Welcome to the

CoGrammar

Lecture: The MERN Stack Back-end Integration with React

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]()

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

CoGrammar

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

CoGrammar

Lesson Objectives

- ❖ Initialise a Node.js project. configure it with Express.js & CORS.
- ❖ Initialise a React.js project, set up Axios for HTTP requests, and configure the proxy for back-end communication.
- ❖ Create new API endpoints in the back-end, fetch data from these endpoints in the React.js front-end, and manage application state to display the fetched data.
- ❖ Simultaneously run and debug both the back-end and front-end servers to ensure seamless integration and correct data flow between them.

CoGrammar

Review and Recap

- In the passed few lectures we've learnt how to do **back-end development with Express.js**. This involved:
  - **Routing:** creating a server, handling HTTP messages.
  - **MongoDB:** interacting with databases using Mongoose.
- Earlier in our course, we looked at creating an **interactive and dynamic front-end with React**.
- The goal of today's lecture is to integrate our React.js front-end server with our Express.js back-end server.

CoGrammar

Express.js: Back-end

1. Create a new project directory and change directory to it.

```
mkdir backend
cd backend
```

2. Initialise NPM so that dependencies can be installed.

```
npm init -y
```

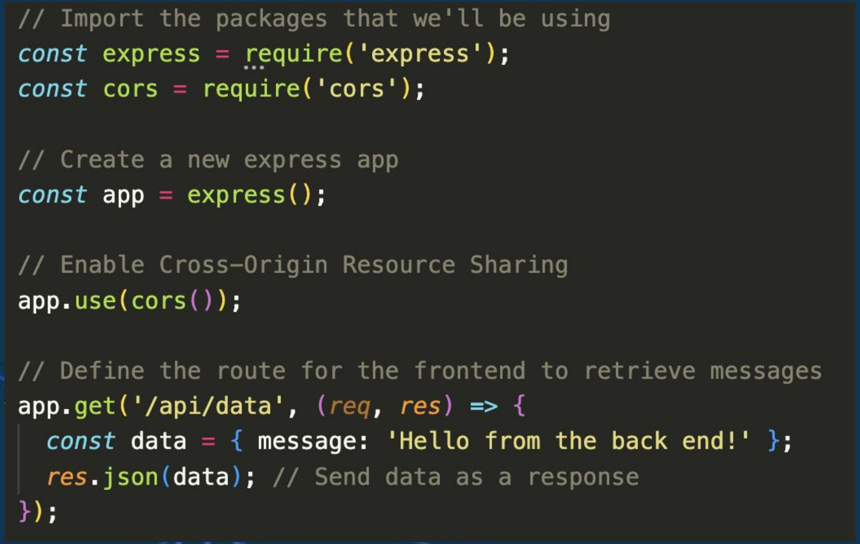
3. Install Express, for creating the server and routing, and CORS, for cross-origin resource sharing. This allows web pages located on one domain to access restricted resources on a different domain.

```
npm install cors express
```

CoGrammar

Express.js: Back-end

4. Create a JavaScript file which will contain the code needed to create a server. We will do this in a file called server.js.

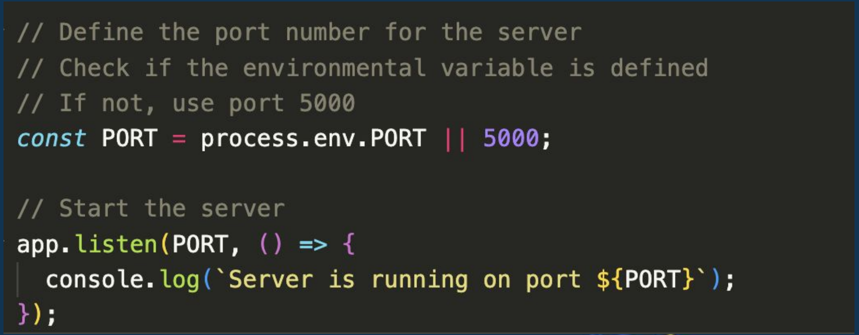


```
// Import the packages that we'll be using
const express = require('express');
const cors = require('cors');

// Create a new express app
const app = express();

// Enable Cross-Origin Resource Sharing
app.use(cors());

// Define the route for the frontend to retrieve messages
app.get('/api/data', (req, res) => {
  const data = { message: 'Hello from the back end!' };
  res.json(data); // Send data as a response
});
```



```
// Define the port number for the server
// Check if the environmental variable is defined
// If not, use port 5000
const PORT = process.env.PORT || 5000;

// Start the server
app.listen(PORT, () => {
  console.log(`Server is running on port ${PORT}`);
});
```

CoGrammar

Let’s Breathe!

