CS 628: Full-Stack Development – Web App

City University of Seattle

School of Technology & Computing

Professor Sam Chung

## HOS06: React Router

Samantha Hipple

August 7, 2023

# PLEASE NOTE

Screenshots in this guide may differ from your environment (e.g., directory paths, version numbers, etc.). When choosing between a stable or most recent release, we advise you install the stable release rather than the best-testing version. Additionally, there may be subtle discrepancies along the steps, please use your best judgment to complete the tutorial. If you are unfamiliar with terminal, command line, and bash scripts, we recommend watching [this video](https://youtu.be/Dp7uw9c6QH8) prior to moving forward with this guide. Not all steps are fully explained. Lastly, we advise that you avoid copy-pasting code directly from the guide or GitHub repositories. Instead, type out the code yourself to improve familiarity.

More information on this guide can be found under the related module in [this repository](https://github.com/samchung0117/cs628-examples). Please save a screenshot of the app at the end of each section and save it in the current module folder with the relevant section number.

# SECTION CONTENTS

1. Accessing GitHub Codespaces
2. BrowserRouter
3. MemoryRouter
4. HashRouter
5. Nested routing
6. Querying parameters
7. Using NavLink
8. Redirect using Navigate
9. History management

# SECTION 1. ACCESSING GITHUB CODESPACES

GitHub Codespaces is an online cloud-based development environment that allows users to easily write, run and debug code. Codespaces is fully integrated with your GitHub repository and provides a seamless experience for developers. In order to access Codespaces, users only need a GitHub account and an active internet connection.

After downloading the current HOS assignment, in the top-right corner of the repo, click on the **<> Code** drop-down menu and select **Create codespace on main** as shown in the following image. The free and pro GitHub subscriptions include free use of GitHub Codespaces *up to a fixed amount of usage each month*. In order to avoid unexpected charges, please review the [billing information](https://docs.github.com/en/billing/managing-billing-for-github-codespaces/about-billing-for-github-codespaces).

**SECTION 2.** [**BROWSERROUTER**](https://reactrouter.com/en/main/router-components/browser-router#browserrouter)

React Router is a widely used library in the React ecosystem that facilitates efficient client-side routing for single-page applications (SPAs). React Router enables developers to create navigable user interfaces within a single HTML page, eliminating the need for full-page reloads. By defining routes and matching them to specific components, React Router dynamically renders the appropriate content based on Uniform Resource Locator (URL) changes, resulting in a seamless and fluid user experience. This library supports advanced features like nested routing, query parameter handling, and programmatic navigation, making it a powerful tool for managing client-side routing in React applications.

A **<BrowserRouter>** stores the current location in the browser's address bar using clean URLs and navigates using the browser's built-in history stack. The browser router is typically used as the top-level router component in the application and serves as the entry point for defining routes and mapping them to specific components.

1. Use the following terminal commands to start the project:

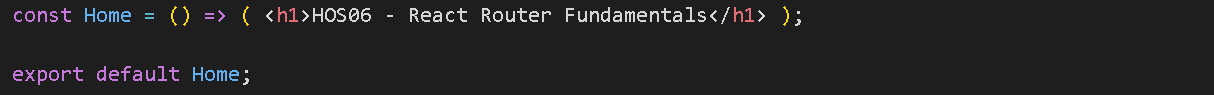
**>>>npx create-react-app ./myapp**

**>>>cd myapp**

**>>>npm install react-router-dom**

**>>>npm start**

1. Replace the code in **App.css** file with the [content found here](https://github.com/samchung0117/cs628-examples/blob/main/Module%2006/client-src-examples/section2/style.css).
2. Create a file named **home.js** and add the following code:



1. Create a file named **about.js** and add the following code:

A computer screen shot of text

Description automatically generated

1. Create a file named **details.js** and add the following code:

A computer screen with white text

Description automatically generated

1. Updated **App.js** to match the following:

A screen shot of a computer screen

Description automatically generated

1. Refresh the browser and test the changes by viewing the different routes.

A screenshot of a computer

Description automatically generated

A screenshot of a computer

Description automatically generated

A screenshot of a computer

Description automatically generated

# SECTION 3. [MEMORYROUTER](https://reactrouter.com/en/main/router-components/memory-router#memoryrouter)

The **<MemoryRouter>** stores URL changes in memory instead of the user’s browser. This keeps a history of URLs internally, bypassing the address bar and disabling the back & forward buttons on the browser. This is most beneficial for testing and non-browser environments, such as React Native, where browser-specific navigation features are not needed. Additionally, the memory router allows developers to simulate URL changes and test routing behavior without affecting the actual browser history.

1. Update **App.js** by replacing the **<BrowserRouter>** with a **<MemoryRouter>**:

A screen shot of a computer program

Description automatically generated

1. Refresh the browser and test the changes by viewing the different routes.

A screenshot of a computer

Description automatically generated

A screenshot of a computer

Description automatically generated

A screenshot of a computer

Description automatically generated

Notice that the URL does not update with the routes when using the memory router.

