HyperionDev

Full-Stack Web Development

HyperionDev

Connecting React Frontend with Express Backend

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**

Objectives

- ❖ Understand how to connect a React frontend with an Express backend.
- ❖ Fetch data from the Express API in a React application.
- ❖ Display fetched data in React components.
- ❖ Handle errors in data fetching.

Initial Assessment

- ❖ Have you connected a frontend to a backend before? (Yes/No)
- ❖ How comfortable are you with fetching data in React? (1-5 scale)
- ❖ What challenges have you faced in integrating frontends with backends? (Open-ended)

Introduction

Understanding the connection between frontend and backend is essential for building full-stack applications.

Two Parts of the Same Whole: Think of a web application as a car. The frontend is like the dashboard and controls that the driver interacts with, while the backend is the engine and machinery that makes the car run. For the car (or application) to work properly, both parts need to communicate effectively.

The frontend is where users see and interact with the application. It displays information and receives input from users. The backend processes this input, stores data, and performs calculations. If the frontend and backend can’t communicate well, users won’t get the information they need or may not be able to save their data properly.

Data Flow between frontend and backend

- **User Interaction**:
  - A user interacts with the frontend (e.g., clicks a button or submits a form).
- **Making a Request**:
  - The frontend sends an HTTP request (like GET or POST) to the backend to fetch or send data.
- **Backend Processing**:
  - The backend processes the request, validating it and possibly interacting with a database to retrieve or store data.
- **Sending a Response**:
  - The backend sends a response back to the frontend, usually in JSON format, containing the requested data or a success message.
- **Updating the Frontend**:
  - The frontend updates the user interface based on the response, displaying the new data or any messages.
- **Handling Errors**:
  - If an error occurs, the frontend displays an appropriate message to the user, ensuring a smooth experience.

CORS (Cross-Origin Resource Sharing)

- **Security**:
  - CORS helps prevent malicious sites from stealing data by enforcing a "same-origin policy" that restricts access to data from different domains.
- **Data Sharing**:
  - It allows developers to specify which domains can access their resources, enabling safe data sharing between applications.
- **Flexibility**:
  - CORS enables developers to create applications that communicate with multiple servers or APIs without compromising security.
- **Error Handling**:
  - If a request is made without proper permissions, the browser blocks it and shows an error, helping developers identify and fix issues.

Demonstration - Setting Up the React Project

Use CRA (create-react-app) to create a react frontend.

Setting up a proxy

Setting up routes from the frontend to the backend.

Fetching data

https://www.freecodecamp.org/news/how-to-fetch-data-from-an-api-using-the-fetch-api-in-javascript/

HTTP METHODS

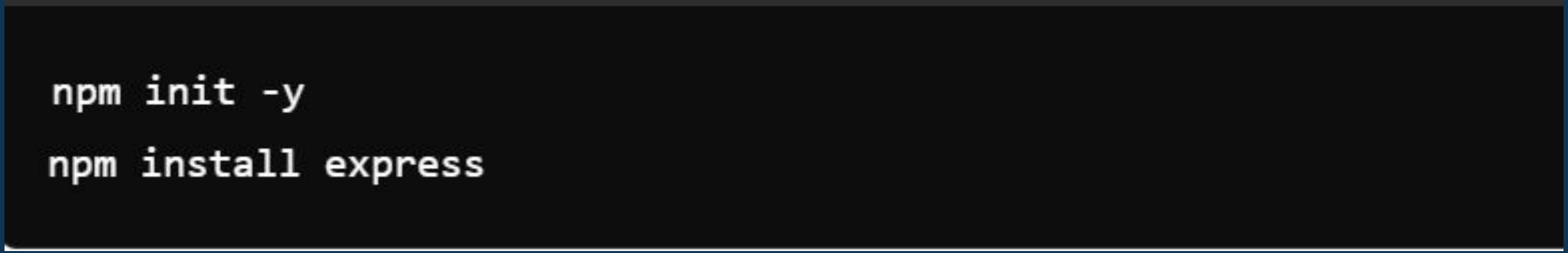
- **Concept:**
  - Use standard HTTP methods to perform operations on resources
- **Methods:**
  - GET: Retrieve a resource
  - POST: Create a new resource
  - PUT: Update an existing resource
  - DELETE: Remove a resource
  -
- **Benefits:**
  - Predictability: Consistent behaviour across APIs
  - Interoperability: HTTP is universally supported

