CoGrammar

Welcome to this session:

React Routing and Context API

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.***
- Find all the lecture **content** in your **Lecture Backpack** on GitHub.
- If you are hearing impaired, kindly use your computer's function through Google chrome to enable captions.

Learning Outcomes

- Explain routing in React
- Explain state management
- Implement React Router DOM

Lecture Overview

- Implement folder structure
- Setup routing with React Router DOM
- Implement conditional rendering in Navigation
- Implement Context API

The Document Object Model (DOM)

- What is the Document Object Model (DOM)?
  - **The Document Object Model (DOM)** is a programming interface for web documents.
  - It represents the **structure** of HTML documents as a hierarchical **tree** of **objects**.
  - Each **node** in the tree corresponds to a **part** of the document, such as elements, attributes, or text content.
  - The DOM **provides** a structured representation of the document, allowing **scripts** to **dynamically** access, modify, and manipulate its content and structure.

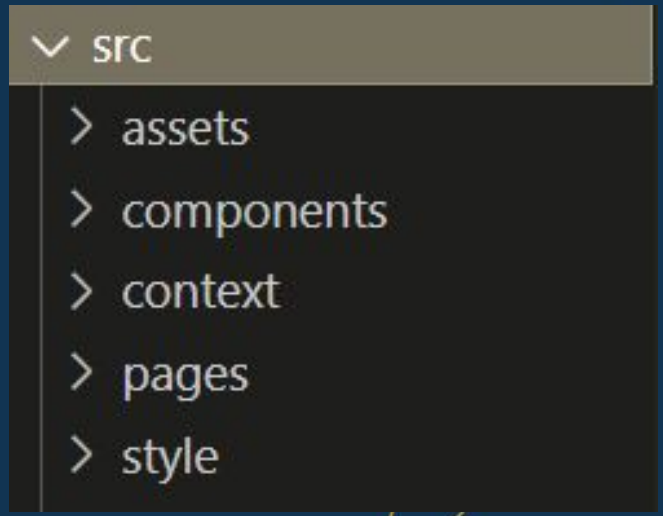
CoGrammar

Setup Folder Structure

- Grouping related operations makes navigating a codebase easier, this improves the following aspects of the application:
  - Readability
  - Maintainability
- Well structured projects also make it easier to work with other developers.

Setup Folder Structure

- **Assets:** Stores images and other small pieces of media
- **Components:** Stores all of our custom components
- **Context:** Stores our custom context
- **Pages:** Stores the pages that make up our application
- **Style:** Stores our style sheets.



CoGrammar

Setting up React Router DOM

- In order to render different pages based on the URL, we can use the **React Router DOM** to handle these operations.

```
$ npm i react-router-dom
```

CoGrammar

Setting up React Router DOM

- The **main.jsx** file will be used to store any global context for our application.
- We'll utilize the **BrowserRouter** context from **React Router DOM** to manage routing, making the defined routes accessible to all child components within the context.
- The **<App />** component will serve as the primary layout for our application.

Setting up React Router DOM

src > main.jsx

```jsx
import { StrictMode } from 'react'
import { createRoot } from 'react-dom/client'
import { BrowserRouter } from 'react-router-dom'
import App from './App.jsx'

createRoot(document.getElementById('root')).render(
  <StrictMode>
    <BrowserRouter>
      <App />
    </BrowserRouter>
  </StrictMode>,
)
```

CoGrammar

Setting up React Router DOM

- In the **App.jsx** file, we will define the structure of our application.
- We will us the **Routes** component to define the different paths in our application, and the elements that will be rendered when we get to that specific path.
- We will use the **Route** component to specify the individual routes and the elements that will be rendered at those paths.

Setting up React Router DOM

src > App.jsx > ...

```jsx
import { Routes } from 'react-router-dom'

function App() {

  return (
    <>
      {/* NAV BAR PLACEHOLDER */}

      <Routes>
        <Route path="/" element={<h1>Home Page</h1>} />
      </Routes>

      {/* FOOTER PLACEHOLDER */}
    </>
  )
}

export default App
```

CoGrammar

Setting up React Router DOM

- We want to render react components inside the **Route**
- Let's create 2 pages
  - **Login.jsx:** Allows a user to login
  - **Dashboard.jsx:** Shows information about the logged in user

Setting up React Router DOM

src > pages > Login.jsx > ...

```jsx
import { useState } from "react"

export default function Login() {
    const [userDetails, setUserDetails] = useState();

    const handleLogin = (e) => {
        e.preventDefault();
        const username = e.target.username.value;
        const password = e.target.password.value;

        if (! (username || password)) {
            alert("Please enter username and password");
            e.target.username.focus();
            return;
        }

        const user = validateUser(username);

        if (!user) {
            alert("Invalid username or password");
            e.target.username.focus();
            return;
        }

        setUserDetails(user);
    }

    return (
        <>
            <h1>Login Page</h1>

            <form>
                <label for="username">Username:</label>
                <input type="text" id="username" name="username" />
                <br />
                <label for="password">Password:</label>
                <input type="password" id="password" name="password" />
                <br />
                <button type="submit">Login</button>
            </form>
        </>
    )
}
```

src > pages > Dashboard.jsx > ...

```jsx
export default function Dashboard() {
    return (
        <>
            <h1>Dashboard Page</h1>

            <p>Welcome to the dashboard page!</p>
        </>
    )
}
```

