REACT ROUTER

ROUTER

Routing is the process of keeping the browser URL in sync with what's being rendered on the page. React Router lets you handle routing declaratively. The declarative routing approach allows you to control the data flow in your application, by saying "the route should look like this":

```
<Route path="/contact" component={Contact}/>
```

React Router is a third-party library that's widely popular for its design and simplicity

Setting up React Router

The React Router library comprises three packages: react-router, react-router-dom, and react-router-native. react-router is the core package for the router, whereas the other two are environment specific. You should use react-router-dom if you're building a website, react-router-native if you're on a mobile app development environment.

Use npm to install react-router-dom

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

Router

You need a router component and several route components to set up a basic route. Since we're building a browser-based application, we can use two types of routers from the React Router API:

```
<BrowserRouter></BrowserRouter>

<HashRouter></HashRouter>
```

The **<BrowserRouter>** is more popular between the two because it uses the HTML5 History API to keep track of your router history. The **<HashRouter>,** on the other hand, uses the hash portion of the URL (window.location.hash) to remember things. If you intend to support legacy browsers, you should stick with **<HashRouter>.**

Note: A router component can only have a single child element. The child element can be an HTML element – such as a div – or a React component.

Le's explore some examples:

Create Home, About, Students, Teachers and App components

Home.js

```
import React from 'react'
const Home = ()=>{
  return(
```

```
    <h2>Home</h2>
  )
}
export default Home
```

About.js

```
import React from 'react'
const About = ()=>{
  return(
    <h2>About</h2>
  )
}
export default About
```

Students.js

```
import React from 'react'

const Students = ()=>{
  return(
    <div>
      <h2>Students</h2>
    </div>
  )
}

export default Students
```

Teachers.js

```
import React from 'react'

const Teachers = ()=>{
  return(
    <h2>Teachers</h2>
  )
}

export default Teachers
```

App.js

```
import React, {Component} from 'react'

class App extends Component {
    render(){
      return(
        <div>
        </div>

      )
    }
}
export default App
```

Next, create an index.js file, import ReactDom from 'react-dom', {BrowserRouter} from 'react-router-dom' and import App component also.

index.js

```
import React from 'react';
import ReactDom from 'react-dom';
import {BrowserRouter} from 'react-router-dom'
import App from './App'


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

We will now import all other components into the App component like this:

```
import React, {Component} from 'react'
import Home from './Home'
import About from './About'
import Students from './Students'
import Teachers from './Teachers'


class App extends Component {
    render(){
      return(
        <div>
        </div>
      )
    }
}


export default App
```

We have imported the needed components. We would need to navigate to the different components. To achieve this, let's import {Route, Link} from 'react-router-dom'. The next lines of codes will show you how to implement that

```
import React, {Component} from 'react'
import {Route, Link} from 'react-router-dom'
import Home from './Home'
import About from './About'
import Students from './Students'
import Teachers from './Teachers'

class App extends Component {
    render(){
      return(
        <div>
          <ul>
            <li><Link to="/">Home</Link></li>
            <li><Link to="/about">About</Link></li>
            <li><Link to="/students">Students</Link></li>
            <li><Link to="/teachers">Teachers</Link></li>
          </ul>
```

```
          <Route exact path="/" component={Home} />
          <Route path="/about" component={About} />
          <Route path="/students" component={Students} />
          <Route path="/teachers" component={Teachers} />
        </div>

      )
    }
}
export default App
```

With the example above, we can now navigate to different components.

Redirect

Like the server-side redirects, <Redirect> will replace the current location in the history stack with a new location. The new location is specified by the "to" props.
If you want to redirect from a particular component to an About component:
1. import {Redirect} from 'react-router-dom' in the component you want to redirect from
2. Place <Redirect to='./about' /> in the return function

Higher Order omponents

The explaination of HOC is best explained in React Online Documntation. This is how they explained it.
A higher-order component (HOC) is an advanced technique in React for reusing component logic. HOCs are not part of the React API, per se. They are a pattern that emerges from React's compositional nature.
Concretely, a higher-order component is a function that takes a component and returns a new component.
const EnhancedComponent = higherOrderComponent(WrappedComponent);

Whereas a component transforms props into UI, a higher-order component transforms a component into another component.
HOCs are common in third-party React libraries, such as Redux's connect and Relay's createFragmentContainer.
Let's explore this example:

```
import React, {Component} from 'react'
const CursiveFont = BaseComponent=> {
  return class EnhancedComponent extends Component{
```

```
    render(){
      const styles = {
        fontFamily:"cursive",
        color:"green"
      }
      return(
        <BaseComponent styles={styles} />
      )
    }
  }
}
export default CursiveFont
```

Then, you require the HOC in any component that is needed. To make use of it in the Home component, we would import CursiveFont to supercharge the Home component

```
import React from 'react'
import CursiveFont from './CursiveFont'
const Home = (props)=>{
  return(
    <h2 style={props.styles}>Home</h2>
  )
}
export default CursiveFont(Home)
```

Axios

Axios is one of the most downloaded NPM package even though React.JS comes with its native FETCH API which supposedly does the same job as Axios does.

Axios is a Javascript library used to make http requests from node.js or XMLHttpRequests from the browser and it supports the Promise API that is native to JS ES6. You can perform CRUD methods with axios. For more on this you can click on this link https://github.com/axios/axios

To use axios in our react example, we would have to install axios
$ npm install axios --save
Then, import it into the Component where you want to make use of it.
import axios from 'axios'

Let's explore this example:
Create a Post component and import axios

Post.js

