REACT JS

React Router

Sreekanth M. E.

Freelance Trainer & Consultant

http://www.SreekanthME.com

In an SPA, routing is extremely important to allow users to move from view to view without hitting the server.

Current version of react router is v4.

In v4, everything is now “just components”.

The same intuition required for React components and how to compose them, will hold good for the routing code also.

React dev tools inside Chrome browser can be used to see how all React components (including route components) of a React SPA come together.

React-router uses “Dynamic Routing”, which is quite different from “Static Routing”.

React Router followed static routing up until v4.

Static Routing: Routes are declared as part of app's initialization before any rendering takes place (i.e. before the app listens).

Creating Static routing is basically like updating a configuration file with all routes in one place.

Dynamic Routing: Routing takes place as the app is rendering. Not in a configuration or convention outside of the running app.

Route is just a component and it can be used with other components to compose the SPA.

React Router has been broken into three packages:

- react-router
- react-router-dom
- react-router-native

react-router-dom and react-router-native provide environment specific (browser and react-native) components.

No need to install react-router separately. All its features come with react-router-dom or react-router-native.

npm install --save react-router-dom

https://unpkg.com/react-router/umd/react-router.min.js

For browser based projects, react-router-dom has <BrowserRouter> and <HashRouter> components.

HashRouter: URL for about page would look something like : http://example.com/#/about . It is suitable for static pages only.

BrowserRouter: URL for about page would be an ordinary URL like : http://example.com/about . It is suitable for dynamic pages.

<BrowserRouter>

A <Router> that uses the HTML5 history API (pushState, replaceState and the popstate event) to keep your UI in sync with the URL.

<HashRouter>

A <Router> that uses the hash portion of the URL (i.e. window.location.hash) to keep your UI in sync with the URL.

Entire React app must be inside <Router>

<Router> expects to receive a single child element. To work within this limitation, it is useful to create an <App> component that renders the rest of your application OR wrap child elements in a single DIV tag.

HTML anchor elements reload page when clicked.

<Link> component prevents that from happening. When clicked, only the rendered content will change without reloading the page.

to: string

The pathname or location to link to.

<NavLink> is a special version of the <Link> that will add styling attributes to the rendered element when it matches the current URL.

activeClassName: string

CSS class to be used when link is active.

exact: bool

Apply activeClassName style if the location is matched exactly.

<Route> renders some UI when a location matches the route's path.

path: string

Any valid URL path including path params.

Example: <Route path="/users/:id" component={User}/>

Routes without a path always match.

exact: bool

When true, will only match if the path matches the location.pathname exactly.

Some important <Route> render content:

- component: Name of the React component to be rendered.
- render: A function that returns JSX to be rendered. It is useful for inline rendering and passing extra props to components.

The Route component (and the route render method) will be passed the same three route props namely:

- match
- location
- history

They can be accessed as props of the component (like so: this.props.match.<key>, this.props.location.<key>, this.props.history.<key>)

Each router creates a history object, which it uses to keep track of the current location and re-render the website whenever that changes.

The other components provided by React Router rely on having that history object available to them.

HISTORY object keys:

length - (number) The # of entries in the history stack

action - (string) Current action (PUSH, REPLACE, or POP)

location - (object) Current location. May have the following properties:

- pathname - (string) The path of the URL
- search - (string) The URL query string
- hash - (string) The URL hash fragment

Location object represent where the app is now, where it has to go, or even where it was.

pathname - (string) The path of the URL

search - (string) The URL query string

hash - (string) The URL hash fragment

A match object contains information about how a <Route path> matched the browser URL.

match objects contains the following properties:

params - (object) Key/value pairs parsed from the URL corresponding to the dynamic segments of the path. or path parameters

isExact - (boolean) true if the entire URL was matched (no trailing characters)

path - (string) The path pattern used to match.

url - (string) The matched portion of the URL.

<Switch> renders the first child <Route> that matches the location.

<Switch> renders a route exclusively. In contrast, every <Route> that matches the location renders inclusively.

Example:

```
<Route path="/about" component={About}/>
<Route path="/:user" component={User}/>
<Route component={NoMatch}/>
```

Without switch, If the URL is /about, then <About>, <User>, and <NoMatch> will all render because they all match the path.

END OF CHAPTER

APPENDIX