React.4

▼ What is the <Switch /> component?

The <Switch /> component will wrap several <Route /> components and will render only the **first** <Route /> that matches the current browser URL.

The <Route /> has to be a direct descendant of the <Switch />. That means that you should not have a or other elements in between the <Switch /> and the <Route />.

```
import React from "react";
import {BrowserRouter, Switch, Route} from "react-router-dom";

function App() {
    return <BrowserRouter>
        <div>The rest of your app goes here</div>

        <Switch>
            <Route exact path="/about">
                <About />
            </Route>
            <Route exact path="/">
                <Home />
            </Route>
        </Switch>
    </BrowserRouter>;
}
```

▼ What is the <Link /> component?

The `<Link />` component takes a `to` prop. When the user clicks on the link, React Router will navigate to the path provided by the **to** prop.

This `<Link />` element will render an `<a>` element, but with some event handlers that will **instruct React Router to do an instant redirect instead of a page navigation**.

So React Router is performing an `event.preventDefault()` to prevent the page from navigating to a new page. You don't have to do that, it's all declarative.

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

function App() {
  return (
    <BrowserRouter>
        <nav>
          <ul>
            <li>
              <Link to="/">Home</Link>
            </li>
            <li>
              <Link to="/contact">Contact</Link>
            </li>
          </ul>
        </nav>

        {/* Switch and Route goes here*/}
    </BrowserRouter>
  );
}
```

▼ What does the HTML <nav> element represent?

section of a page whose purpose is to provide navigation links

▼ What does the HTML <main> element represent?

the dominant content of the <body> of a document

▼ Why place <Switch /> and <Route /> components inside the <main> element?

you're instructing the browser, search engines, etc. that this is where the main content is on your page

▼ Why do we use React Router's URL parameters?

to create pages that will have the same page structure but the data will be different

e.g. different product pages but they are all Product component

```
<Route exact path="/products/:id">
    <Product />
```

```
</Route>
```

▼ What does the useParams() hook allow you to do?

The useParams() hook allows us to extract the parameters from the URL.

▼ What does the useParams() hook return?

an object containing the URL parameters as key/value pairs

```
// Product.js
import React from "react";
import {useParams} from "react-router-dom";

function Product() {
  const params = useParams();
  const id = params.id; // "42"
  const type = params.type; // "food"

  return <h2>Product</h2>;
}
```

The values of the URL parameters are always in strings because they are extracted from the URL.

▼ What is the Effect Hook, useEffect()?

The Effect Hook, useEffect, adds the ability to perform side effects from a function component. It serves the same purpose as componentDidMount, componentDidUpdate, and componentWillUnmount in React classes, but unified into a single API.

When you call useEffect, you're telling React to run your "effect" function after flushing changes to the DOM.

▼ Product Details page (example)

```
// index.js
import React from "react";
import {render} from "react-dom";
import {BrowserRouter, Switch, Route} from "react-router-dom";
import StoreFront from "./StoreFront.js";
import ProductDetails from "./ProductDetails.js";

function App() {
```

```
    return (
        <BrowserRouter>
            <Switch>
                <Route exact path="/">
                    <StoreFront />
                </Route>
                <Route exact path="/products/:id">
                    <ProductDetails />
                </Route>
            </Switch>
        </BrowserRouter>
    );
}

render(<App />, document.querySelector("#react-root"));
```

```
// StoreFront.js

import React, {useEffect, useState} from "react";
import Product from "./Product.js";
import Loader from "./Loader.js";
import useFetch from "./useFetch.js";

export default function StoreFront() {
    const [products, setProducts] = useState([]);
    const {get, loading} = useFetch("https://react-tutorial-demo.firebaseio.com/");

    useEffect(() => {
        get("products.json")
        .then(data => {
            setProducts(data);
        })
        .catch(error => console.log(error))
    }, []);

    return <div className="store-front">
        {loading && <Loader />}
        {products.map(product => <Product key={product.id} details={product} />)}
    </div>;
}
```

```
// Product.js
import React, {useState} from "react";
import {Link} from "react-router-dom";

export default function Product(props) {
    const [count, setCount] = useState(0);

    const {details} = props;
```

```
    function handleIncrementClick() {
        setCount(count + 1);
    }
    function handleDecrementClick() {
        if (count > 0){
            setCount(count - 1);
        }
    }

    return (
    <Link to={`products/${details.id}`}>
        <div className="product">
            <img src={details.image} width="50" alt={details.name} />
            <div className="product-info">
                <h2>{details.name}</h2>
                <p>{details.description}</p>
            </div>
        </div>
    </Link>
    );
}
```

```
// ProductDetails.js
import React, {useEffect, useState} from "react";
import {useParams, Link} from "react-router-dom";
import useFetch from "./useFetch.js";

export default function ProductDetails () {
    const params = useParams();
    const [product, setProduct] = useState({});
    const {get} = useFetch("https://react-tutorial-demo.firebaseio.com/");

    const id = params.id;

    useEffect(() => {
        get(`productdetails/id${id}.json`)
        .then(data => {
            setProduct(data);
        })
        .catch(error => console.log(error))
    }, []);

    return <div>
        <Link to="/">Back home</Link>
        {product && <div>
            <h2>{product.name}</h2>
            <p>{product.description}</p>
            <h3>${product.price}</h3>
            <img src={product.image} width="100" />
        </div>}
```

```
    </div>
}
```

▼ What are nested routes?

it's the rendering of two routes at the same time

aka two different contents from two routes at the same time

e.g. /about/us page is rendering the content from the /about route as well as the content from the /about/us route

▼ Nested routes (example)

Making nested routes work in React Router is a 2 step process:

1. You can make **any** component return a (nested) `<Switch />` which contains routes. In our example, the `<About />` component will return a `<Switch />`.
2. The route where you'd like to have nested routes support should **NOT** have the `exact` prop anymore. In our example, the `/about` supports nested routes so we should remove the `exact` prop from that `<Route />`.

Whenever you need to use nested routing, you **have** to get rid of the `exact` prop only on that specific route.

That's because if you keep the `exact` prop on the `<Route path="/about">`, it won't match for `/about/us` or `/about/team`. So you won't get any nested routes.

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

function About() {
  return <>
      <h1>About</h1>
      <ul>
        <li><Link to="/about/us">About us</Link></li>
        <li><Link to="/about/team">About the team</Link></li>
      </ul>

      <Switch>
        <Route exact path="/about/us">
          <h2>About us</h2>
        </Route>
        <Route exact path="/about/team">
```

```
          <h2>About the Team</h2>
        </Route>
      </Switch>
    </>;
}

function App() {
    return (
        <BrowserRouter>
          <Switch>
            <Route exact path="/">
                <Link to="/about">Go to /about page</Link>
            </Route>
            <Route path="/about">
                <About />
            </Route>
          </Switch>
        </BrowserRouter>
    );
}

render(<App />, document.querySelector("#react-root"));
```

▼ How will React Router match without an exact prop?

When the exact prop is NOT used in the Route, then React router will match the beginning of the URL. If the beginning of the URL matches, then the <Route /> will be considered matching.

<Route path="/about">...</Route>

this will match everything after /about

▼ Why do we need Switch component?

The `<Switch />` component will always make sure to render the **first** (and only the **first**) `<Route />` that matches the path.Once it finds this route, it will skip the remaining routes.

Whereas if you omit the `<Switch />` and directly define your routes, then several routes **could** be rendered at the same times if they match the current path.

▼ useRouteMatch

a hook from React Router that returns an object describing the path, url and, params of the current route that matched the browser URL

▼