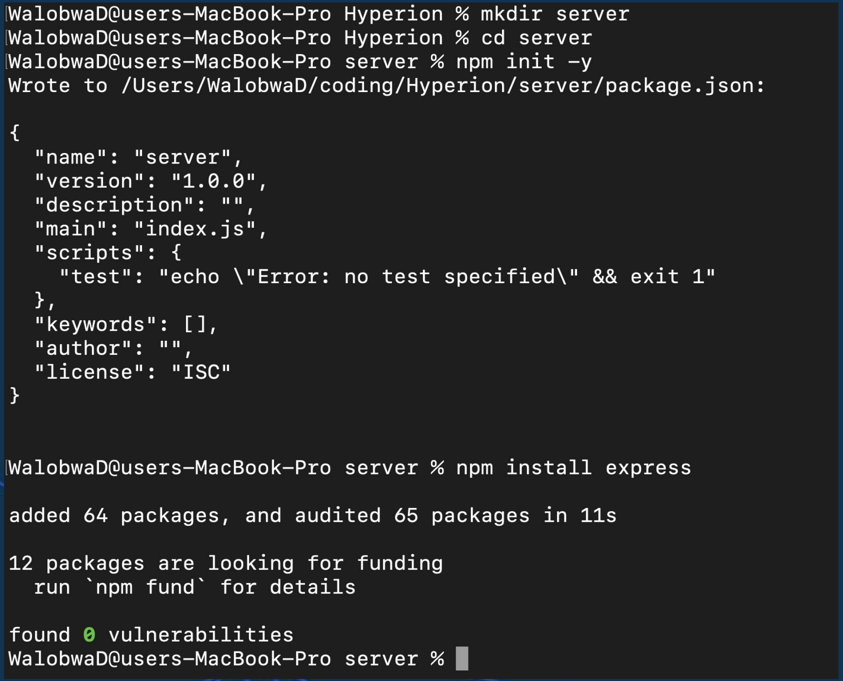
Setting up Express.js

- ❖ The commands executed should initialize a package.json file with predefined settings.
- ❖ After installing Express.js, the package name should be listed in the dependencies section of the package. json.
- ❖ All packages installed are stored in the node_modules folder. NOTE: Make sure the node_modules folder is .gitignored to avoid pushing it to github.

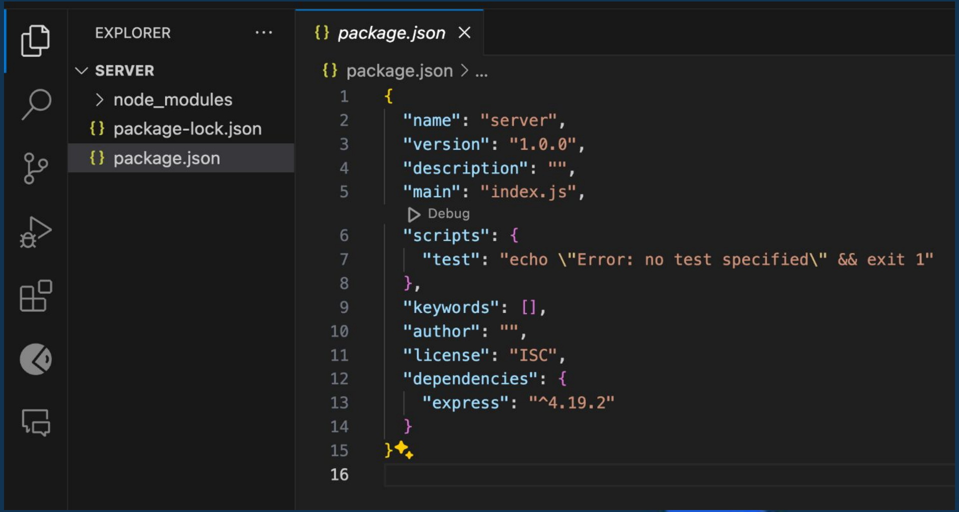
CoGrammar

Installation and Configuration

Note the express inside the dependencies.



```
WalobwaD@users-MacBook-Pro Hyperion % mkdir server
WalobwaD@users-MacBook-Pro Hyperion % cd server
WalobwaD@users-MacBook-Pro server % npm init -y
Wrote to /Users/WalobwaD/coding/Hyperion/server/package.json:

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


WalobwaD@users-MacBook-Pro server % npm install express

added 64 packages, and audited 65 packages in 11s

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

found 0 vulnerabilities
WalobwaD@users-MacBook-Pro server %
```



```
{
  "name": "server",
  "version": "1.0.0",
  "description": "",
  "main": "index.js",
  "scripts": {
    "test": "echo \"Error: no test specified\" && exit 1"
  },
  "keywords": [],
  "author": "",
  "license": "ISC",
  "dependencies": {
    "express": "^4.19.2"
  }
}
```

CoGrammar

Creating an Express.js Server

CoGrammar

Creating a server

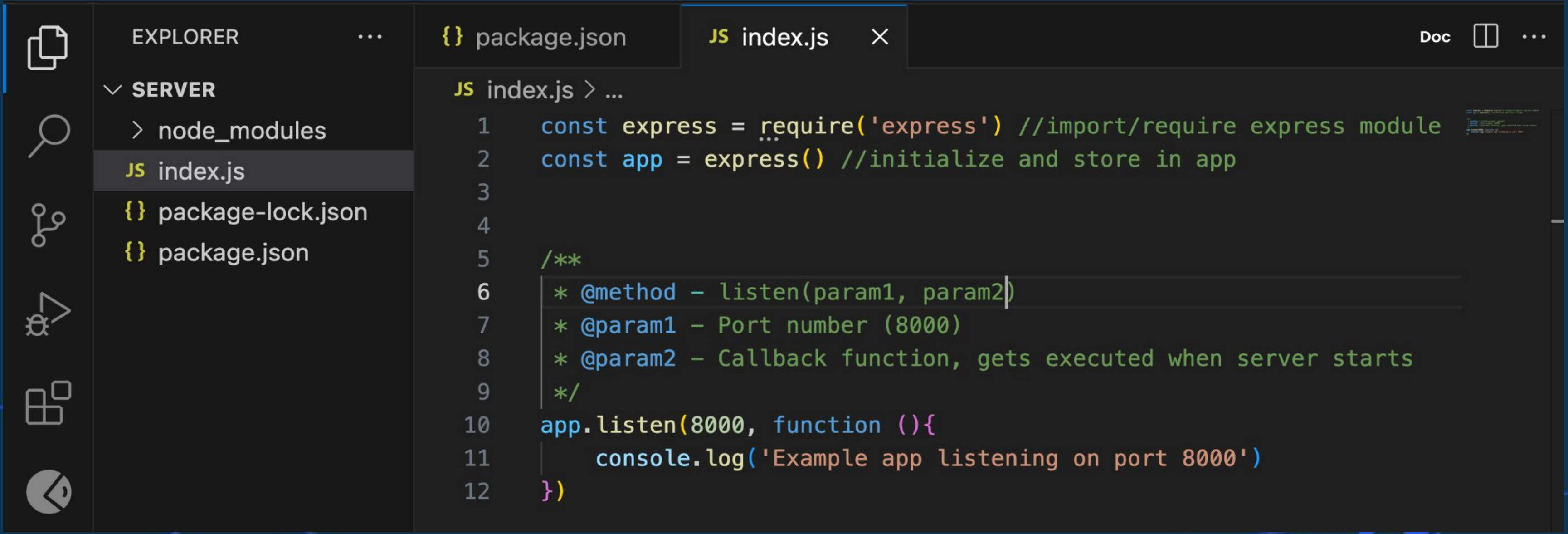
Running a port on your local machine

- ❖ From the configuration we just built, we can create an **index.js** file to act as your root file.
- ❖ We'll go ahead and import the express.js we just installed using common js syntax and reference it to a variable called app so whenever we need an express property, we'll use the app variable.
- ❖ The express module contains a **listen method** which takes in two arguments (**the port number** and **a callback function**). This will be the method to create the needed server for our app to run.

