WEB AGE TECHNOLOGY WEBINAR SERIES

Routing and Navigation with React-Router

TOPICS

- Routing and Navigation
- Setting up the Dev Environment for react-router
- Router
- Routes
- Route and Query Parameters
- Routing and Redux

DEMONSTRATION

- Review demo app code

Chapter 1 - React Router

Objectives

Key objectives of this chapter

- Routing and Navigation
- Setting up the Dev Environment for react-router
- Router
- Routes
- Route and Query Parameters
- Routing and Redux

1.1 Routing and Navigation

- Applications typically consist of multiple views
- Routing defines possible view transitions
  - Happens at design-time
  - Involves definition of routes
- Navigation executes specific view transitions
  - Navigational links are defined at design-time
  - Links are invoked by an end-user at run-time.

1.2 react-router

- The react-router and related packages allow developers to implement routing and navigation between React components.
- Several packages are available:
  - react-router - core package
  - react-router-dom - for use with web applications
  - react-router-native - for use with react-native applications
  - react-router-redux - for web apps that also use redux
  - react-router-config - for supports static routes for server-side rendering
- We will be looking at **react-router-dom** in this chapter

1.3 Creating a react-router based project

Steps:

- Use the Create React App utility to create a basic React project.

```
npm install create-react-app
create-react-app app-name
```

- Install react-router-dom

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

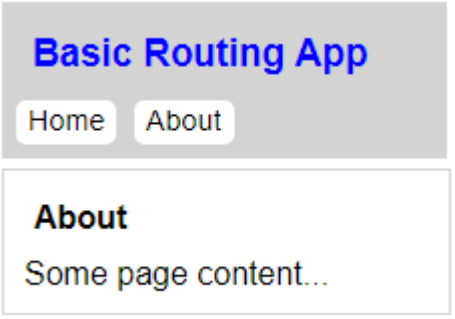
- Import to the file that defines the component that implements routing:

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

1.4 A Basic Routed Component

This sample app defines two routes

- One that shows the Home component
- One that shows the About component
- Code is on the next page



1.5 A Basic Routed Component

- Basic Routing Code:

```
const BasicRoutingComponent = function(){
  return (
    <BrowserRouter>
      <div>
        <PageNav />
        <section>
          <Route exact path="/" component={Home} />
          <Route path="/about" component={About} />
        </section>
      </div>
    </BrowserRouter>
  )
}
```

- Note how the <BrowserRouter> component wraps around the navigation and Routes sections.
  - The <BrowserRouter> component can only have one child element. Here it is .

Notes:

The complete component listing:

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

const BasicRoutingComponent = function(){
  return (
    <Router>
      <div>
        <PageNav />
        <section >
          <Route exact path="/" component={Home} />
          <Route path="/about" component={About} />
        </section>
      </div>
    </Router>
  )
}

const PageNav = () => (
  <nav>
    <h3>Basic Routing App</h3>
    <Link to="/">Home</Link> 
    <Link to="/about">About</Link> 
  </nav>
);



...
```

```
const Home = () => (
  <div>
    <h4>Home</h4>
    <p>Some page content...</p>
  </div>
)

const About = () => (
  <div>
    <h4>About</h4>
    <p>Some page content...</p>
  </div>
)

export default BasicRoutingComponent
```

1.6 Router vs. BrowserRouter

- It is common to alias BrowserRouter like this:

```
import { BrowserRouter as Router}
from 'react-router-dom'
```

- This allows you to use the shorter term 'Router' in your JSX code

```
<Router>
  <div>
    <ul>
      <li><Link to="/">Home</Link></li>
      <li><Link to="/about">About</Link></li>
    </ul>
    <hr/>
    <Route exact path="/" component={Home}/>
    <Route path="/about" component={About}/>
  </div>
</Router>
```

1.7 The Route component

- Is used to define routes
- Main attributes:
  - **path** - the route is chosen when a navigation matches the path
  - **component** - when the path is matched this component is displayed
  - **exact** - when this is specified only an "exact" path match will choose the route.
    - <Route path="/stuff" … /> matches these locations:

      ```
      \stuff, \stuff\more, \stuff\more\1, etc.
      ```

    - <Route exact path="/stuff" … /> matches only this location:

      ```
      \stuff
      ```

1.8 <Switch>

- All Route components with matching paths are displayed. This allows for more than one to be shown at the same time:
- If you need to have only one Route in a set of Routes to be displayed you need to wrap the Routes with the <Switch> component

