




Welcome to the CoGrammar Lecture: Node.js

The session will start shortly...

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



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](#)

Skills Bootcamp

8-Week Progression Overview

Fulfil 4 Criteria to Graduation

✓ Criterion 1: Initial Requirements

Timeframe: First 2 Weeks

Guided Learning Hours (GLH):

Minimum of 15 hours

Task Completion: First four tasks

Due Date: 24 March 2024

✓ Criterion 2: Mid-Course Progress

60 Guided Learning Hours

Data Science - **13 tasks**

Software Engineering - **13 tasks**

Web Development - **13 tasks**

Due Date: 28 April 2024

Skills Bootcamp Progression Overview

✓ Criterion 3: Course Progress

Completion: All mandatory tasks,
including Build Your Brand and
resubmissions by study period end
Interview Invitation: Within 4 weeks
post-course
Guided Learning Hours: Minimum of
112 hours by support end date
(10.5 hours average, each week)

✓ Criterion 4: Demonstrating Employability

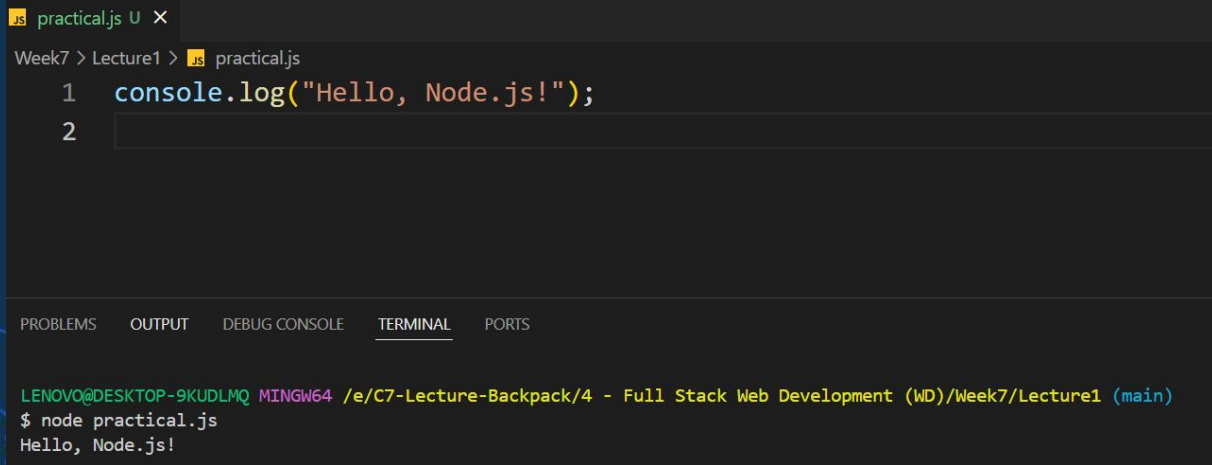
Final Job or Apprenticeship
Outcome: Document within 12
weeks post-graduation
Relevance: Progression to
employment or related
opportunity

Learning Objectives

- ❖ Explore the concept of modules in Node.js and learn how to create, import, and use modules effectively.
- ❖ Gain familiarity with NPM (Node Package Manager) and its role in managing dependencies, versioning, and scripts.
- ❖ Develop proficiency in setting up a basic Node.js server using the built-in http module.

What is Node.js?

- ❖ Node.js is a runtime environment that allows you to run JavaScript code on the server-side.
- ❖ It uses an event-driven, non-blocking I/O model, making it efficient for handling asynchronous operations.



```
practical.js U X
Week7 > Lecture1 > practical.js
1 console.log("Hello, Node.js!");
2
PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS
LENOVO@DESKTOP-9KUDLMQ MINGW64 /e/C7-Lecture-Backpack/4 - Full Stack Web Development (WD)/Week7/Lecture1 (main)
$ node practical.js
Hello, Node.js!
```



What are Modules?

- ❖ Modules in Node.js are encapsulated units of functionality that can be reused throughout your application.
- ❖ They promote code organization, maintainability, and reusability.
- ❖ Node.js implements the CommonJS module system, allowing modules to be defined using `require()` and exported using `module.exports`.



Creating and Using Modules

- ❖ Modules in Node.js are encapsulated units of functionality that can be reused throughout your application.
- ❖ They promote code organization, maintainability, and reusability.

```
const greet = () => {  
  console.log("Hello, world!");  
};  
module.exports = greet;
```

```
const greet = require("./greet");  
greet();
```



Core Modules vs. User-defined Modules

- ❖ Node.js provides several core modules like http, fs, and path, which can be used without installation.
- ❖ User-defined modules are created by developers to encapsulate specific functionality.



NPM (Node Package Manager)

- ❖ NPM is the default package manager for Node.js, used for installing, managing, and sharing packages of JavaScript code.
- ❖ It provides access to a vast repository of open-source packages and tools for Node.js development.



Let's Breathe!

Let's take a small break
before moving on to
the next topic.



Managing Dependencies with NPM

- ❖ Define project dependencies in the package.json file.
- ❖ Use npm install to install dependencies listed in package.json.

```
$ npm install express
```



Creating a package.json File

- ❖ Use **npm init** to generate a package.json file interactively or with default values.
- ❖ **package.json** serves as the manifest for your project, documenting project metadata, dependencies, and scripts.

Understanding package.json Structure

- ❖ **name:** The name of the project.
- ❖ **version:** The version of the project.
- ❖ **dependencies:** List of project dependencies and their version specifications.
- ❖ **scripts:** Custom scripts for tasks like testing, building, and deployment.

Understanding package.json Structure

```
{
  "name": "my-node-app",
  "version": "1.0.0",
  "dependencies": {
    "express": "^4.17.1"
  },
  ▶ Debug
  "scripts": {
    "start": "node index.js"
  }
}
```


Managing Scripts in package.json

- ❖ Use the **scripts** field in **package.json** to define custom scripts.
- ❖ Scripts can be executed using **npm run <script-name>**.

```
"scripts": {  
  "start": "node index.js",  
  "test": "mocha"  
}
```

Setting Up a Node.js Server

- ❖ Use the built-in http module to create an HTTP server.
- ❖ Listen for incoming requests on a specified port and handle them accordingly.

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

const server = http.createServer((req, res) => {
  res.writeHead(200, { "Content-Type": "text/plain" });
  res.end("Hello, world!\n");
});

const PORT = process.env.PORT || 3000;
server.listen(PORT, () => {
  console.log(`Server is running on port ${PORT}`);
});
```

Questions and Answers



Thank you for attending



Department
for Education

CoGrammar