CoGrammar

Creating a server

Running a port on your local machine



```javascript
const express = require('express') //import/require express module
const app = express() //initialize and store in app


/**
 * @method - listen(param1, param2)
 * @param1 - Port number (8000)
 * @param2 - Callback function, gets executed when server starts
 */
app.listen(8000, function (){
    console.log('Example app listening on port 8000')
})
```

CoGrammar

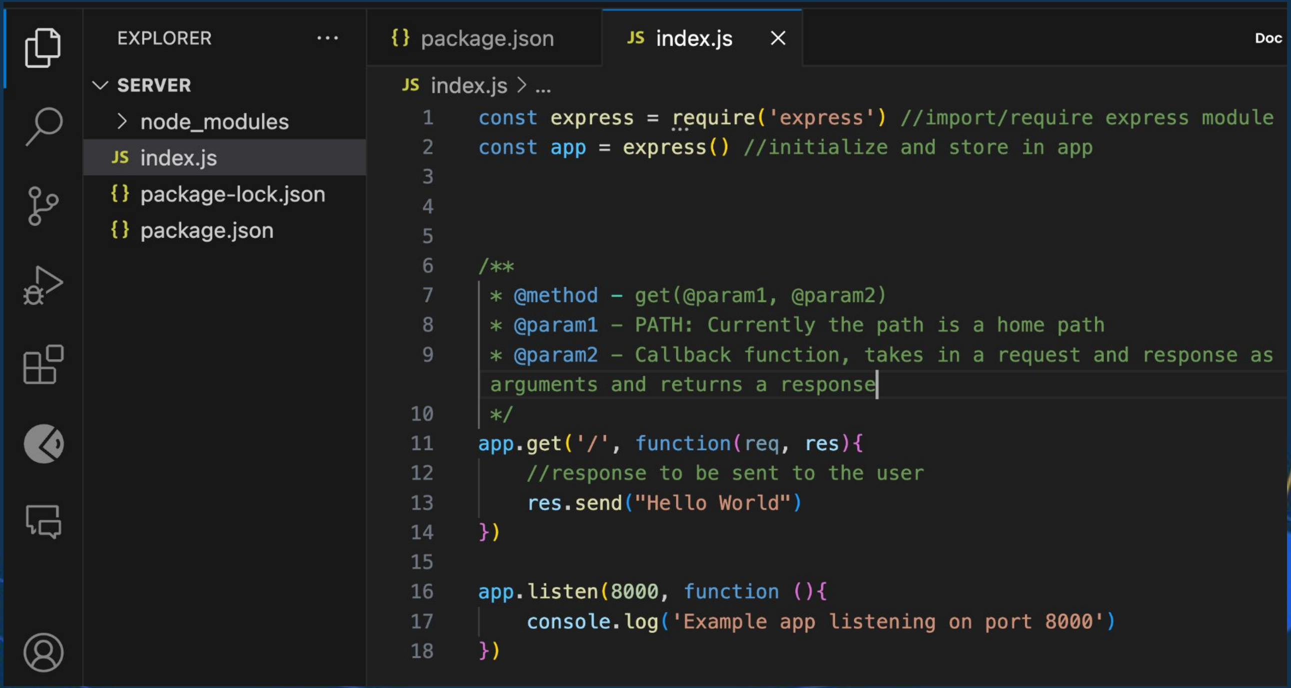
Creating a route for your application

- We’ll create our first path with the GET method.
- From the app variable, we can call the app.get() which takes in two main arguments. (**The path** and **a callback function**).
- The callback function in this case becomes the route handler, it determined the kind of response the user will get after making a request to a specific path on the server.

