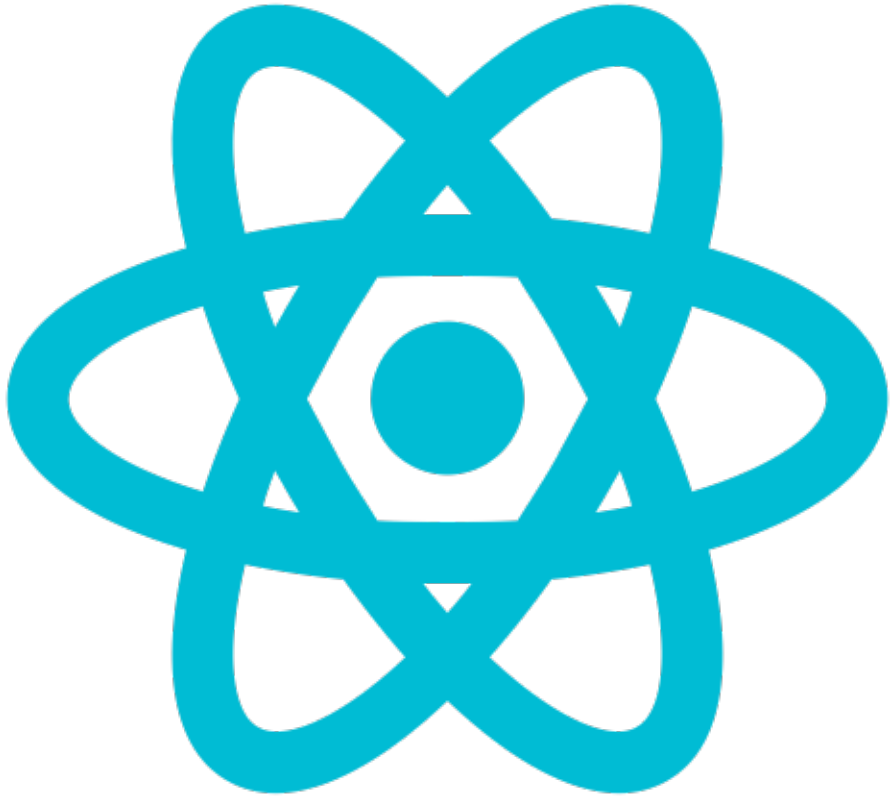
REACT ROUTER



React Router

- React Router is a collection of **navigational components** that compose declaratively with your application. Whether you want to have **bookmarkable URLs** for your web app or a composable way to navigate in **React Native.**

Browser Router

- **<BrowserRouter>**
- A <Router> that uses the HTML5 history API to keep your UI in sync with the URL.
- Wrap your component with this tag

```
<BrowserRouter
  basename={optionalString}
  forceRefresh={optionalBool}
  getUserConfirmation={optionalFunc}
  keyLength={optionalNumber}
>
  <App />
</BrowserRouter>
```

Route

- **<Route>**
- The Route component is perhaps the most important component in React Router. Its most basic responsibility is to render some UI when its path matches the current URL.
- **Route render methods**
- The recommended method of rendering something with a <Route> is to use children elements, as shown above. There are, however, a few other methods you can use to render something with a <Route>. These are provided mostly for supporting apps that were built with earlier versions of the router before hooks were introduced.
- <Route component>
- <Route render>
- <Route children> function

Route - component

- When you use component (instead of render or children, below) the router uses React.createElement to create a new React element from the given component.
- That means if you provide an inline function to the component prop, you would create a new component every render. This results in the existing component unmounting and the new component mounting instead of just updating the existing component.
- When using an inline function for inline rendering, use the render or the children prop (below).
  render: func **children: func**

Route Props

- **Route props**
- All three render methods will be passed the same three route props
- match
- location
- history

match

- A match object contains information about how a <Route path> matched the URL. match objects contain the following properties:

  - params - (object) Key/value pairs parsed from the URL corresponding to the dynamic segments of the path
  - isExact - (boolean) true if the entire URL was matched (no trailing characters)
  - path - (string) The path pattern used to match. Useful for building nested <Route>s
  - url - (string) The matched portion of the URL. Useful for building nested <Link>s

location

- Locations represent where the app is now, where you want it to go, or even where it was. It looks like this:

```
<li><Link to={{
    pathname: '/new-post',
    hash: '#submit',
    search: '?quick-submit=true'
}}>New Post</Link></li>
```

history

- The term “history” and "history object" in this documentation refers to the history package, which is one of only 2 major dependencies of React Router (besides React itself), and which provides several different implementations for managing session history in JavaScript in various environments.
- The following terms are also used:
  - “browser history” - A DOM-specific implementation, useful in web browsers that support the HTML5 history API
  - “hash history” - A DOM-specific implementation for legacy web browsers
  - “memory history” - An in-memory history implementation, useful in testing and non-DOM environments like React Native

https://reactrouter.com/web/api/history

Link

- **<Link>**
- Provides declarative, accessible navigation around your application.
  - **to: string**
    - A string representation of the Link location, created by concatenating the location's pathname, search, and hash properties.
  - **to: object**
    - An object that can have any of the following properties:
    - pathname: A string representing the path to link to.
    - search: A string representation of query parameters.
    - hash: A hash to put in the URL, e.g. #a-hash.
    - state: State to persist to the location.

Redirect

- **<Redirect>**
- Rendering a <Redirect> will navigate to a new location. The new location will override the current location in the history stack, like server-side redirects (HTTP 3xx) do.

Switch

- **<Switch>**
- Renders the first child <Route> or <Redirect> that matches the location.
- **How is this different than just using a bunch of <Route>s?**
- <Switch> is unique in that it renders a route *exclusively*. In contrast, every <Route> that matches the location renders *inclusively*. Consider these routes: