**Introduction to React Router 5**

## React Router v5 vs React Router v3

Before v5 there was **React Router v3**, **React router v5** is a complete re-write so what's the difference between these two React routers? here is a summary list of most differences:

* With **React router v5**, routing is not centralized anymore instead it becomes a part of the rest of the app layout and UI.
* Browser specific routing components live in **react-router-dom** instead of **react-router** so imports need to be changed to be from **react-router-dom** package.
* Introducing new components such as **BrowserRouter** and **HashRouter** for specific use cases (see below).
* No more use of **{props.children}** for nesting components in v5 React Router.
* React Router v3 routing rules were exclusive meaning only one route will be matched at one time. For v5, routing rules are inclusive meaning multiple routes can be matched and then rendered.

**React-router-dom** is the version of React Router v5 designed for web applications, React Router v5 was divided into three packages:

* **react-router**: common core components between dom and native versions.
* **react-router-dom**: the dom version designed for browsers or web apps.
* **react-router-native**: the native version designed for react-native mobile apps.

## react-router vs react-router-dom vs react-router-native

react-router hosts the core components for routing for React applications, react-router-dom provides browser specific components for routing web apps and react-router-native provides specific components for react-native or mobile apps created with React Native. So you should either install **react-router-dom** or **react-router-native** as both export their corresponding environments components plus what **react-router** exports.

## Installing React Router 5

Since we are building a web application not a native mobile app we need to install **react-router-dom** package, so inside your React project run the following command using your terminal (Linux or MAC) or command prompt (Windows):

npm install --save react-router-dom

## Understanding and Using React Router 5 Routers (BrowserRouter vs HashRouter)

* BrowserRouter: This is a sub-class or a concrete implementation of Router interface that makes use of HTML5 history API to sync your UI with the current browser's url or actually the url's path i.e window.location.
* HashRouter: Just like the previous router but only uses the hash part of the URL i.e window.location.hash.
* MemoryRouter
* NativeRouter: Used for routing inside react-native mobile apps.
* StaticRouter: Used for static routing just like React Router v3.

### BrowserRouter vs HashRouter

There are many types of Router components, among them **< BrowserRouter >** and **< HashRouter >** for client side React apps. If you are using a dynamic server that can handle dynamic URLs then you need to use the **BrowserRouter** component but if you are using a server that only serves static files then a **HashRouter** component is what to be used in this case.

## Understanding and Using React Router 5 Routes

The **< Route >** component is one of the most useful components of React Router v5 and the idea behind it is simple, wherever you want to render something when only there is a match with the location's path you can use a Route component.

The **Route** component takes many properties such as:

* path property: of type string, it holds the name of path to be matched.
* component property: it holds the name of the component to be rendered if there is a match.
* exact property: this property tells Route to exactly match the path (see inclusive and exclusive routing)
* strict property: this property tells Route to match only a path that has a trailing slash.

There are two other properties which can replace the component property to tell the Route component what it needs to render when there is a match:

* render property:a function that return a React element. More useful for inline rendering or for wraping rendered component.
* children: also a function which renders a React element. Except that this one will always render even if there is no path match.

Most of the times, you will use component propery but there are also many situations when you'll have to use either render or children properties instead. Thre three methods will be passed these three props:

* match
* location
* history

For example:

Using component:

<Route exact path="/" component={HomePage} />

Will render the HomePage component when browser's location path matches exactly /.

Using render:

For inline rendering:

<Route path="/home" render={() => <div>Home</div>}/>

For wrapping:

const FadingRoute = ({ component: Component, ...rest }) => (

<Route {...rest} render={props => (

<FadeIn>

<Component {...props}/>

</FadeIn>

)}/>

)

<FadingRoute path="/cool" component={Something}/>

Using children:

<ul>

<ListItemLink to="/somewhere"/>

<ListItemLink to="/somewhere-else"/>

</ul>

const ListItemLink = ({ to, ...rest }) => (

<Route path={to} children={({ match }) => (

<li className={match ? 'active' : ''}>

<Link to={to} {...rest}/>

</li>

)}/>

)

For more information about how React Router v5 matchs paths see [path-to-regexp](https://github.com/pillarjs/path-to-regexp) the module used for matching paths.

## Understanding React Router 5 URL/Path/Route Parameters

Usually there are variable parts of the pathname used to pass information between diffrent routes of an application so how do we capture these variables and pass them to components? We can just append the name to be used for the variable plus a colon **:** to the end of the route's path, for example:

<Route path="/:param1" component={Home}/>

const Home = ({ match }) => (

<div>

<h1> Parameter 1 : {match.params.param1}</h1>

</div>

)

When there is a path match an object which has the following properties will be created and passed to the component:

* url: the matched part of the URL.
* path: simply the path.
* isExact: equals **True** if path equals exacly the current location's path-name.
* params: an object containing URL parameters.

## Understanding and Using Links in React Router 5

Links are React Router v5 components designed as a replacment of anchor links to create navigation elements which enable users to navigate between differenet pages of React apps. Unlike anchors ,which reloads the whole page, Links only reload the portion(s) of the UI that match(s) the browser's location path.

A **Link** component takes a **to** property which tells React Router the destination to navigate to. For example:

import { Link } from 'react-router-dom'

const Nav = () => (

<Link to='/'>Home</Link>

)

When clicked will take us to location with path: /

the **to** prop can either take a string or a location (pathname, hash, search, and [state](https://www.techiediaries.com/react-setstate)) object, for example:

<Link to={ {

pathname: '/me',

search: '?sort=asc',

hash: '#hash',

state: { fromHome: true }

} } />

**Link** can take also another property: **replace** if **True**, when clicked the link entry will be replaced in the history.

