React Routing

Lesson 09

Lesson Objectives

At the end of this module on React fundamentals you will be able to:

- Explain and demonstrate

- Routing and SPAs
- Setting Up the Router Package
- react-router vs react-router-dom
- Preparing the Project For Routing
- Switching Between Pages, Routing-Related Props
- The "withRouter" HOC & Route Props
- Passing & extracting route/query parameters
- Using Switch to Load a Single Route
- Navigating Programmatically

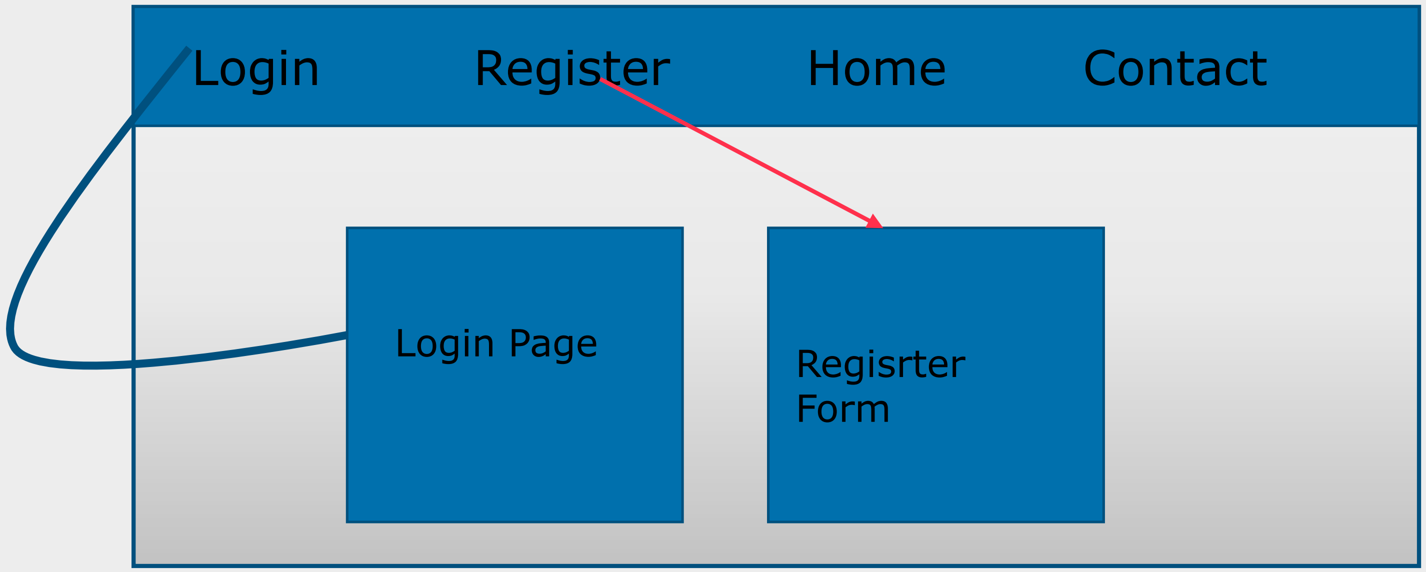
Routing and SPAs

Single-page apps are different from normal multi-page apps. The main difference between them is

- Navigating a single-page app doesn't involve going to an entirely new page. Instead, loads the content within the same page itself.

For Implementing SPA we make use of Routing(React-Router)

- Routing is where you try to map URLs to destinations that aren't physical pages such as the individual views in your single-page app.



What’s in a URL :

A URL is a reference to a web resource

Are building blocks for creating SPA’s.

Single-page applications (SPAs) are web apps that load once and then dynamically update elements on the page using JavaScript. Every React app we've built so far has been a type of SPA.

There are many routing libraries for React, and best of it is **React Router**. React Router gives us a good foundation for building rich application which has views and urls

routing involves two functionality:

(1)Modifying the location of the app (the URL)

(2) determining what React components to render at a given location.

Setting Up the Router Package

To use routing, we have to pull down React, React Router and React DOM:

We have to make use below command for installation

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

React-route vs react-route-dom in React Router v4

The latest version of React Router, v4, is a major shift from its predecessors.

1. With **React router v4**, routing is not centralized anymore instead it becomes a part of the rest of the app layout and UI.
2. 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
3. Introducing new components such as **BrowserRouter** and **HashRouter** for specific use cases
4. No more use of **{props.children}** for nesting components in v4 React Router.
5. React Router v3 routing rules were exclusive meaning only one route will be matched at one time. For v4, routing rules are inclusive meaning multiple routes can be matched and then rendered.

React Router v4 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-route vs react-route-dom in React Router v4 contd.

In React router v4,

`react-router` exports the core components and functions.

`react-router-dom` exports DOM-aware components, like `<Link>` (which renders an `<a>`) and `<BrowserRouter>` (which interacts with the browser's `window.history`).

`react-router-dom` re-exports all of `react-router`'s exports, so you only need to import from `react-router-dom` in your project.

So in future 99% for web development we will be using react-router-dom

React Router's core components

There are three types of components in React Router

1. Router components
2. Route matching components
3. Navigation components.

All of the components that you use in a web application should be imported from react-router-dom.

```
import { BrowserRouter, Route, Link } from 'react-router-dom`
```

React-Routing

A routing library is a key component of any complex, single-page application.
React Router isn’t the only viable solution in the React/Redux ecosystem, its growing and there are many in the system.

React-Redux package allows us to wrap up a component with a container that provides access to the dispatcher and store and , keep the wrapped component synchronized with store mutations.

