CoGrammar

Welcome to this session:
React – Routing

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.

Learning Outcomes

By the end of this lesson, learners should be able to:

- **Describe the concept of Routing** and its purpose in web applications.
- **Configure Routing using react-router-dom** and understand it's core components.
- **Implement basic routing** by setting up routes for different components as pages.
- **Handle dynamic routing** and pass state values using inbuilt hooks from react-router-dom

What is the purpose of state management in React? (Select all that apply)

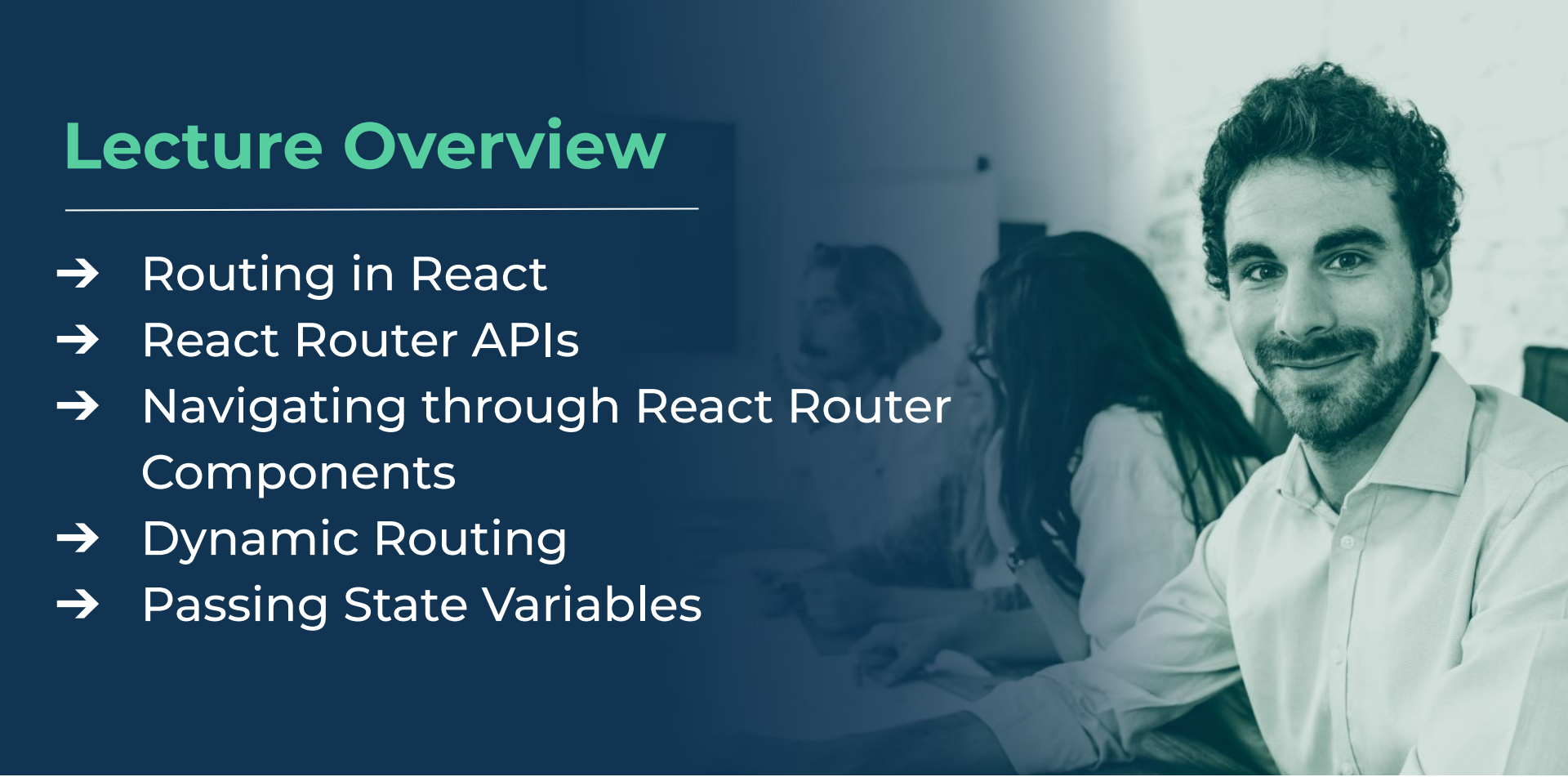
A. To control how the UI updates based on user input.
B. To make it easier to manage data flow between components.
C. To store and retrieve data from external databases.
D. To ensure consistent data is displayed across multiple components.

What do you think is meant by conditional rendering in React?
(Select all that apply)

A. Rendering different UI elements based on certain conditions.
B. Dynamically loading components based on user preferences.
C. Showing or hiding elements based on external API responses.
D. Rendering UI based on the component lifecycle.

