Week 6: React Router DOM

React Router Dom is a ReactJS library that helps to switch between React components based on the URL.

If you have multiple components and you only want to display one by one.

Tutorial 1: Installation

- Open command prompt or Terminal (Mac) and run: **npm install react-router-dom** (you can use yarn)

```
thaoanh@Thaos-MacBook-Air week4reactapp % cd ..
thaoanh@Thaos-MacBook-Air Downloads % npm install react-router-dom

added 19 packages, and audited 20 packages in 2s

found 0 vulnerabilities
thaoanh@Thaos-MacBook-Air Downloads %
```

- Import React router dom components:

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

Import with and without {}:
https://stackoverflow.com/questions/36795819/when-should-i-use-curly-braces-for-es6-import

Tutorial 2: Basic routing

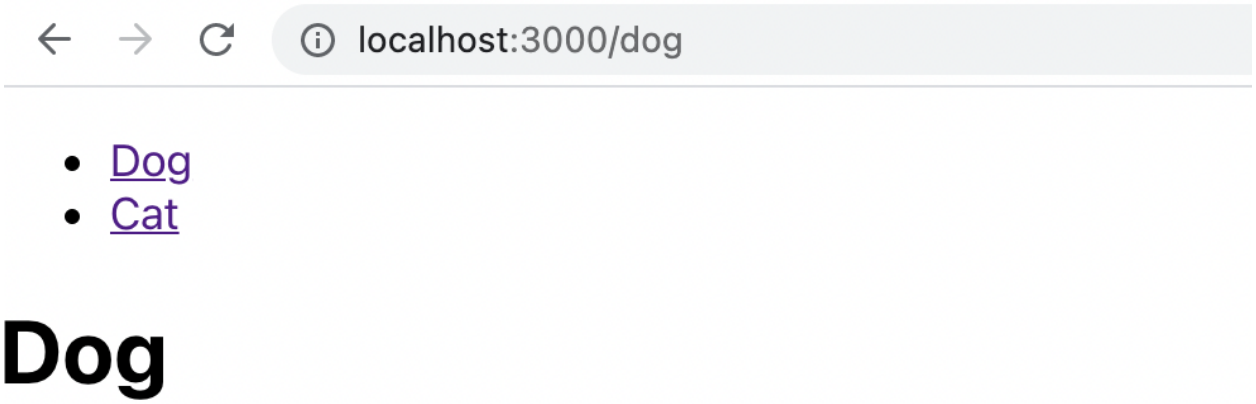
- Setup a simple routing in App.js

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


function Dog(){
 return(
   <h1>Dog</h1>
 )
}
```

```
function Cat(){
 return(
   <h1>Cat</h1>
 )
}

function App() {

 return (
   <div>

    <Router>
     <div>
       <nav>
         <ul>
           <li>
             <Link to="/dog">Dog</Link>
           </li>
           <li>
             <Link to="/cat">Cat</Link>
           </li>
         </ul>
       </nav>

       {/* A <Switch> looks through its children <Route>s and
           renders the first one that matches the current URL. */}
       <Switch>
         <Route path="/dog">
           <Dog />
         </Route>
         <Route path="/cat">
           <Cat />
         </Route>
       </Switch>
     </div>
   </Router>

   </div>
```

```
 );
}

export default App;
```

- This is what you see when you run it:



You may see lots of problems Hook Invalid calls when using router. Check this out: https://reactjs.org/warnings/invalid-hook-call-warning.html

Tutorial 3: With params

Use useParams hook to get path variable from URL

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

```
// Params are placeholders in the URL that begin
// with a colon, like the `:id` param defined in
// the route in this example. A similar convention
// is used for matching dynamic segments in other
// popular web frameworks like Rails and Express.

export default function App() {
 return (
   <Router>
     <div>
       <h2>Accounts</h2>

       <ul>
         <li>
           <Link to="/netflix">Netflix</Link>
         </li>
         <li>
           <Link to="/zillow-group">Zillow Group</Link>
         </li>
         <li>
           <Link to="/yahoo">Yahoo</Link>
         </li>
         <li>
           <Link to="/modus-create">Modus Create</Link>
         </li>
       </ul>

       <Switch>
         <Route path="/:id" children={<Child />} />
       </Switch>
     </div>
   </Router>
 );
}

function Child() {
 // We can use the `useParams` hook here to access
 // the dynamic pieces of the URL.
 let { id } = useParams();
```

