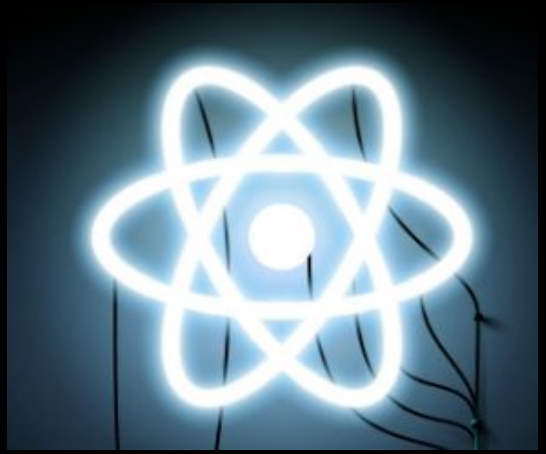
Lecture 4.2

React Router



Topics

- Defining Basic Routes
- Defining Multiple Routes
- Route Parameters
- Navigating with History

Routing


3,963 MI
CORIN
245 MI
182 MI
REMOT
2,035 MI
GUADALAJARA
KANGAROOS
MECCA
2,443 MI
TIMES SQUARE
2,540 MI
ASHKELON
7 BKS.
WATERFRONT
GORGE
48 MI
2,530 MI
WALDEN POND
A LONG WAY

React Router

- React Router is the most popular and commonly used library for routing in React applications.
- As your React application grows, it's ideal to choose a good router to help manage the transitions between views, redirects, get URL parameters etc.
- React Router 4, allows you to define your routes declaratively.
- React Router 4 API is basically just components, making it easy to use if you already have React components

Router Setup

- React Router is composed of these packages: react-router, react-router-dom, and react-router-native.

- react-router: comprises of the core routing components.
- react-router-dom: comprises of the routing API required for browsers.
- react-router-native: comprises of routing API for mobile applications.

- Use NPM to install react-router-dom

```
npm install --save react-router-dom
```

```
"dependencies": {
  "react": "^16.8.3",
  "react-dom": "^16.8.3",
  "react-router-dom": "^4.3.1",
  "react-scripts": "2.1.5"
},
```

Basic Routing

- There are two types of Router components that you can use in your React web application. The BrowserRouter and HashRouter.

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

- HashRouter gives you a URL without the #, while the latter gives you a URL with the #.
    - Note: If you are building a web application that supports legacy browsers, it's recommended that you use the HashRouter.

```
import { BrowserRouter, Route } from "react-router-dom";
```

Setting Up Basic Routes

```
import {
  BrowserRouter,
  Route } from "react-router-dom";
```

```
class App extends Component {
  render() {
    return (
      <BrowserRouter>
        <Route>
        </Route>
      </BrowserRouter>
    );
  }
}
```

- BrowserRouter and Route are imported from the react-router-dom
- The <App/> component is then wrapped with BrowserRouter.
- The BrowserRouter component is the first step to routing. It serves as the container for every other route component.
- The Router component can have only one child element or component

Defining Routes

```
import {
  BrowserRouter,
  Route,
  Switch,
  Link } from "react-router-dom";
```

```
<BrowserRouter>
  <div>
    <ul>
      <li>
        <Link to="/">Home</Link>
      </li>
      <li>
        <Link to="/newroute">New Route</Link>
      </li>
    </ul>
    <Switch>
      <Route path="/" component={Home} exact />
      <Route path="/newroute" component={NewRoute} />
    </Switch>
  </div>
</BrowserRouter>
```

<Link> Provides declarative, accessible navigation around your application

- to: string representation of the location to link
- to: object with properties { pathname, search, hash, state }

<Switch> renders the first child <Router> or <Redirect> that matches the location

- It renders a route exclusively.

Defining Multiple Routes

```
<BrowserRouter>
  <div>
    <ul>
      <li><Link to="/">Home</Link></li>
      <li><Link to="/contact">Contact</Link></li>
      <li><Link to="/about">About</Link></li>
    </ul>
    <Switch>
      <Route path="/" component={Home} exact />
      <Route path="/about" component={About} />
      <Route path="/contact" component={Contact}/>
      <Route component={Error} />
    </Switch>
  </div>
</BrowserRouter>
```

- Exact when true, will only match if the path matches the location.pathname exactly.
- If no route match is found the default route will be display the Error component.
- <NavLink> can be used in place of <Link> when using a style on the active link

```
<NavLink to="/about" activeStyle={{ color: "green" }}>
```

Video : Routing

https://youtu.be/W6m2qUCYLuk

Router Parameters

```
<Route
  path="/student/:studentname/:studentno?"
  component={Student}
/>
```

The following routes will be accepted:

- /student/Randy Savage
- /student/Lex Luger/96000015

- React Router allows information to be read from the URL as parameters.
- Parameterized Route, any segment that starts with a colon will be treated as a parameter
- Optional parameters are defined by placing a question mark at the end of a parameter

Using Route Parameters

```
const User = ({ match }) => {

  const { username } = match.params;
  return (
    <div>
      <p>Student</p>
      <div>
        <div>{`Name: "${username}"!`}</div>
      </div>
    </div>
  );
};
```

- Route parameters are passed to the component as props
- Optional parameters are passed alongside the mandatory ones.
- Optional parameters will be undefined, if they're not in the URL
- The URL is passed in as match.url
- The Route path is passed in as match.path
- Parameters are passed as match.params

Nested Routes

There is has no “nesting” API in React Router v4. Route is just a component, just like div. So to nest a Route, you just create another component with a Route in it.

```
const App = () => (
  <BrowserRouter>
    {/* here's a div */}
    <div>
      {/* here's a Route */}
      <Route path="/tacos" component={Tacos} />
    </div>
  </BrowserRouter>
);
```

```
// when the url matches `/tacos` this component renders
const Tacos = ({ match }) => (
  // here's a nested div
  <div>
    {/* here's a nested Route,
        match.url helps us make a relative path */}
    <Route path={match.url + "/carnitas"} component={Carnitas} />
  </div>
);
```

History Package

```
import {
    createBrowserHistory,
    createHashHistory,
    createMemoryHistory
  } from 'history'
```

- History package provides core functionality for React Router.
- It enables projects to easily add location based navigation on the client-side
- There are three types of history: browser, hash and memory.
- We create history objects, so we don't interact with history directly

Navigating with History

```
history.push('/')

history.goForward()

history.goBack()

history.go(4)
```

- History object has a location property and some navigation methods that allow you to change the current location.
- The push method allows you to go to a new location.
- By default, when you click on a <Link> from React Router, it will use history.push to navigate
- There are three related methods: goBack, goForward and go

Conditional Routes + Simple Authorization

The routes will be added and handled by the router only if the <Route> component is rendered by passing the hasRole evaluation

```
import React from 'react';
import {
    BrowserRouter,
    Switch,
    Route } from 'react-router-dom';

const App = ({ user }) => (
  <BrowserRouter>
    <Switch>
      {hasRole(user, ['user']) && <Route path='/user' component={User} />}
      {hasRole(user, ['admin']) && <Route path='/admin' component={Admin} />}
      <Route exact path='/' component={Home} />
    </Switch>
  </BrowserRouter>
);
```