Q&A

- Please ask any questions if you want any clarification

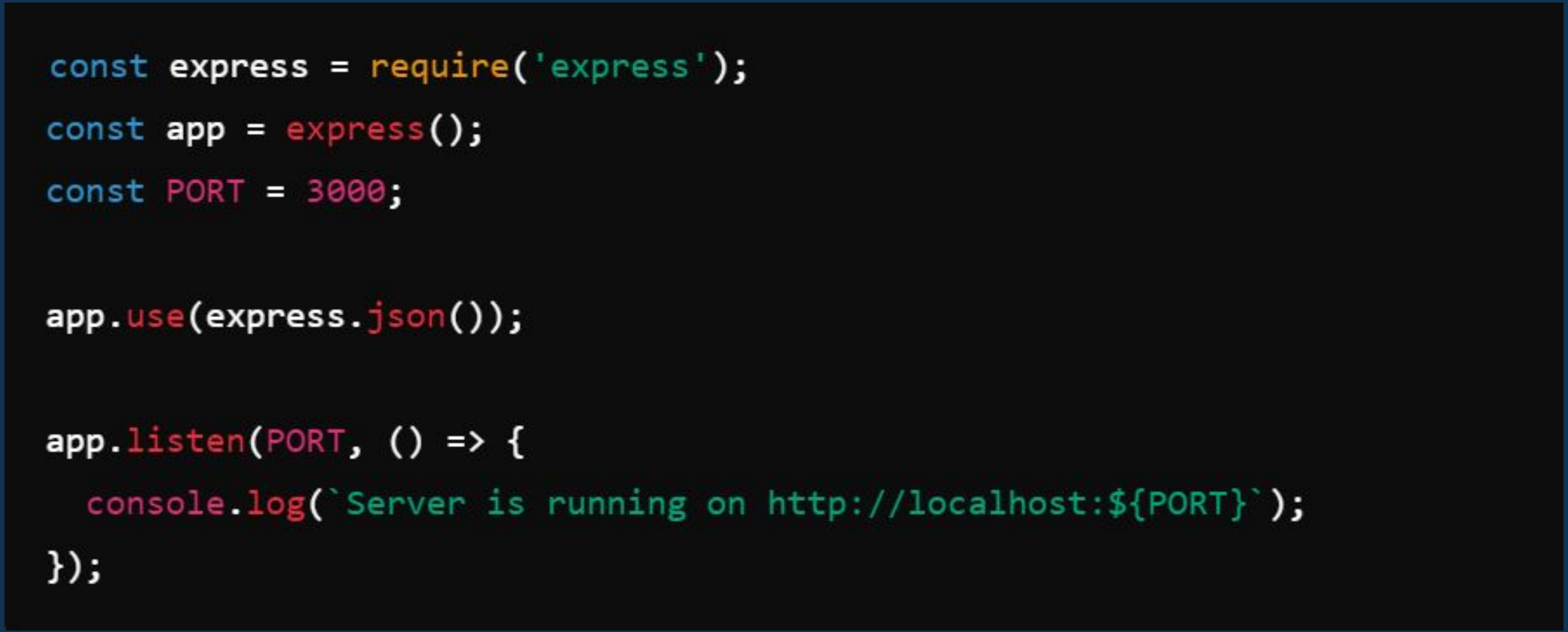
Creating a simple CRUD Application

- ❖ Setup (10 mins)
- ❖ Install Express



```
npm init -y
npm install express
```

SETUP OF BASIC SERVER



```
const express = require('express');
const app = express();
const PORT = 3000;

app.use(express.json());

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

Hyperiondev

Questions and Answers

HyperionDev

Thank You for attending!