## React Router 5 Link vs. NavLink

**NavLink** is a subclass of **Link** which adds styling information to the rendered element(s), for example:

import { NavLink } from 'react-router-dom'

<NavLink

to="/me"

activeStyle=

activeClassName="selected">My Profile</NavLink>

## Writing our First Example with React Router DOM

Now let's write an example React app which shows you how to use **BrowserRouter** to implement routing.

First we import the necessary routing components such as **Route** and **BrowserRouter**

import { BrowserRouter } from 'react-router-dom'

import { Route } from 'react-router-dom'

Next we create the base layout component, besides common HTML tags we also use React Router v5 components **Link** and **Route**:

const BaseLayout = () => (

<div className="base">

<header>

<p>React Router v5 Browser Example</p>

<nav>

<ul>

<li><Link to='/'>Home</Link></li>

<li><Link to='/about'>About</Link></li>

<li><Link to='/me'>Profile</Link></li>

<li><Link to='/login'>Login</Link></li>

<li><Link to='/register'>Register</Link></li>

<li><Link to='/contact'>Contact</Link></li>

</ul>

</nav>

</header>

<div className="container">

<Route path="/" exact component={HomePage} />

<Route path="/about" component={AboutPage} />

<Route path="/contact" component={ContactPage} />

<Route path="/login" component={LoginPage} />

<Route path="/register" component="{RegisterPage}" />

<Route path="/me" component={ProfilePage} />

</div>

<footer>

React Router v5 Browser Example (c) 2017

</footer>

</div>

)

Next we create our pages:

const HomePage = () => <div>This is a Home Page</div>

const LoginPage = () => <div>This is a Login Page</div>

const RegisterPage = () => <div>This is a Register Page</div>

const ProfilePage = () => <div>This is the Profile Page</div>

const AboutPage = () => <div>This is an About Page</div>

const ContactPage = () => <div>This is a Contact Page</div>

And finally we create the App component which **BrowserRouter** component to hold our base layout component then render the app.

const App = () => (

<BrowserRouter>

<BaseLayout />

</BrowserRouter>

)

render(<App />, document.getElementById('root'))

As you can see, it's very easy to use the React Router v5 components to create apps with routing.

## Understanding React Router 5 Inclusive Routing

In our example app we used the prop **exact** in the Route for component HomePage

<Route path="/" exact component={HomePage} />

That's because React Router v5 uses inclusive routing instead of exclusive routing used by React Router v3 so without exact property the home component will be rendered with all other components, for example when the user visits /login path both / and /login paths will be matched and their corresponding components LoginPage and HomePage will be rendered. But that's not the behavior we are looking for, that's why we need to add the exact prop which tells the Route component to match exactly the / path.

Now let's see how we can use inclusive routing in our advantage, lets suppose we have a sub-menu component that needs to be available only when we are on the profile page We can easily change our basic layout to add this requirment:

const BaseLayout = () => (

<div className="base">

<header>

<p>React Router v5 Browser Example</p>

<nav>

<ul>

<li><Link to='/'>Home</Link></li>

<li><Link to='/about'>About</Link></li>

<li>

<Link to='/me'>Profile</Link>

<Route path="/me" component={ProfileMenu} />

</li>

...

)

So as you can see all Routes with path '/me' will be rendered when we visit '/me' path not just the first match, that's **inclusive routing**.

## Understanding React Router 5 Exclusive Routing

Exclusive routing is the inverse of inclusive routing, it was the default routing in React Router v3 where only the first match is rendered so what if you want exlusive routing back? that also can be done using v5 router using the **Switch** component. In a **Switch** component only the first child **< Route >** or **< Redirect >**, that matches the location, will be rendered. For example:

import { Switch, Route } from 'react-router'

<Switch>

<Route exact path="/" component={HomePage}/>

<Route path="/about" component={AboutPage}/>

<Route path="/me" component={ProfilePage}/>

<Route component={NotFound}/>

</Switch>

## React Router 5 Browser History

**React Router v5** provides a **history** object that exposes a simple API with different implementations (HTML5 history API for dom, legacy hash history for dom, in-memory history for react-native) to manage/manipulate browser history.

You can also navigate inside your React application using methods from the **history** object, for example:

history.push("/my-path")

history.replace("/my-path")

Which are equivalent to:

<Link to="/my-path"/>

<Redirect to="/my-path"/>

## How to Redirect with Redirect Component

Whenever you want to redirect to another location, you can place component which is when rendered will redirect to the location specified in **to** prop that can either be a string or a location object, for example:

<Redirect to={ {

pathname: '/register',

search: '?utm=techiediaries',

state: { referrer: techiediaries.com }

} }/>

Or simply:

<Redirect to="/register"/>

## Conclusion

**React Router v5** makes it dead easy to create React apps with complex UIs that has routing between different portions, you can simply declare a **Router** component such as **BrowserRouter** or **HashRouter** and put,inside of that, a bunch of child **Route**s components that has props which indicate the **path** to be matched and the **component** to be rendered inclusively whenever there is a match (i.e all matched **Route**s will be rendered). In case you need to do exclusive rendering (Just like React Router v3: i.e only the first matched **Route** will rendered) then you can simply use a **Switch** component where only one child (the first found) will be rendered. You can also pass different information between routes using parameters that can be captured and retrieved from the **match** object which gets created once a match is established and then passed to the current rendered component. Finally all building blocks or components of **React Router v5** designed for web apps routing are available and can be imported from **react-router-dom**.