Demo 1: Basic Routing

We've covered everything you need to know in order to create a basic router. Let's try and build one:

src/App.js

```
import {Routes, Route, Link } from "react-router-dom";
import "./styles.css";

/* Home component */
const Home = () => (
 <div>
   <h2>Home</h2>
 </div>
);

/* Category component */
const Category = () => (
 <div>
   <h2>Category</h2>
 </div>
);

/* Products component */
const Products = () => (
 <div>
   <h2>Products</h2>
 </div>
);
```

```
/* App component */

const App = () => {
 return (
   <>
     <div>
       <nav className="navbar navbar-light">
         <ul className="nav navbar-nav">
           {/*  Link components are used for linking to other views */}
           <li>
             {" "}
             <Link to="/">Homes</Link>
           </li>
           <li>
             <Link to="/category">Category</Link>
           </li>
           <li>
             <Link to="/products">Products</Link>
           </li>
         </ul>
       </nav>
       {/*  Route components are rendered if the path prop matches the current URL  *
       <Routes>
         <Route path="/" element={<Home />} />
         <Route path="/category" element={<Category />} />
         <Route path="/products" element={<Products />} />
       </Routes>
     </div>
   </>
 );
};
export default App;
```

Demo 1: Basic Routing

We've declared the components for Home, Category and Products inside App.js. This is fine enough for now, but when the components start to grow bigger, it's better to have a separate file for each component. As a rule of thumb, we usually create a new file for a component if it occupies more than 10 lines of code. (Starting from the second demo, we'll be creating a separate file for components that have grown too big to fit inside the App.js file).

Inside the App component, we've written the logic for routing. The <Route>'s path is matched with the current location and a component gets rendered. The component that should be rendered is passed in as the `element` prop.

Here `/` matches both `/` and `/category`. In previous version of react-router-dom, we have to pass a prop called `exact` to avoid that both component get rendered.

```
<Route exact  path="/" … />
```

In the **current** version of **react-router-dom (V6),** we don't have to that anymore. All path are by default exact.

Previous next