Reactjs – 4

Margit Tennosaar

Last session:

- SPA

- function components

- class components

- Props and states

- Handling events

- Manipulating states

This session

- Styling in react
- Conditional rendering
- React Router

Styling in react

Styling in react

external (and inline styling)

```
render() {
  return <div className="menu navigation-menu">
           <h1>Hello World1!</h1>
           <h1 style={{color: "red"}}>Hello World2!</h1>Menu
  </div>
}
```

https://reactjs.org/docs/faq-styling.html

Use in props and states

```
render() {
  let className = 'menu';

  if (this.props.isActive) {
    className += ' menu-active';
  }
  return <span className={className}>Menu</span>
}
```

https://reactjs.org/docs/dom-elements.html#style

As JS object

```
render() {
    const style = {
      backgroundColor: "blue",
      border: "1px solid green",
      padding: "10px",
      cursor: "pointer"
    };

    return (
      <div className="App">
        <h1>Hi, I'm a React App</h1>
        <button style={style} onClick={this.addLikes}> Like </button>
       </div>
)
```

https://reactjs.org/docs/dom-elements.html#style

Conditional rendering

```
function Greeting(props) {
  const isLoggedIn = props.isLoggedIn;
    if (isLoggedIn) {
      return <UserGreeting />;
      }
  return <GuestGreeting />;
}

ReactDOM.render(<Greeting isLoggedIn={false}/>,
document.getElementById('root')
);
```

Example from counter

```
render() {
    let circleClass = "";

    if (this.state.counter === 0) {
      circleClass = ”circle";
    } else if (this.state.counter % 2 === 0) {
      circleClass = ”circle even";
    } else {
      circleClass = ”circle odd";
    }

    return (
      <div>
        <Header />
        <main>
          <h1 className={circleClass}>{this.state.counter}</h1>
```

Example from counter

```
render() {
    let circleClass = "";

    this.state.counter === 0
      ? (circleClass = ”circle")
      : this.state.counter % 2 === 0
      ? (circleClass = ”circle even")
      : (circleClass = ”circle odd");

    return (
      <div>
        <Header />
        <main>
          <h1 className={circleClass}>Total likes: {this.state.counter}</h1>
```

React Router

React Router

React Router, and dynamic, client-side routing, allows us to build a single-page web application with navigation without the page refreshing as the user navigates.

https://reacttraining.com/react-router/web/guides/quick-start

Routing means conditional rendering of components based on the url in the browser.

When the URL in the address bar changes, the content of the page is only manipulated using Javascript, and the browser will not load new content form the server. Using the back and forward actions, as well as making bookmarks, is still logical like on a traditional web page.

How to use

Install

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

Import

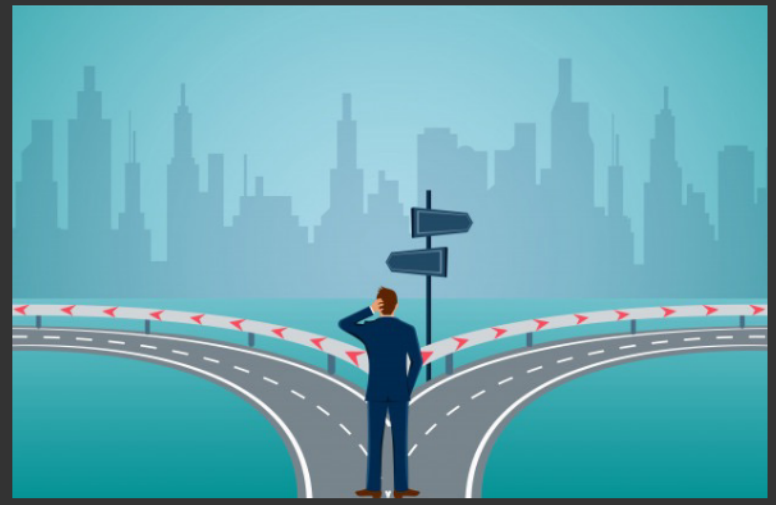
```
import { BrowserRouter as Router, Switch, Route, Link } from "react-router-dom";
```

How to make happen?

Routers - BrowserRouter as Router and HashRouter

Route matchers - Route and Switch

And navigation - Link, NavLink, and Redirect



Routers

The Router component needs to be the highest parent. For that wrap your App component inside Router component:

```
const App = () => {
  return (
    <Router>
      <div>Hello World!</div>
    </Router>
   );
};
```

or

```
ReactDom.render(<Router><App /></Router>)
```

BrowserRouter and HashRouter

The main difference between the two is the way they store the URL and communicate with your web server.

A <BrowserRouter> uses regular URL paths. These are generally the best-looking URLs, but they require your server to be configured correctly. Create React App supports this out of the box in development, and comes with instructions on how to configure your production server as well.

A <HashRouter> stores the current location in the hash portion of the URL, so the URL looks something like http://example.com/#/your/page. Since the hash is never sent to the server, this means that no special server configuration is needed.

https://reactrouter.com/web/guides/primary-components/routers

Route Matchers

There are two route matching components: Switch and Route. When a <Switch> is rendered, it searches through its children <Route> elements to find one whose path matches the current URL. When it finds one, it renders that <Route> and ignores all others. This means that you should put <Route>s with more specific (typically longer) paths before less-specific ones.

If no <Route> matches, the <Switch> renders nothing (null).

https://reactrouter.com/web/guides/primary-components/route-matchers

Navigation (or Route Changers)

React Router provides a <Link> component to create links in your application. Wherever you render a <Link>, an anchor (<a>) will be rendered in your HTML document.

The <NavLink> is a special type of <Link> that can style itself as “active” when its to prop matches the current location.

Any time that you want to force navigation, you can render a <Redirect>. When a <Redirect> renders, it will navigate using its to prop.

https://reactrouter.com/web/guides/primary-components/navigation-or-route-changers

example

Simple routing

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

const Home = () => {
    return (
      <div> <h1>This is HOME page</h1> </div>
    );
  };

const About = () => {
    return (
      <div> <h1>This is ABOUT page</h1> </div>
    );
  };

const Contact = () => {
    return (
      <div> <h1>This is CONTACT page</h1> </div>
    );
  };
```

```
const App = () => {
  return (
    <Router> // BrowserRouter as Router
      <nav>
        <ul>
         <li>
          <Link to="/">Home</Link> // Navigation
         </li>
         <li>
          <Link to="/about">About</Link>
         </li>
         <li>
          <Link to="/contact">Contact</Link>
         </li>
        </ul>
      </nav>
      <main>
        <Switch> // Route matcher
         <Route path="/" exact component={Home} /> // Route
         <Route path="/about" component={About} />
         <Route path="/contact" component={Contact} />
        </Switch>
      </main>
    </Router>
  );
};
```

Let’s test it out

Task – building the base app

The idea is to make a blog/homepage for cooking and recipes

- Make a clean create-react-app
- Clean it and start to work with it
- Make App, Header, Main, Footer, Nav, Home, Recipes, About components
- Using React Router set up simple navigation with three pages – home, recipes, about
- Start styling your app

Prototype to project

Reading list until next time:

https://www.w3schools.com/react/react_intro.asp
https://www.w3schools.com/react/react_getstarted.asp
https://www.w3schools.com/react/react_es6.asp
https://www.w3schools.com/react/react_render.asp
https://www.w3schools.com/react/react_jsx.asp
https://www.w3schools.com/react/react_components.asp