```
<Router>
  ...
  <Switch>
    <Route exact path="/" component={Home}/>
    <Route path="/about" component={About}/>
    <Route path="*" component={Default} />
  </Switch>
</Router>
```

- If a navigation path doesn't match an existing Route then:
  - The previously matched Route component is removed
  - No new component is displayed
- You can use the "*" path along with <Switch> to create a default route that gets displayed when no other Route matches

1.9 Redirect Route

- Sometimes you need to redirect from a path provided in a navigation to a different path that matches an existing route
- You would use the <Redirect> component to do this:

```
<Router>
  ...
  <Switch>
    <Redirect from="/" exact to="/home" />
    <Route path="/home" component={Home}/>
    <Route path="/about" component={About}/>
    <Route path="*" component={Default} />
  </Switch>
</Router>
```

- Here we are redirecting from the exact root path "/" to the "/home" path

1.10 Navigating with <Link>

- Routes are activated when a user clicks on an anchor link <a>
- Anchor links for navigation are created by using the react-router-dom <Link> element

```
<Router>
  <Link to="/">Home</Link>
  <Link to="/about">About</Link>
       ...
</Router>
```

- <Link> elements must be nested inside <Router>
- The "to" attribute provides the path you want to navigate to
- The replace attribute is used when you want the new path location to replace the existing one in the browser's history instead of adding to it:

```
<Link to="/about" replace >About</Link>
```

1.11 Navigating with <NavLink>

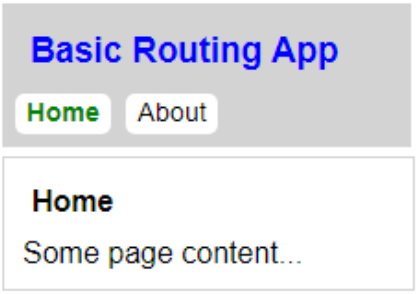
The currently selected link can be highlighted by using <NavLink> instead of <Link>

```
<NavLink to="/home" >Home</NavLink>
```

Then adding a CSS style for the "active":

```
.active{
    color: green;
    font-weight:bold;
}
```

A class name other than 'active' can be specified with the 'activeClassName' attribute



(the home link above is shown with the "active" styling)

1.12 Route Parameters

- Route Parameters can be used to pass data to a component during navigation.
- Route parameters are added to a Route's path like this:

```
<Route path='/details/:index'
component={ProductDetails} />
```

- ":index" which appears in the above Route will hold whatever value is placed after '/details/' when navigating:
- For example for this link:

```
<Link to='/details/P001'>Details</Link>
```

- Index is equal to:

```
"P001"
```

1.13 Retrieving Route Parameters

- To make use of a Route Parameter that has been passed to it a component first needs to retrieve its value.

```
const ProductDetails = (params) => {
  let index = params.match.params.index;
  let selectedProduct = state.products[index];
  return (
    <div>{selectedProduct.name}, … </div>
  )
}
```

- Notice how the "index" route parameter value is retrieved using:

```
params.match.params.index
```

1.14 QueryString Parameters

- Query String parameters are another way to pass data to a component during navigation

```
<NavLink
  to='/sendmail?name=mark&email=mark@abc.com'
>Send Email</NavLink>
```

- This link will match the following Route, even with 'exact' specified:

```
<Route exact path='/about' component={About} />
```

- The parameter string can be retrieved via the props object:

```
props.location.search
(retrieves: "?name=mark&email=mark@abc.com" )
```

- To get individual parameters you can do this (or use a 3rd party lib):

```
const search = props.location.search;
const params = new URLSearchParams(search);
const name = params.get('name');
const email = params.get('email');
```

1.15 Using Router with Redux

- React-Router is compatible with Redux - both can be used together.
- In this case the Redux <Provider> wraps around the react-router <Router> which then wraps the application's Routes

```
// in index.js
ReactDOM.render(
  <Provider store={store}>
    <Router>
      <div>
        <h2>React, Redux, Route App</h2>
        <Link to="/">Home</Link>
        <Link to="/about">About</Link>
        <Route exact path="/" component={Home}/>
        <Route path="/about" component={About}/>
      </div>
    </Router>
  </Provider>,
  document.getElementById('root')
);
...
```

- If you need to be able to rewind actions using Time Travel you will
- More sophisticated users may wish to add the *react-router-redux* package. This package incorporates the app's location url into the redux state thus allowing you to rewind navigational actions use Time Travel.

1.16 Summary

In this Chapter we Covered:

- Routing and Navigation
- Setting up the Dev Environment for react-router
- Router
- Routes
- Route and Query Parameters
- Routing and Redux

UPCOMING CLASSES

Register Now

- WA2583 React JavaScript Programming
- WA2488 JavaScript changes with ECMAScript 2015 Training
- WA2676 MERN JavaScript Technology Stack

QUESTIONS?

CONTACT

Web Age Solutions

www.webagesolutions.com/contactus/

US - 215-517-6540

Canada - 1-866-206-4644