Estructura de Datos | |

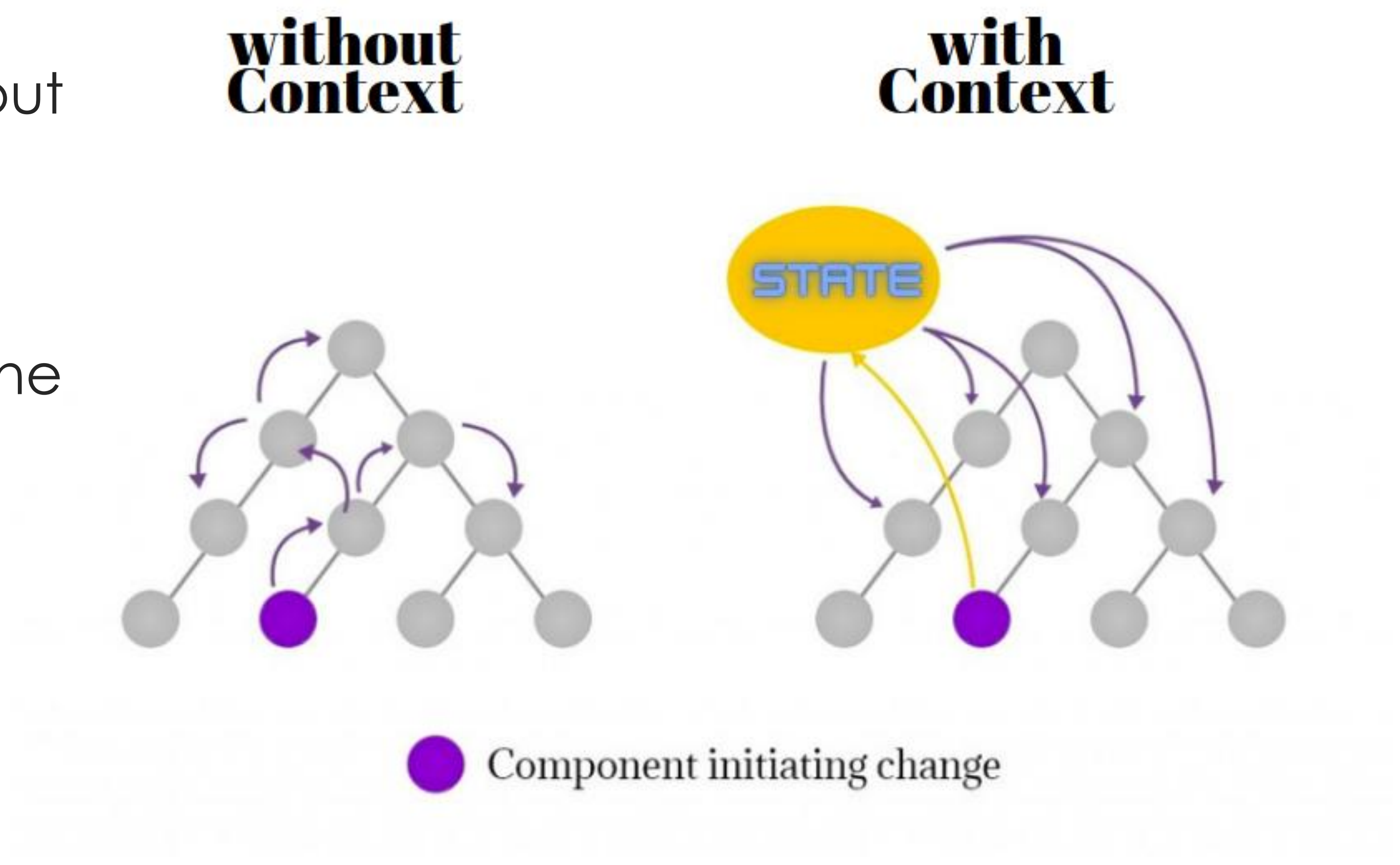
Jonathan López Londoño
jlopezl@uao.edu.co
315 926 5443

React JS

Context API

It is a way to share global state between multiple components without **prop drilling**.