CoGrammar

Creating a server

Adding a start script to the server



```
const express = require('express') //import/require express module
const app = express() //initialize and store in app



/**
 * @method - get(@param1, @param2)
 * @param1 - PATH: Currently the path is a home path
 * @param2 - Callback function, takes in a request and response as
 arguments and returns a response
 */
app.get('/', function(req, res){
    //response to be sent to the user
    res.send("Hello World")
})

app.listen(8000, function (){
    console.log('Example app listening on port 8000')
})
```

CoGrammar

Creating a server

Adding a start script to the server

- ❖ We now need to start our server, you can run it directly using Node.js by executing: **`node index.js`** on the terminal.
- ❖ Instead we're going to use a library called nodemon to assist.
  - ➢ Nodemon is a tool that helps develop Node.js based applications by automatically restarting the node application when file changes in the directory are detected.
- ❖ We need to install it in order to use it using the command:

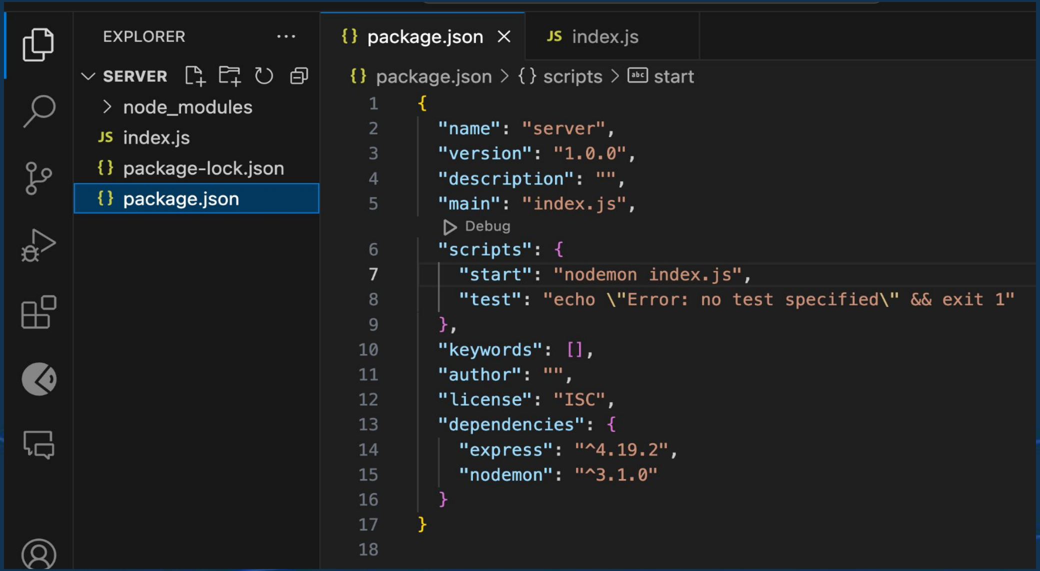
```
npm install nodemon
```

CoGrammar

Creating a server

Adding a start script to the server

❖ After installing nodemon, in your package.json file, you can insert a “start” property inside your scripts object and include the text: **nodemon {nameOfFile}**



```json
{
  "name": "server",
  "version": "1.0.0",
  "description": "",
  "main": "index.js",
  "scripts": {
    "start": "nodemon index.js",
    "test": "echo \"Error: no test specified\" && exit 1"
  },
  "keywords": [],
  "author": "",
  "license": "ISC",
  "dependencies": {
    "express": "^4.19.2",
    "nodemon": "^3.1.0"
  }
}
```

CoGrammar

Creating a server

Adding a start script to the server

- ❖ You can now run the project using

```
npm start
```

- ❖ At the moment from the configuration done so far, you'll be able to see a **"Hello World"** text being displayed on the UI.
- ❖ This means the server is rendering a response saying Hello World when the user requests for the home path of the website.

CoGrammar

Questions and Answers

CoGrammar

Thank you for attending

SKILLS FOR LIFE
SKILLS BOOTCAMPS

Department for Education

CoGrammar