CoGrammar

Setting up React Router DOM

- To navigate the application, we can create a **NavBar** component
- This component will use the **<Link>** to take use to the different routes that we will define

src > components > NavBar.jsx > ...

```
import { Link } from "react-router-dom";

export default function NavBar() {
    return (
        <nav>
            <ul>
                <li><Link to="/">Home</Link></li>
                <li><Link to="/dashboard">Dashboard</Link></li>
                <li><Link to="/login">Login</Link></li>
            </ul>
        </nav>
    )
}
```

CoGrammar

Setting up React Router DOM

- We are going to add the **NavBar** component to the **App.jsx** file.
- We also need to add the paths for the **Dashboard** and **Login** pages

CoGrammar

Setting up React Router DOM

src > App.jsx > ...

```
1  import { Route, Routes } from 'react-router-dom'
2  import NavBar from './components/NavBar'
3  import Dashboard from './pages/Dashboard'
4  import Login from './pages/Login'
5
6  function App() {
7
8    return (
9      <>
10       {/* New Code */}
11       <NavBar />
12       {/* New Code */}
13
14       <Routes>
15         <Route path="/" element={<h1>Home Page</h1>} />
16
17         {/* New Code */}
18         <Route path="/dashboard" element={<Dashboard />} />
19         <Route path="/login" element={<Login />} />
20         {/* New Code */}
21       </Routes>
22     </>
23   )
24 }
25
26 export default App
```

CoGrammar

Let's take a break

CoGrammar

Global State

- When a user logs in, we need to share their details throughout the application.
- To manage state globally, we can either make use of the **WebStorage API** to store the users' information temporarily.
- Another way would be to use the **Context API**.
- With the Context API, we can share data between all of the components that are children of the context provider.

CoGrammar

Global State

- We will create an **AuthContext.jsx** file and create out context object.

src > context > AuthContext.jsx > ...

```
1  import { createContext } from "react";
2
3  export const AuthContext = createContext();
```

CoGrammar

Global State

- In the **App.jsx** we will wrap all of the components that should share the context within the context provider.
- We will use the **useState** to get the details from the **<Login/>** component and link that to the provider
- We will also need to make sure that the **Login.jsx** component takes the **setUserDetails** function as an argument.
- We will also make use of the **useNavigate()** hook to take us to the dashboard page after successfully logging in.

Global State

src > App.jsx > ...

```jsx
import { Route, Routes } from 'react-router-dom'
import NavBar from './components/NavBar'
import Dashboard from './pages/Dashboard'
import Login from './pages/Login'
import { AuthContext } from './context/AuthContext'
import { useState } from 'react'

function App() {
  const [userDetails, setUserDetails] = useState(null);

  return (
    <>

      {/* NEW */}
      <AuthContext.Provider value={userDetails}>

        <NavBar />

        <Routes>
          <Route path="/" element={<h1>Home Page</h1>} />

          <Route path="/dashboard" element={<Dashboard />} />

          {/* Updated */}
          <Route path="/login" element={<Login setUserDetails={setUserDetails} />} />
        </Routes>
      </AuthContext.Provider>
    </>
  )
}
```

CoGrammar

Global State

src > pages > Login.jsx > ...

```
1
2  export default function Login({setUserDetails}) {
3
```

CoGrammar

Global State

src > pages > Login.jsx > ...

```jsx
import { useNavigate } from "react-router-dom";

export default function Login({setUserDetails}) {
    const navigate = useNavigate();

    const handleLogin = (e) => {
        e.preventDefault();
        const username = e.target.username.value;
        const password = e.target.password.value;

        if (! (username || password)) {
            alert("Please enter username and password")
            e.target.username.focus();
            return;
        }

        const user = validateUser(username);

        if (!user) {
            alert("Invalid username or password");
            e.target.username.focus();
            return;
        }

        setUserDetails(user);
        navigate("/dashboard");
    }

```

CoGrammar

Global State

- With the context setup, we can now access the user information from the **Dashboard.jsx** file.
- We will use the **useContext()** function to get the data that is stored in the **AuthContext** object.
- We will also use conditional rendering to only display the user details when the context contains a value.

CoGrammar

Global State

src > pages > Dashboard.jsx > ...

```jsx
import { useContext } from "react";
import { AuthContext } from "../context/AuthContext";

export default function Dashboard() {
    const userDetails = useContext(AuthContext);

    return (
        <>
            {
                userDetails ?
                    (
                        <div>
                            <h1>Welcome {userDetails.username}</h1>
                            <p>Your age is {userDetails.age}</p>
                            <p>Your favorite color is {userDetails.favoriteColor}</p>
                            <p>Your job title is {userDetails.jobTitle}</p>
                        </div>
                    ) :
                    (
                        <h1>Please login</h1>
                    )
            }
        </>
    )
}
```

CoGrammar

Global State

- Once the user has logged in, we can add their name to the **NavBar** component as well.

src > components > NavBar.jsx > ...

```
1   import { useContext } from "react";
2   import { Link } from "react-router-dom";
3   import { AuthContext } from "../context/AuthContext";
4
5   export default function NavBar() {
6       const userDetails = useContext(AuthContext);
7
8       return (
9           <nav>
10              <ul>
11                  <li><Link to="/">Home</Link></li>
12                  <li><Link to="/dashboard">Dashboard</Link></li>
13                  <li><Link to="/login">Login</Link></li>
14              </ul>
15
16              <p>
17                  {userDetails ? `Welcome ${userDetails.username}` : ""}
18              </p>
19          </nav>
20      )
21  }
```

CoGrammar

Questions and Answers

CoGrammar

Thank you for attending

CoGrammar

SKILLS FOR LIFE
SKILLS BOOTCAMPS

Department for Education