Lecture Overview

- Routing in React
- React Router APIs
- Navigating through React Router Components
- Dynamic Routing
- Passing State Variables



Routing
Definition and Use Cases

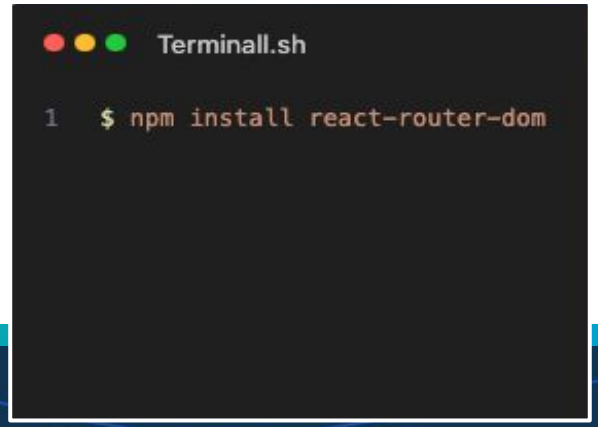
- Routing can be termed as the **conditional rendering** of components based on the **URL** in the browser.
- Routing allows users to **navigate between different pages or views** within a web application.
- Routing with plain HTML/CSS used to be **file based**, the anchor (<a>) were used to create hyperlinks that link to different web pages which were the different (.html) files in your project.

Routing in React

- In the context of React, **client side routing** is executed.
- This allows your app to **update the URL from a link click** without making another request for another document from the server, making your application **render immediately**.
- In simple terms, routing in React involves **dynamically updating the content** of the website without reloading the entire page.
- Routing in React is mostly implemented using **routing libraries** or frameworks. Two common libraries in use for a seamless routing experience are React Router DOM and Reach Router.

React Router DOM

- Achieves client side routing in your React application by using its inbuilt routing APIs.
  - To use React Router in your application, you need to install it first using npm or yarn:



Configuration

- After installing React Router, you need to **configure** your app to use it. This will be done in the root of you Javascript file (**main.jsx**).

```
//other React imports
import { createBrowserRouter, RouterProvider } from 'react-router-dom';

const paths = createBrowserRouter([
  {
    path: '/',
    element: <h1>Hello World</h1>
  }
])


const root = ReactDOM.createRoot(document.getElementById('root'));
root.render(
  <React.StrictMode>
    <RouterProvider router={paths} /> {/** replaced <App/> */}
  </React.StrictMode>
);
```

React Router APIs

- From the configuration example shown, we made two **important imports**:
  - **createBrowserRouter**: this configures Browser Router which enables client side Routing in our React application.
    - It is a function that takes in a list of available paths in our application, the paths will be defined by objects.
    - Currently, we've only created one path which is the home path using a '/' and it renders a <h1> text saying Hello World.

React Router APIs

- From the configuration example shown, we made two **important imports**:
  - **RouterProvider:** All path objects created by the createBrowserRouter API are passed to the provider component as a value of the router prop to render your app and enable routing.
- After this configuration, upon running your React server, you will have a text displaying Hello World on the home page.

Multiple Components

- Having multiple pages in our React app is one of the main achievements of routing.
- We do this by creating **other path objects** and **pointing** the path elements to their specific components.
- The **element property** of the path object will be replaced by a **React component from your project**.
- In this case, we have three components representing three pages and all are stored in a folder called pages for best practice purpose.

Multiple Components

```
//other React imports
import App from './pages/App';
import About from './pages/About'
import Contact from './pages/Contact'
import { createBrowserRouter, RouterProvider } from 'react-router-dom';

const paths = createBrowserRouter([
  {
    path: '/',
    element: <App/>
  },
  {
    path: '/about',
    element: <About/>
  },
  {
    path: '/contact',
    element: <Contact/>
  }
])


const root = ReactDOM.createRoot(document.getElementById('root'));
root.render(
  <React.StrictMode>
    <RouterProvider router={paths} />
  </React.StrictMode>
);
```

Navigating through React Router Components

- For hyperlinks, we are used to utilizing the **<a>** tag in **HTML**. Using **<a href=''>** causes a page refresh which can lead to losing an application's state.
- To achieve complete client side routing with **React Router**, we use its **<Link>** element to navigate from page to page. Instead of the {href='/path'} attribute in <a> tags, the link element provides a **{ to='/path'}** property to direct the link to the desired URL path.
- The **<Link>** element **does not cause** a page refresh hence the application's state cannot be lost.

Example

- The **{ Link }** element is imported from **'react-router-dom'**

NavBar.js

```
1   import { Link } from "react-router-dom"
2
3   const NavBar = () =>{
4       return (
5           <nav>
6               <Link to="/">Home</Link>
7               <Link to="/about">About</Link>
8               <Link to="/contact">Contact</Link>
9           </nav>
10      )
11  }
12
13  export default NavBar
```

App.js

```
1   import NavBar from "../components/NavBar"
2
3   function App () {
4       return (
5           <section>
6               <NavBar/>
7               <h1>Home</h1>
8           </section>
9       )
10  }
11
12  export default App
```

Dynamic Routing

- Dynamic routing is a way of rendering a new component by updating a particular segment in the URL called params.
- We achieve this by adding { **:id** } to the path, the colon section of the path will represent the dynamic segment. The suffix of the path will be replaced by respective path id or name.
- Note that you can name the id to anything as long as it rhymes with the intention. i.e { **:itemId** }, { **:userId** }.

