CoGrammar

Welcome to this session:
Tutorial - Node.js

The session will start shortly...

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

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 Bootcamp Cloud Web Development

- 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**

Skills Bootcamp Cloud Web Development

- 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
- If you are hearing impaired, please kindly use your computer's function through Google chrome to enable captions.

CoGrammar

Node.js

Tutorial Outcomes

- Initialize and set up an NPM package by installing dependencies and creating custom scripts.
- Utilise built-in modules, fs (file system), http (HTTP server), to enhance application functionality.
- Demonstrate the use of package functions to manipulate data effectively within a Node.js environment.
- Create and execute custom NPM scripts to automate tasks and streamline project workflows.

Tutorial Overview

- → Introduction to Node.js
- → Introduction to NPM
- → Tutorial using the http and fs modules

What is the primary purpose of the HTTP module in Node.js?

A. To make HTTP requests to other servers.
B. To handle routing and manage URL paths.
C. To create and handle HTTP servers for web applications.
D. To parse and manipulate HTTP headers.
E. To handle HTTPS and secure communication.

CoGrammar

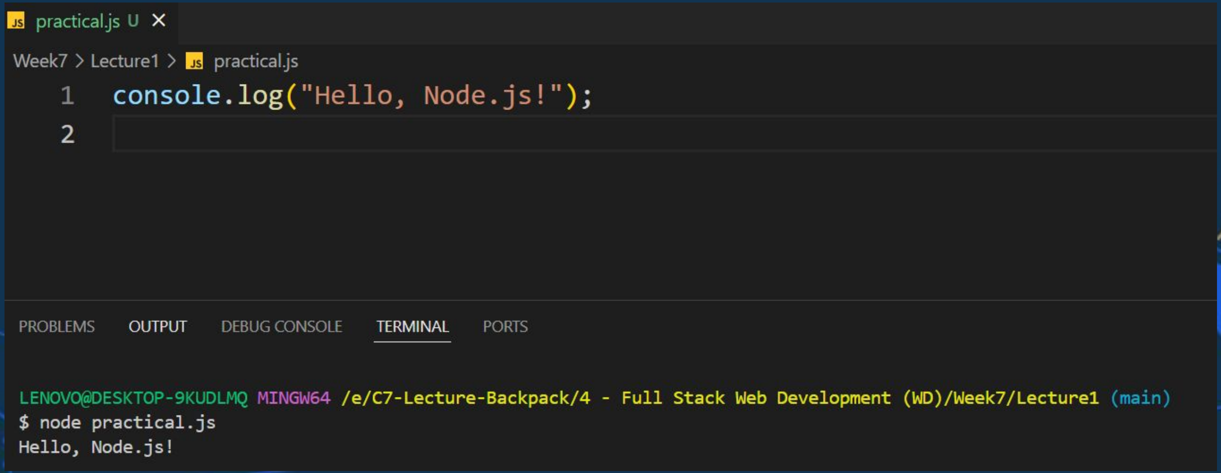
What is the primary purpose of the *fs* module in Node.js?

A. To handle HTTP requests and responses.
B. To interact with the file system (read, write, update files).
C. To manage user authentication and sessions.
D. To enable WebSocket communication between clients and servers.
E. To parse JSON data from files.

CoGrammar

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.



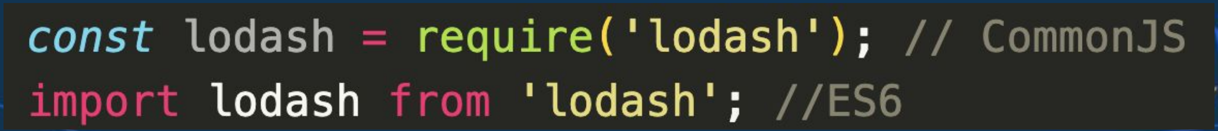
```javascript
console.log("Hello, Node.js!");
```

```
LENOVO@DESKTOP-9KUDLMQ MINGW64 /e/C7-Lecture-Backpack/4 - Full Stack Web Development (WD)/Week7/Lecture1 (main)
$ node practical.js
Hello, Node.js!
```

CoGrammar

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 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.



```
const lodash = require('lodash'); // CommonJS
import lodash from 'lodash'; //ES6
```

CoGrammar

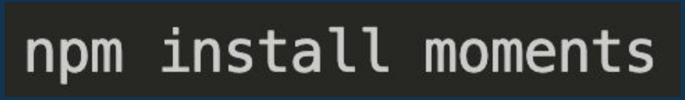
Node Package Manager

- 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.