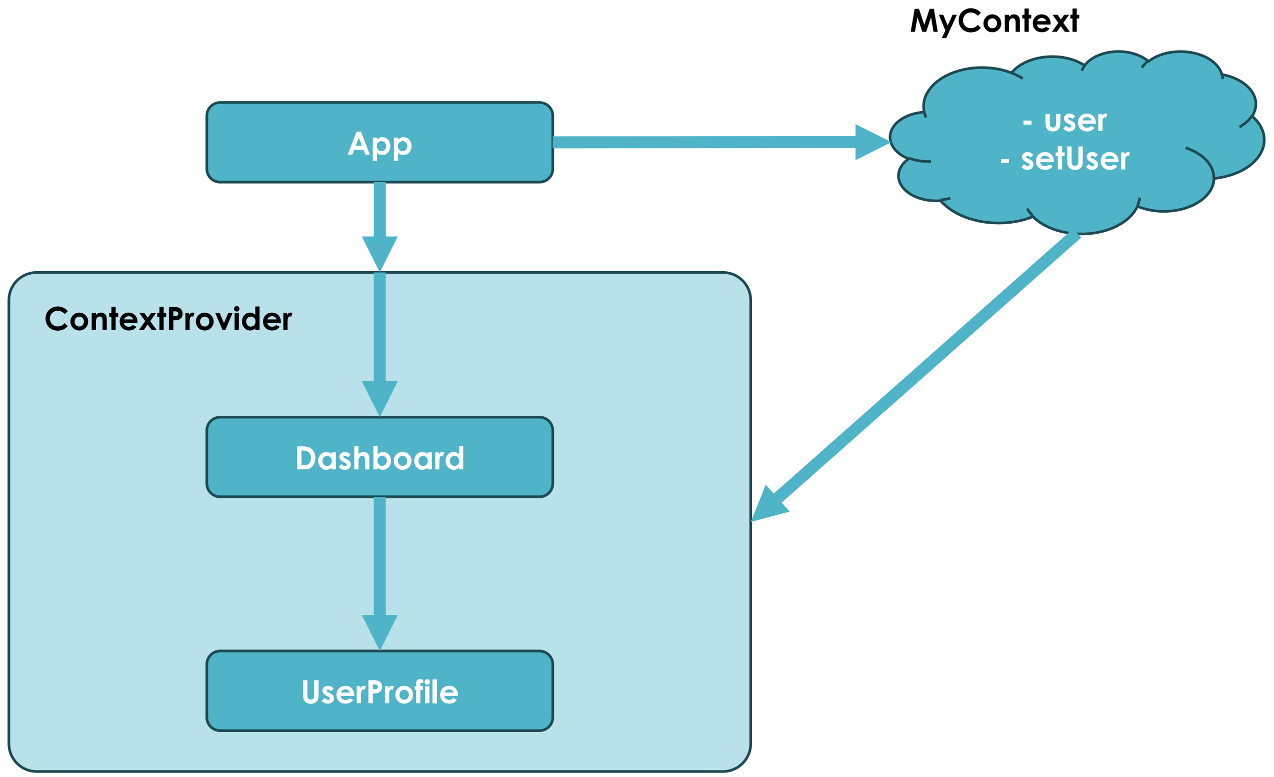
Is required to use one context and one provider



Create Context API

To create a context:

1. Use <u>*CreateContext*</u> Function to create a new ***ContextName***
2. Use ***ContextName***.Provider as a Component and send data to be shared into the value prop.
3. Add Children who must be covered by the context



Create Context API

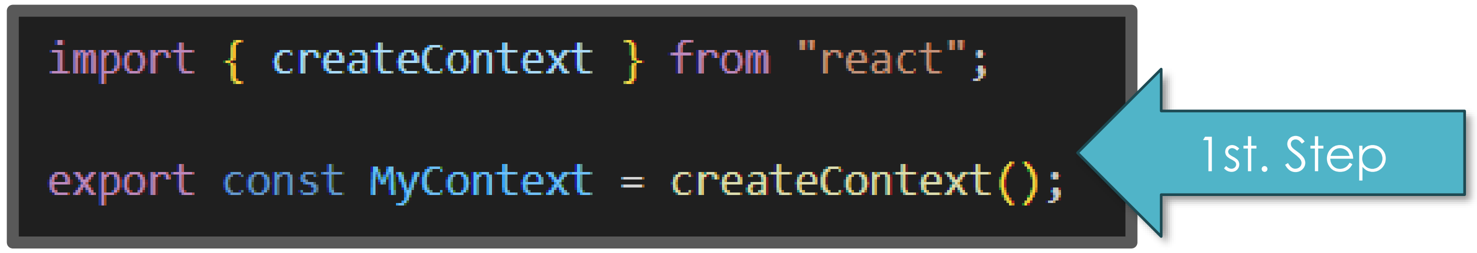
To create a context:

1. Use CreateContext function to create a new ***ContextName***

Example:

Mycontext

/Context.tsx



```
import { createContext } from "react";

export const MyContext = createContext();
```

Create Context API

To create a context:

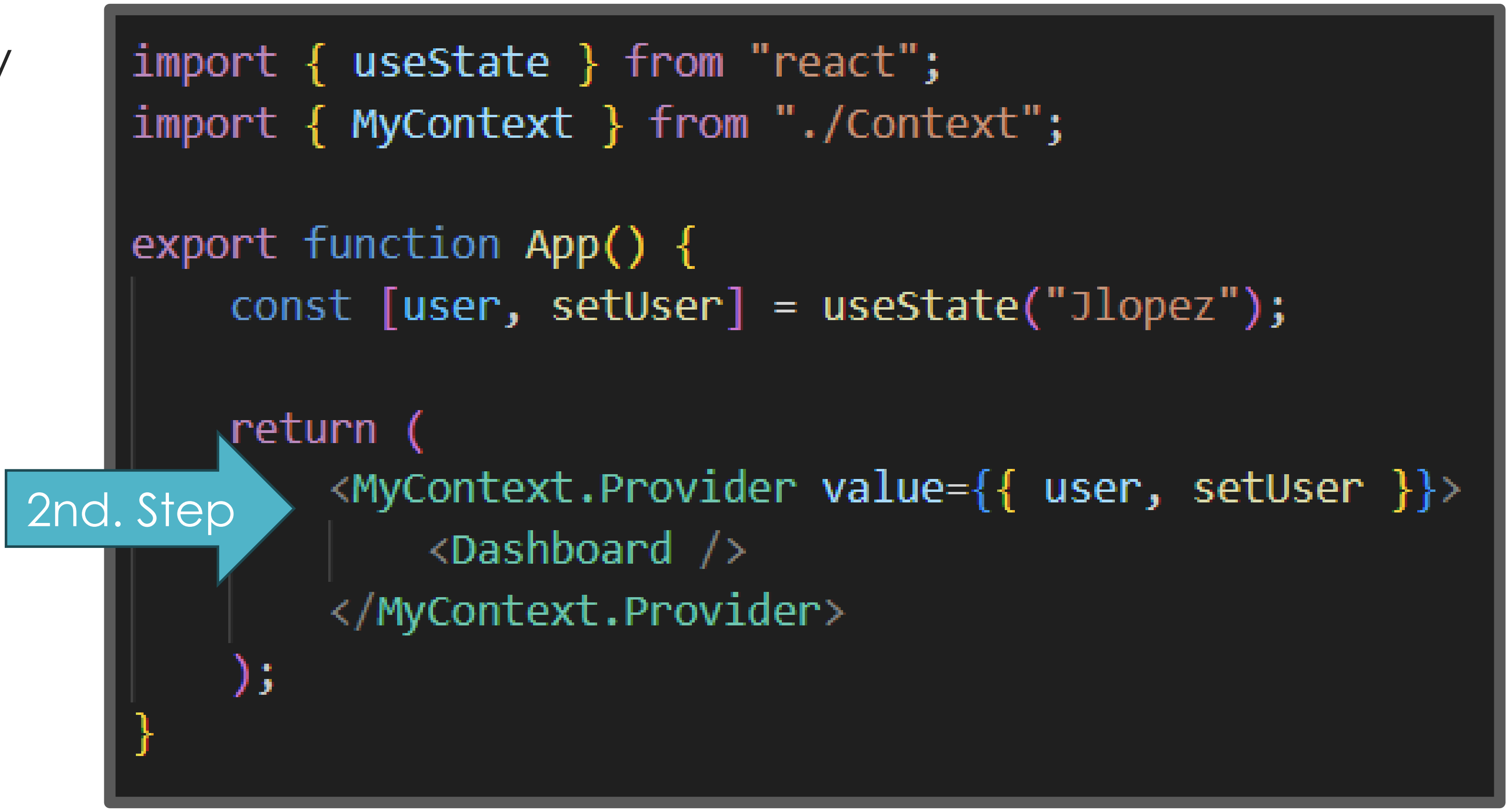
1. Use _CreateContext_ Function to create a new ***ContextName***
2. Use ***ContextName***.Provider as a Component and send data to be shared into the ***value*** prop as an object.