```
 return (
   <div>
     <h3>ID: {id}</h3>
   </div>
 );
}
```

Alternatively, you an use:

```
 <Switch>
         <Route path="/:id"><Child/></Route>
       </Switch>
```

Exercises:

- Add another params called name
- Build a components with at least 3 params

Tutorial 4: Query params

Similar to path params, react router can work with query params (?=&)

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

// React Router does not have any opinions about
// how you should parse URL query strings.
//
// If you use simple key=value query strings and
// you do not need to support IE 11, you can use
// the browser's built-in URLSearchParams API.
//
// If your query strings contain array or object
// syntax, you'll probably need to bring your own
```

```
// query parsing function.

export default function QueryParamsExample() {
 return (
   <Router>
     <QueryParamsDemo />
   </Router>
 );
}

// A custom hook that builds on useLocation to parse
// the query string for you.
function useQuery() {
 return new URLSearchParams(useLocation().search);
}

function QueryParamsDemo() {
 let query = useQuery();

 return (
   <div>
     <div>
       <h2>Accounts</h2>
       <ul>
         <li>
           <Link to="/account?name=netflix">Netflix</Link>
         </li>
         <li>
           <Link to="/account?name=zillow-group">Zillow Group</Link>
         </li>
         <li>
           <Link to="/account?name=yahoo">Yahoo</Link>
         </li>
         <li>
           <Link to="/account?name=modus-create">Modus Create</Link>
         </li>
       </ul>

       <Child name={query.get("name")} />
```

```
      </div>
    </div>
  );
}

function Child({ name }) {
  return (
    <div>
      {name ? (
        <h3>
          The <code>name</code> in the query string is &amp;quot;{name}
          &amp;quot;
        </h3>
      ) : (
        <h3>There is no name in the query string</h3>
      )}
    </div>
  );
}
```

Tutorial 5: Nesting

Use link/switch inside a child component

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

// Since routes are regular React components, they
```

```
// may be rendered anywhere in the app, including in
// child elements.
//
// This helps when it's time to code-split your app
// into multiple bundles because code-splitting a
// React Router app is the same as code-splitting
// any other React app.

export default function NestingExample() {
 return (
   <Router>
     <div>
       <ul>
         <li>
           <Link to="/">Home</Link>
         </li>
         <li>
           <Link to="/topics">Topics</Link>
         </li>
       </ul>

       <hr />

       <Switch>
         <Route exact path="/">
           <Home />
         </Route>
         <Route path="/topics">
           <Topics />
         </Route>
       </Switch>
     </div>
   </Router>
 );
}

function Home() {
 return (
   <div>
```

```
      <h2>Home</h2>
    </div>
  );
}

function Topics() {
  // The `path` lets us build <Route> paths that are
  // relative to the parent route, while the `url` lets
  // us build relative links.
  let { path, url } = useRouteMatch();

  return (
    <div>
      <h2>Topics</h2>
      <ul>
        <li>
          <Link to={`${url}/rendering`}>Rendering with React</Link>
        </li>
        <li>
          <Link to={`${url}/components`}>Components</Link>
        </li>
        <li>
          <Link to={`${url}/props-v-state`}>Props v. State</Link>
        </li>
      </ul>

      <Switch>
        <Route exact path={path}>
          <h3>Please select a topic.</h3>
        </Route>
        <Route path={`${path}/:topicId`}>
          <Topic />
        </Route>
      </Switch>
    </div>
  );
}

function Topic() {
```

```
// The <Route> that rendered this component has a
// path of `/topics/:topicId`. The `:topicId` portion
// of the URL indicates a placeholder that we can
// get from `useParams()`.
let { topicId } = useParams();

return (
  <div>
    <h3>{topicId}</h3>
  </div>
);
}
```

Learn more here:
https://reactrouter.com/web/guides/quick-start

Notes

You may see lots of problems with Hook **Invalid calls** when using the router. Check this out:
https://reactjs.org/warnings/invalid-hook-call-warning.html
Try to create a new React project

Exercises:

1. Build a 3 components Car, Phone, House and use a router to switch among them (basic routing)
2. Build a component Student with params: name. When users click on a student, that component will show Hello + name (params)
3. Build a component with nesting to display different Story details when users click on a story: Cinderella, Snow White, Tam Cam (nesting)