# SECTION 4. [HASHROUTER](https://reactrouter.com/en/main/router-components/hash-router#hashrouter)

The **<HashRouter>** uses URL hashes to enable client-side navigation in SPAs. This is particularly useful in scenarios where server-side handling of URL changes is unavailable as it enables the back and forward buttons on the browser while the server disregards the hash portion of the URL and continues serving the **index.html** file for every request, leaving that handling of the hash values solely to the client-side application.

1. Update **App.js** by replacing the **<MemoryRouter>** with a **<HashRouter>**:

A screen shot of a computer program

Description automatically generated

A screen shot of a computer program

Description automatically generated

1. Refresh the browser and test the changes by viewing the different routes.

A screenshot of a computer

Description automatically generated

A screenshot of a computer

Description automatically generated

A screenshot of a computer

Description automatically generated

# SECTION 5. [NESTED ROUTING](https://reactrouter.com/en/main/components/outlet#outlet)

Nested routing in React Router defines routes within other routes, creating a hierarchical structure of components corresponding to specific URLs. With nested routing, you can render nested components based on the current URL path, allowing for more organized and modular code. Parent components can act as layouts, rendering common elements, while child components are swapped in and out based on the nested URL segments.

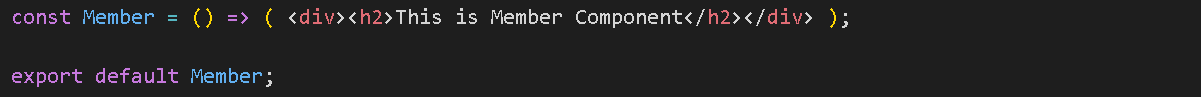
1. Create a new file called **team.js** and add the following code:

A computer screen shot of a black screen

Description automatically generated

The **<Outlet />** component is provided by the **react-router-dom** library and is used to render child routes within a parent route.

1. Create a new file called **member.js** and add the following code:



1. Update **App.js** to match the following:

A screen shot of a computer program

Description automatically generated

1. Refresh the browser and test the changes by clicking on the team and member links.

A screenshot of a computer

Description automatically generated

A screenshot of a computer

Description automatically generated

Here, **Member** is the child component rendered under the **Team** component using nested routing.

# SECTION 6. [QUERYING PARAMETERS](https://reactrouter.com/en/main/hooks/use-params#useparams)

In React Router, **useParams** is a hook provided by the **react-router-dom** library that allows access to and extraction of dynamic parameters from the URL. The hook is specifically used to retrieve values from route parameters defined in the route path. When you define a route with a parameter placeholder in the path, such as **“/member/:id”**, the **useParams** hook allows you to extract the value of **:id** from the URL. Let us see an example.

1. Update **team.js** to match the following:

A computer screen shot of a code

Description automatically generated

1. Update **member.js** to match the following:

A screen shot of a computer

Description automatically generated

A screen shot of a computer

Description automatically generated

1. Update the member **Route** in the **App** component to match the following:



1. Refresh the browser and test the changes by clicking on the team and member links.

A screenshot of a computer

Description automatically generated

A screenshot of a computer

Description automatically generated

# SECTION 7. [USING NAVLINK](https://reactrouter.com/en/main/components/nav-link#navlink)

In React Router, **NavLink** is a component provided by the **react-router-dom** library that is used to create navigation links within your application. It is like the regular **Link** component but comes with additional features specific to navigation. The **NavLink** component allows you to define navigation links that can have an **active** state based on the current URL. When the user clicks on a **NavLink**, React Router automatically applies an **active** class to the link's rendered HTML element if the current URL matches the link's **to** prop. Let us modify our **App** component to use **NavLink** components instead of **Link**.

1. Replace the **Link** components with **NavLink** in **App.js** as follows:

A screen shot of a computer code

Description automatically generated

1. Update **App.css** to include the following style:

A black rectangle with white text

Description automatically generated

1. Refresh the browser and test the changes by viewing the different routes.

A screenshot of a computer

Description automatically generated

A screenshot of a computer

Description automatically generated

Notice that the selected tab is highlighted due to the styling we added in **App.css** without requiring you to provide a value to the **activeClassName** prop within the **NavLink** component.

**SECTION 8.** [**REDIRECT USING NAVIGATE**](https://reactrouter.com/en/main/components/navigate#navigate)

In React Router, the **<Navigate>** component is used to redirect users to a different route within the application programmatically. This enables the developer to navigate users to specific URLs without requiring any user interaction (i.e., click a link or button). Let us implement **Navigate** under the **Team** component.

1. Update **team.js** to match the following:

A screen shot of a computer program

Description automatically generated

A screen shot of a computer program

Description automatically generated

1. Refresh the browser and test the changes by pressing the new button under **Team**.

A screenshot of a computer

Description automatically generated

# SECTION 9. [HISTORY MANAGEMENT](https://reactrouter.com/en/main/hooks/use-navigate#usenavigate)

The **useNavigate()** hook was introduced in React Router v6 as an alternative to the **useHistory()** hook found in previous versions. With **useHistory()**, developers would access the React Router history object and use **push** or **replace** methods for navigation. The hook enabled moving to specific URLs as well as navigating back and forth between pages. Let us update the **Team** and **Details** components with go back and redirect functions using the **useNavigate()** hook.

1. Update **team.js** to match the following:
2. Update **details.js** to match the following:
3. Refresh the browser and test the changes by playing with the navigation.