React-Routing

React isn't a framework, it's a library. Therefore, it doesn't solve all an application's needs.

To create complex SPA (single page application) task like routing requires supporting cast.

React router is one among the front-end router which uses JSX Syntax.

- To install this module : ***npm install react-router***

React router uses <Router> and <Route> components to perform routing.

Like other components <Router> and <Route> does not create DOM, it just defines the rules about how application needs to work based on routes.

```
ReactDOM.render(( <Router>
      //<Route>   Renders Home component  while
vising  the path  /
               <Route path="/" component={Home} />
      </Router> ), document.getElementById('app'));
```

Preparing and setting up react project for routing

Installation

React Router DOM is published to npm so you can install it with either npm

Building a react router firstly we have to import Route and Router from 'react-router' library

import {Router, Route} from 'react-router';

Then add the below code snippet to render method of react Component

```
<Router>
  <Route exact path="/" component={Home}/>
  <Route path="/contactus" component={Contactus}/>
  <Route path="/login" component={Login}/>
  <Route path="/register" component={Register}/>
</Router>
```

The path attribute defines the route URL and component attribute defines the component for this route.

Setting Up react App

Component are used to define the *root*

- <BrowserRouter>
- <Router>

<BrowserRouter /> component is the component where React will replace it's children on a per-route basis.

<Route /> component to create a route available at a specific location available at a url.

(The <Route /> component is mounted at page URLs that match a particular route set up in the route's configuration props.)

To define a route, we'll use the <Route /> component export from react-router and pass it a few props:

- path - The path for the route to be active
- component - The component that defines the view of the route

The <Link /> component requires a prop called to to point to the client-side route where we want to render.

The <Switch /> component will *only render the first matching route* it finds.

Demo

React Router

react-axios-demo-crud(also has roter demo)

Routing related Props

We have three prop choices for how you render a component for a given <Route>: component, render, and children.

component should be used when you have an existing component (either a React.Component or a stateless functional component) that we want to render.

Render, which takes an inline function, should only be used when you have to pass in-scope variables to the component we want to render.

We should not use the component prop with an inline function to pass in-scope variables because you will get undesired component unmounts/remounts.

The "withRouter" HOC & Route Props

You can get access to the history object's properties and the closest <Route>'s match via the withRouter higher-order component. withRouter will pass updated match, location, and history props to the wrapped component whenever it renders.

const ShowTheLocationWithRouter = withRouter(ShowTheLocation);

withRouter does not subscribe to location changes but re-renders after location changes propagate out from the <Router> component. This means that withRouter does *not* re-render on route transitions unless its parent component re-renders.

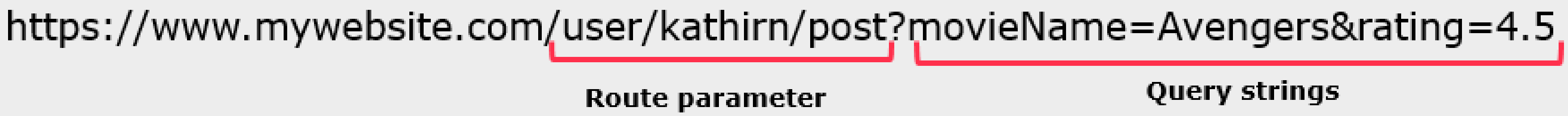
If we are using withRouter to prevent updates from being blocked by shouldComponentUpdate, it is important that withRouter wraps the component that implements shouldComponentUpdate.

To use in program , we have to import as below

import { withRouter } from 'react-router-dom';

Passing & extracting route/query parameters

When you’re building for the web, if you want to pass information via the URL, we use Query Strings. Given below is an example of it, where movieName and rating are query part of query string.



https://www.mywebsite.com/user/kathirn/post?movieName=Avengers&rating=4.5

React Router does not have any opinions about how we parse URL query strings. Some applications use simple key=value query strings, but others embed arrays and objects in the query string. So it's up to us to parse the search string yourself.

In modern browsers that support the URL API, you can instantiate URLSearchParams object fromlocation.search and use that. In browsers that do not support the URL API (read: IE) you can use a 3rd party library such as query-string.

Install query-string using npm and then import in the project

```
npm install query-string
import queryString from 'query-string'
```

Demo

React Router props

react-router-props

React-route-withrouter (need to do)

react-route-query-param

Using Switch to Load a Single Route

<Switch> returns only one first matching route and this will be wrapped around Route components.

Earlier we had an issue with / rendering the Home component and that of other paths. If we create a /welcome/movies path, what happens is, The components of the /welcome and /welcome/movies path will be rendered.

This is where Switch will help in this case again by choosing only one route to render but the most important route must be arranged to come first.

Using Switch, we can avoid what happens in the image above but only for URLs other than /. exact={true} handles it for /.

Switch allows us to specify a route to render if the URL matches no location. For that route, leave the path prop empty.

So finally switch avoids inclusive route rendering.

Navigating Programmatically

The primary way you programmatically navigate using React Router v4 is by using a <Redirect />component

<Redirect /> is
Composable
Declarative
state management (user event -> state change -> re-render)

Syntax of redirect is

<Redirect to="/dashboard"/>

And ensure import is done as shown below

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

Using Switch to Load a Single Route

react-route-switch-demo
react-route-program-programmatically

Summary

By Now you would have been clear with

React Router v4

review

How will you import route and router from ‘react-router’ library

1. import {Router, Route} from 'react-router';
2. import Router, Route from 'react-router';
3. import Router, Route from react-router;
4. import all from 'react-router';