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

CoGrammar

React.js: Front-end

1. Initialise the React App by running this command in the root directory

```
npx create-react-app frontend
```

2. Add the following line to the package.json file in the frontend directory, to set the default local host (the same port as the back-end).

```
"proxy": "http://localhost:5000"
```

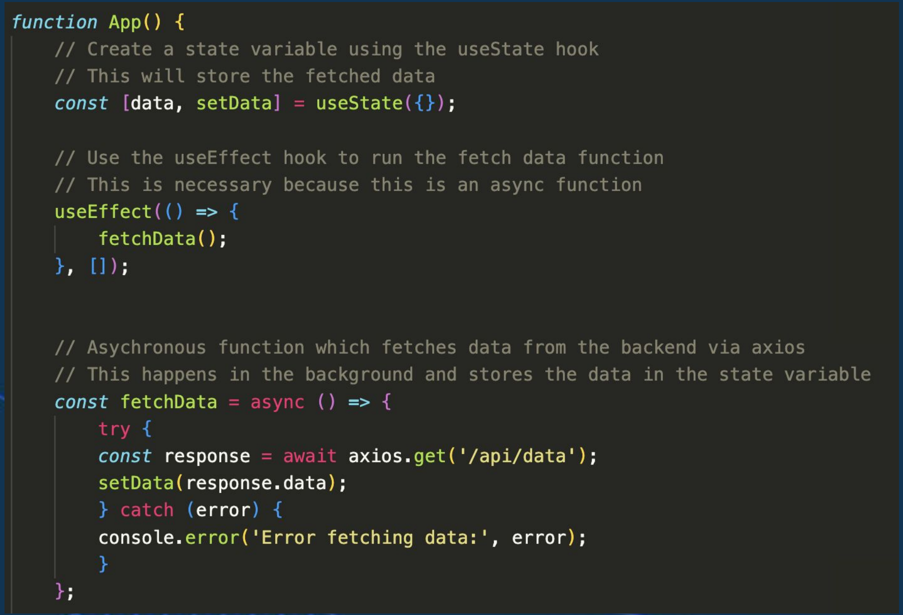
3. Install the Axios library which facilitates HTTP requests from our React app to our back-end.

```
npm install axios
```

CoGrammar

React.js: Front-end

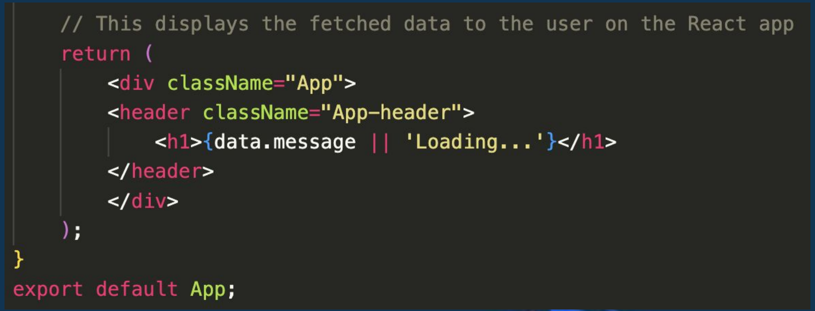
4. Update the App.js file in your front-end directory to include the code which sends an API request to the backend.



```
function App() {
    // Create a state variable using the useState hook
    // This will store the fetched data
    const [data, setData] = useState({});

    // Use the useEffect hook to run the fetch data function
    // This is necessary because this is an async function
    useEffect(() => {
        fetchData();
    }, []);


    // Asychronous function which fetches data from the backend via axios
    // This happens in the background and stores the data in the state variable
    const fetchData = async () => {
        try {
        const response = await axios.get('/api/data');
        setData(response.data);
        } catch (error) {
        console.error('Error fetching data:', error);
        }
    };
```

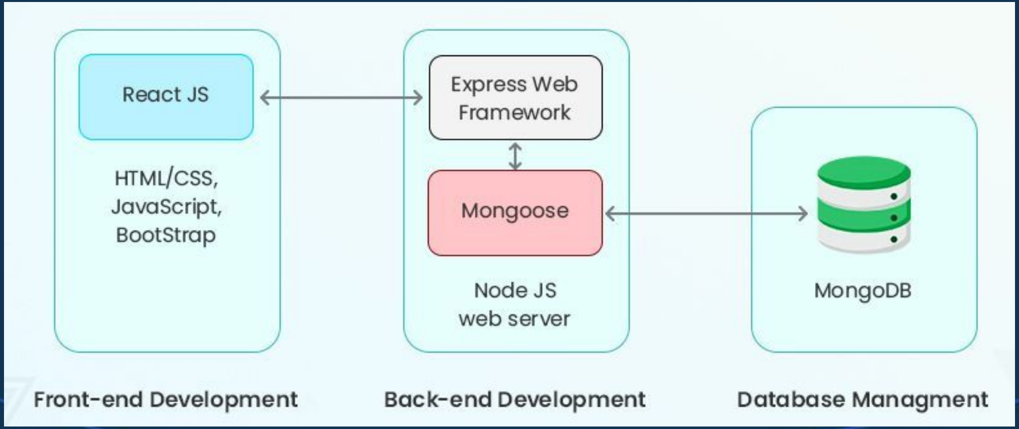


```
    // This displays the fetched data to the user on the React app
    return (
        <div className="App">
        <header className="App-header">
            <h1>{data.message || 'Loading...'}</h1>
        </header>
        </div>
    );
}
export default App;
```

CoGrammar

Run the Application

- ❖ Run the front-end and back-end servers on separate terminals. The web application can now be accessed through your browser.
- ❖ This will be the basis for all our full stack web applications.



Source: Radix

CoGrammar

Questions and Answers

CoGrammar

Thank you for attending

SKILLS FOR LIFE

SKILLS BOOTCAMPS

Department for Education

CoGrammar