Dynamic Routing Example

index.js

```
6   //other React imports
7   import App from './pages/App';
8   import About from './pages/About'
9   import Contact from './pages/Contact'
10  import User from './pages/User';
11  import { createBrowserRouter, RouterProvider } from 'react-router-dom';
12
13
14  const paths = createBrowserRouter([
15    {
16      path: '/',
17      element: <App/>
18    },
19    {
20      path: '/about',
21      element: <About/>
22    },
23    {
24      path: '/contact',
25      element: <Contact/>
26    },
27    {
28      path: '/user/:userId', //dynamic path, has the /:userId suffix
29      element: <User/>
30    }
31  ])
32  //other configurations
```

NavBar.js

```
1   import { Link } from "react-router-dom"
2
3   const NavBar = () =>{
4       return (
5           <nav>
6               <Link to="/">Home</Link>
7               <Link to="/about">About</Link>
8               <Link to="/contact">Contact</Link>
9               <Link to="/user/1">User 1</Link>
10              <Link to="/user/2">User 2</Link>
11              <Link to="/user/3">User 3</Link>
12          </nav>
13      )
14  }
15
16  export default NavBar
```

useParams() Hook

- The useParams hook returns an **object** of **key/value pairs** of the dynamic params from the current URL matched by the **dynamic path**.
- We created a **User.jsx** component that utilized the useParams to access the params of the { **/user/:userId** } path.

User.js

```
1   import { useParams } from "react-router-dom";
2
3   const User = ()=>{
4       const { userId } = useParams()
5       return (
6           <section>
7               User: { userId }
8           </section>
9       )
10  }
11
12  export default User;
```

CoGrammar

Passing State Variables

- State can be passed via the **<Link> element** in the same way we pass props to components. We use an extra prop called { **state** }.
- State can also be passed via a **useNavigate** hook provided by React Router which returns a function that lets you navigate programmatically.
- To access the state, we use a **useLocation** hook which returns a **location object** with the **state property** containing the state passed from the component.

Passing State Variables Example

Using <Link state={data}>

NavBar.js

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

const NavBar = () =>{

    const user1 = {
        id: 1,
        name: 'user1',
        role: 'Frontend Developer'
    }
    const user2 = {
        id: 2,
        name: 'user2',
        role: 'Backend Developer'
    }

    return (
        <nav>
            <Link to="/">Home</Link>
            <Link to="/about">About</Link>
            <Link to="/contact">Contact</Link>
            <Link to="/user/1" state={user1}>User 1</Link>
            <Link to="/user/2" state={user2}>User 2</Link>
            {/**other Link tags */}
```

Using useNavigate hook

NavBar.js

```
import { Link, useNavigate } from "react-router-dom"

const NavBar = () =>{
    const navigate = useNavigate()

    const user1 = {
        id: 1,
        name: 'Dan',
        role: 'Frontend Developer'
    }
    const user2 = {
        id: 2,
        name: 'Walobwa',
        role: 'Backend Developer'
    }

    const handleNavigatestate = (id, userData)=>{
        navigate(`/user/${id}`, { state: userData})

    }

    return (
        <nav>
            <Link to="/">Home</Link>
            <Link to="/about">About</Link>
            <Link to="/contact">Contact</Link>
            <button onClick={()=>handleNavigatestate(user1.id, user1)}>User 1</button>
            <button onClick={()=>handleNavigatestate(user2.id, user2)}>User 2</button>
            {/**other Link tags */}
```

useLocation hook

- The **useLocation** hook is used to access the state passed from its respective dynamic path. We access state from the location object returned by the useLocation hook.
- You need to import **useLocation** from React Router in order to use it. This gives access to **data passed** from both the <Link> element and the **useNavigate** hook.

```
import { useParams, useLocation } from "react-router-dom";

const User = ()=>{
    const { userId } = useParams()

    //accessing state using use location
    const location = useLocation()
    const userData = location.state
    return (
        <section>
            <p>User: { userId }</p>
            <p>Name: { userData.name}</p>
            <p>Role: { userData.role}</p>
        </section>
    )
}

export default User;
```

What is the purpose of the react-router-dom node module in React?

A. To manage and handle routing/navigation within a React application.
B. To handle state management and data flow between components.
C. To facilitate communication with external APIs.
D. To create and manage a local server for React applications.

What is the purpose of the useParams() hook in React?

A. To retrieve parameters from the URL in a React component.
B. To manage and update the state of a component.
C. To fetch data from an external API based on the URL.
D. To navigate programmatically between routes.

Let's take a break

CoGrammar

Questions and Answers

CoGrammar

Thank you for attending

CoGrammar

SKILLS FOR LIFE
SKILLS BOOTCAMPS

Department for Education