CoGrammar

Managing Dependencies with NPM

- ❖ Use npm init to initialize a new NPM package in your project directory and 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.
- ❖ Define project dependencies in the package.json file.
- ❖ Use npm install to install dependencies listed in package.json.



```
npm install moments
```

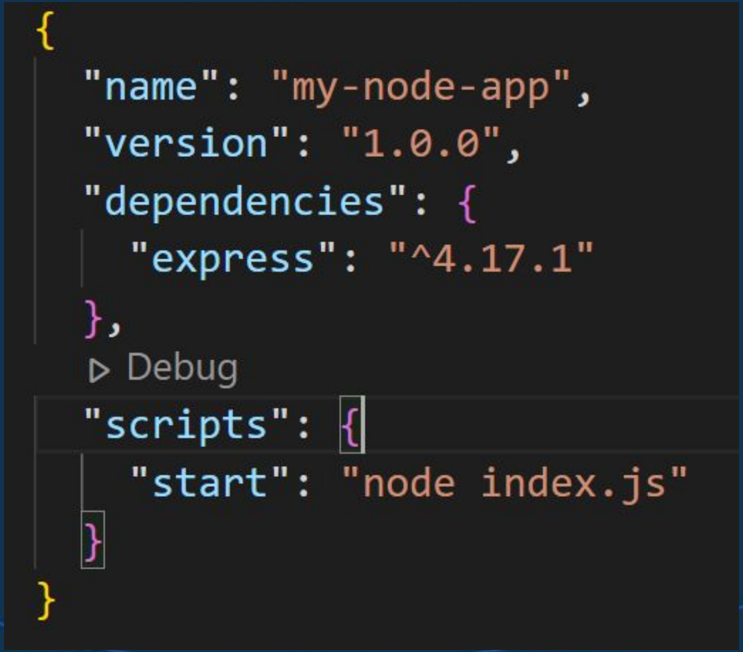
CoGrammar

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.

CoGrammar

Understanding package.json Structure

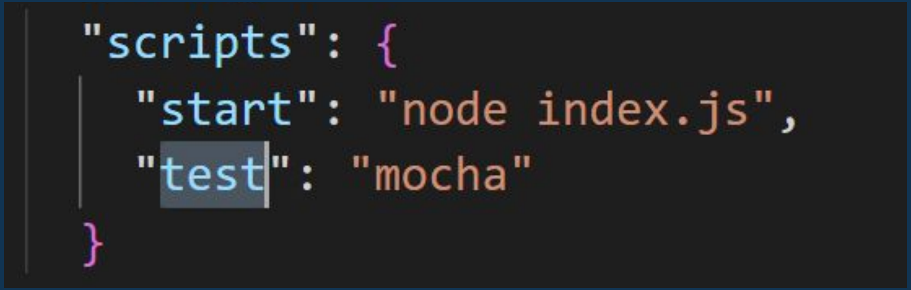


```
{
  "name": "my-node-app",
  "version": "1.0.0",
  "dependencies": {
    "express": "^4.17.1"
  },
  ▷ Debug
  "scripts": {
    "start": "node index.js"
  }
}
```

CoGrammar

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"
}
```

CoGrammar

The HTTP module

❖ This module contains all the code needed to use Node.js to transfer data using the HTTP protocol.

```
const http = require('node:http');  // Newer versions of Node.js
const http = require('http');           // Older versions of Node.js

http.createServer((request, response) => {
   response.write('Hello World!');
   response.end();
}).listen(3000);
```

The FS (File System) module

- This module cenables interacting with the file system.
- Some of the methods in this module:
  - fs.open()
  - fs.readFile()
  - fs.writeFile()
  - fs.appendFile()
  - fs.rename()
  - fs.unlink()

```
  1.    const { unlink } = require('node:fs');  // Newer versions of Node.js
  2.    const { unlink } = require('fs');          // Older versions of Node.js
  3.    const fs = require('fs');                    // Importing the entire module

unlink('/tmp/hello', (err) => {
  if (err) throw err;
  console.log('successfully deleted /tmp/hello');
});
```

What is the purpose of the createServer method within the HTTP module in Node.js?

A. To create a new HTTP request.
B. To create an HTTP server that listens for incoming requests.
C. To handle responses from HTTP clients.
D. To manage HTTP headers and cookies.
E. To send HTTP requests to other servers.

CoGrammar

Which Node.js fs method is used to delete a resource?

A. fs.unlink();
B. fs.remove();
C. fs.delete();
D. fs.destroy();
E. fs.rmdir();

CoGrammar

Let's take a break

CoGrammar

Questions and Answers

CoGrammar

Thank you for attending

CoGrammar

SKILLS FOR LIFE
SKILLS BOOTCAMPS

Department for Education