

Here's a basic example to introduce React Router in a React application:

First, you'll need to install React Router in your project. You can do this using npm or yarn:

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

or

```
```bash
yarn add react-router-dom
```
```

Once React Router is installed, you can create a simple React application with routing:

```
```jsx
// App.js
import React from 'react';
import { BrowserRouter as Router, Route, Link } from 'react-router-dom';

// Components
const Home = () => <h2>Home</h2>;
const About = () => <h2>About</h2>;
const Contact = () => <h2>Contact</h2>;

function App() {
 return (
 <Router>
 <div>
 <nav>

 <Link to="/">Home</Link>

 <Link to="/about">About</Link>

 <Link to="/contact">Contact</Link>

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

```

 <Route path="/" exact component={Home} />
 <Route path="/about" component={About} />
 <Route path="/contact" component={Contact} />
 </div>
</Router>
);
}

export default App;
```

```

In this example:

- We import necessary components from `react-router-dom`, including `BrowserRouter`, `Route`, and `Link`.
- We define three simple functional components: `Home`, `About`, and `Contact`, each rendering a basic `

` element.
- Inside the `App` component, we wrap our application in a `` component from React Router.
- We create a simple navigation bar using `` components. These links will navigate to the specified paths without causing a full page reload.
- We use `` components to define which component should render for each specific path. The `exact` attribute is used to ensure that only the exact path matches for the home route.

With this setup, you can navigate between different pages of your React application using React Router.

Example2:

Sure, here's another example demonstrating nested routing with React Router:

```

```jsx
// App.js
import React from 'react';
import { BrowserRouter as Router, Route, Link } from 'react-router-dom';

// Components
const Home = () => <h2>Home</h2>;
const About = () => <h2>About</h2>;
const Topics = ({ match }) => (
 <div>
 <h2>Topics</h2>


```

```


 <Link to={` ${match.url}/react`} >React</Link>

 <Link to={` ${match.url}/redux`} >Redux</Link>

 <Link to={` ${match.url}/router`} >React Router</Link>

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

const Topic = ({ match }) => (
 <div>
 <h3>{match.params.topicId}</h3>
 </div>
);

function App() {
 return (
 <Router>
 <div>
 <nav>

 <Link to="/" >Home</Link>

 <Link to="/about" >About</Link>

 <Link to="/topics" >Topics</Link>

 </nav>

 <Route path="/" exact component={Home} />
 <Route path="/about" component={About} />
 <Route path="/topics" component={Topics} />
 </div>
 </Router>
);
}

```

```

 </Router>
);
}

export default App;
```

```

In this example:

- We have a main `App` component that renders a navigation bar and uses React Router's `BrowserRouter` to manage routing.
- We define three basic components: `Home`, `About`, and `Topics`.
- The `Topics` component contains nested routing. It renders a list of topics with links to individual topic pages.
- Each topic link includes a dynamic parameter (`topicId`) in the URL.
- The `Topic` component receives the `topicId` from the URL parameters and displays it.
- When the user navigates to `/topics`, the `Topics` component will be rendered along with its child routes (`Topic` and a default message when no topic is selected).
- The `Route` components inside the `Topics` component use the `match.path` and `match.url` properties to create nested routes relative to the parent route.

With this setup, you can navigate between different pages and nested routes within your React application using React Router.

Example:

Certainly! Here's an example that demonstrates navigation between components using React Router:

```

```jsx
// App.js
import React from 'react';
import { BrowserRouter as Router, Route, Link } from 'react-router-dom';
import Home from './components/Home';
import About from './components/About';
import Contact from './components/Contact';

function App() {
 return (
 <Router>
 <div>
 <nav>


```

```

 <Link to="/">Home</Link>

 <Link to="/about">About</Link>

 <Link to="/contact">Contact</Link>

 </nav>

 <Route path="/" exact component={Home} />
 <Route path="/about" component={About} />
 <Route path="/contact" component={Contact} />
</div>
</Router>
);
}

export default App;

```

In this example:

- We import `BrowserRouter`, `Route`, and `Link` from `react-router-dom`.
- We define three components: `Home`, `About`, and `Contact`.
- Inside the `App` component, we wrap our application in a `` component from React Router.
- We create a simple navigation bar using `` components. These links will navigate to the specified paths without causing a full page reload.
- We use `` components to define which component should render for each specific path. The `exact` attribute is used to ensure that only the exact path matches for the home route.

Now, when a user clicks on a link in the navigation bar, the corresponding component will be rendered in the main content area without a full page reload. This allows for seamless navigation between different components within the React application.