


Welcome to the CoGrammar Task Walkthrough: Task 20 - React - Routing

The session will start shortly...

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



CoGrammar Presentation

June 2024

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

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

Learning Objectives

- ❖ Understanding 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.
- ❖ Handling dynamic routing and pass state values using inbuilt hooks from react-router-dom

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

```
Terminall.sh

1  $ npm install react-router-dom
```

Configuration

- ❖ After installing React Router, you need to configure your app to use it. This will be done in the root of your Javascript file ([index.js](#)).

```
index.js

7  //other React imports
8  import { createBrowserRouter, RouterProvider } from 'react-router-dom';
9
10 const paths = createBrowserRouter([
11   {
12     path: '/',
13     element: <h1>Hello World</h1>
14   }
15 ])
16
17
18 const root = ReactDOM.createRoot(document.getElementById('root'));
19 root.render(
20   <React.StrictMode>
21     <RouterProvider router={paths} /> {/** replaced <App/> */}
22   </React.StrictMode>
23 );
```

React Router APIs

- ❖ From the configuration example shown, we made two important imports:
 1. **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

2. **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 Pages

- ❖ 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.

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}`.

Example Task

Example task: Simple Social Media

- You are tasked with creating a simple social media application where users can view a list of profiles and click on individual profiles to see more details.



Questions and answers



Thank you for attending



Department
for Education

CoGrammar