```
import React, {Component} from 'react'
import axios from 'axios'
```

```
class Posts extends Component{
  constructor(){
    super();
    this.state = {
     posts:[]

    }
  }
  componentDidMount(){
    axios.get('https://jsonplaceholder.typicode.com/posts')
    .then(res=>{
      this.setState({posts:res.data.slice(0, 10)})
    })
  }
  render(){
    const posts = this.state.posts.map(post=>{
      return(
        <tr key={post.id}>
          <td>{post.id}</td>
          <td><a href="#">{post.title}</a></td>
          <td>{post.body}</td>
        </tr>
      )
    })
    return(
      <div>
        <h1>Posts</h1>
        <table>
          <tbody>
            <tr>
              <th>ID</th>
              <th>Title</th>
              <th>Body</th>
            </tr>
            {posts}
          </tbody>
        </table>
      </div>
    )
 }
}
export default Posts
```

In the code above, we used lifecyle method, componentDidMount().

From the React docs: componentDidMount() is invoked immediately after a component is mounted (inserted into the tree). Initialization that requires DOM nodes should go here. If you need to load data from a remote endpoint, this is a good place to instantiate the network request.

```
componentDidMount(){
    axios.get('https://jsonplaceholder.typicode.com/posts')
    .then(res=>{
      this.setState({posts:res.data.slice(0, 10)})
    })
  }
```

https://jsonplaceholder.typicode.com/posts is an online API that makes JSON available for developers to practice.
We will use axios.get90 method to retrieve data from https://jsonplaceholder.typicode.com/posts

Then, update the App component like this:

```
import React, {Component} from 'react'
import {Route, Link} from 'react-router-dom'
import Home from './Home'
import About from './About'
import Students from './Students'
import Teachers from './Teachers'
import Posts from './Posts'

class App extends Component {
    render(){
      return(
        <div>
          <ul>
            <li><Link to="/">Home</Link></li>
            <li><Link to="/about">About</Link></li>
            <li><Link to="/students">Students</Link></li>
            <li><Link to="/teachers">Teachers</Link></li>
            <li><Link to="/posts">Posts</Link></li>
          </ul>
          <Route exact path="/" component={Home} />
          <Route path="/about" component={About} />
          <Route path="/students" component={Students} />
          <Route path="/teachers" component={Teachers} />
          <Route path="/posts" component={Posts} />
        </div>
      )
    }
}
export default App
```

Route Parameters

Route parameters are about creation of sub-categories. So as to get a particular post after clicking on its title, we would have to update the Posts component.

```
Replace <td><a href="#">{post.title}</a></td>  with
<td><Link to={`post/${post.id}`}>{post.title}</Link></td>
```

Since, we are using Link, we need to import Link like this:
import {Link} from 'react-router-dom'
Let's create a Post component

Post.js

```
import React, {Component} from 'react'
import {Link} from 'react-router-dom'
import axios from 'axios'
class Post extends Component{
  constructor(){
    super();
    this.state = {
      posts:[]
    }
  }
  componentDidMount(){
    let id = this.props.match.params.id;
    axios.get('https://jsonplaceholder.typicode.com/posts/'+id)
    .then(res=>{
      this.setState({posts:[res.data]})
    })
  }
  render(){
    const post = this.state.posts.map(post=>{
      return(
        <tr key={post.id}>
          <td>{post.id}</td>
          <td><Link to={`${this.props.match.url}`}>{post.title}</Link></td>
          <td>{post.body}</td>
        </tr>
      )
    })
    return(
      <div>
        <h1 className="text-center">Post {this.props.match.params.id}</h1>
        <table className="post-table">
          <tbody>
            <tr>
              <th>ID</th>
              <th>Title</th>
              <th>Body</th>
            </tr>
            {post}
          </tbody>
        </table>
      </div>
    )
  }
}
export default Post
```

Finally,

1. import Post from './Post' in the App component
2. add <Route path="/post/:id" component={Post} /> to the App

Switch Tag

Renders the first child <Route> or <Redirect> that matches the location
<Switch> is unique in that it renders a route exclusively. In contrast, every <Route> that matches the location renders inclusively.
Occasionally, however, we want to pick only one <Route> to render. If we're at /post, we don't want to also match post/:id.
To make use of the Switch,
1. import {Route, Switch} from 'react-router-dom'
2. Wrap the Routes in the Switch tag

```
<Switch>
<Route exact path="/" component={Home} />
<Route path="/about" component={About} />
<Route path="/students" component={Students} />
<Route path="/teachers" component={Teachers} />
<Route path="/posts" component={Posts} />
<Route path="/post/:id" component={Post} />
</Switch>
```

Import Images

Importing images in React is very simple. You just have to import the image file into the file where you want to make use of it.
import image from '../images/image.jpg'
Then, you make use of it like this,

```
<img src={image} />
```