Example:

To share user and setUser data, let's call:

<MyContext.Provider value={{ user, setUser }}>

/App.tsx



```
import { useState } from "react";
import { MyContext } from "./Context";

export function App() {
    const [user, setUser] = useState("Jlopez");

    return (
        <MyContext.Provider value={{ user, setUser }}>
            <Dashboard />
        </MyContext.Provider>
    );
}
```

Create Context API

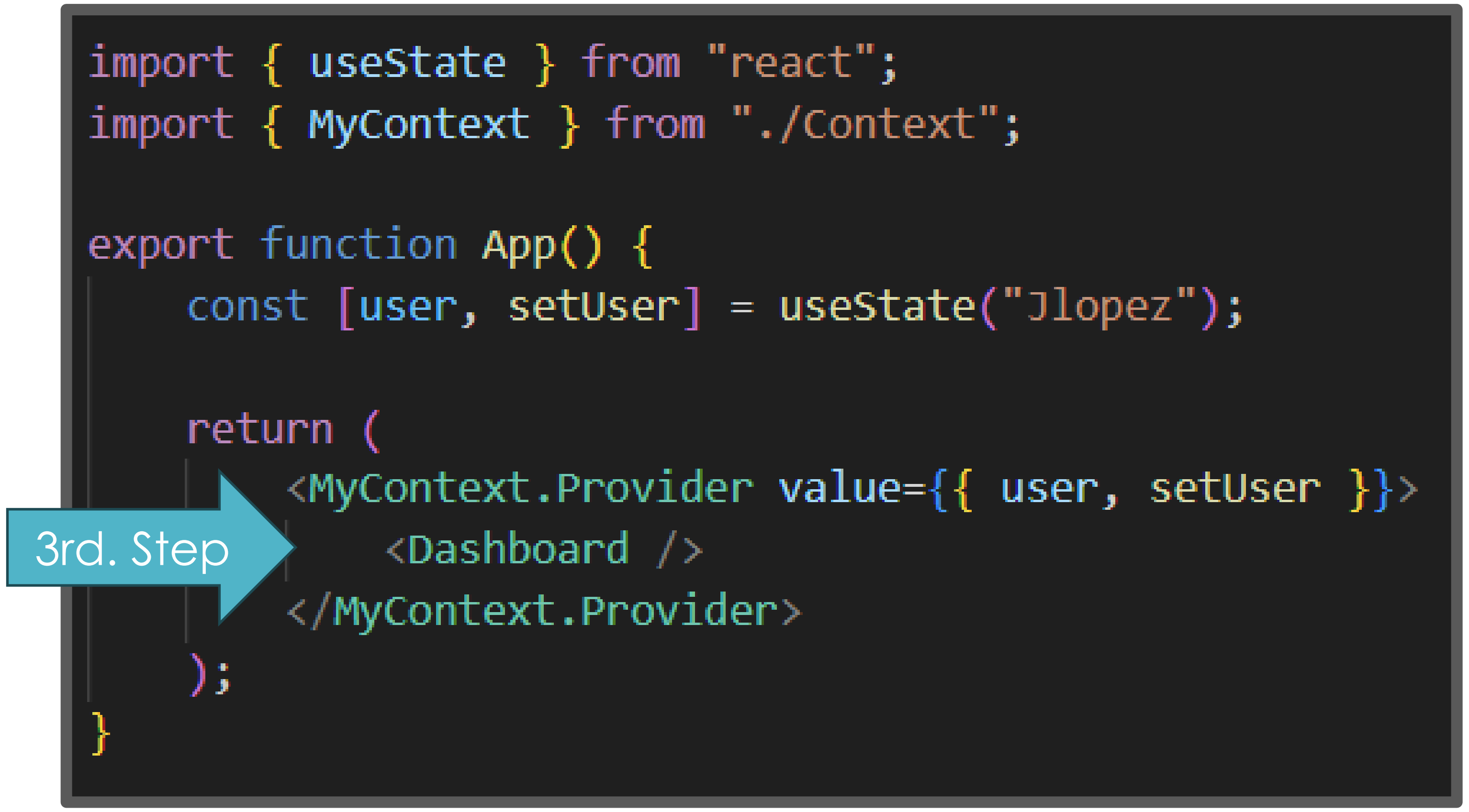
To create a context:

1. Use <u>*CreateContext*</u> Function to create a new ***ContextName***
2. Use ***ContextName***.Provider as a Component and send data to be shared into the value prop.
3. Add Children who must be covered by the context:

Example

To Cover Dashboard and its children, put it as a child of Provider

/App.tsx



```
import { useState } from "react";
import { MyContext } from "./Context";

export function App() {
    const [user, setUser] = useState("Jlopez");

    return (
        <MyContext.Provider value={{ user, setUser }}>
            <Dashboard />
        </MyContext.Provider>
    );
}
```

Create Context API

Hooks

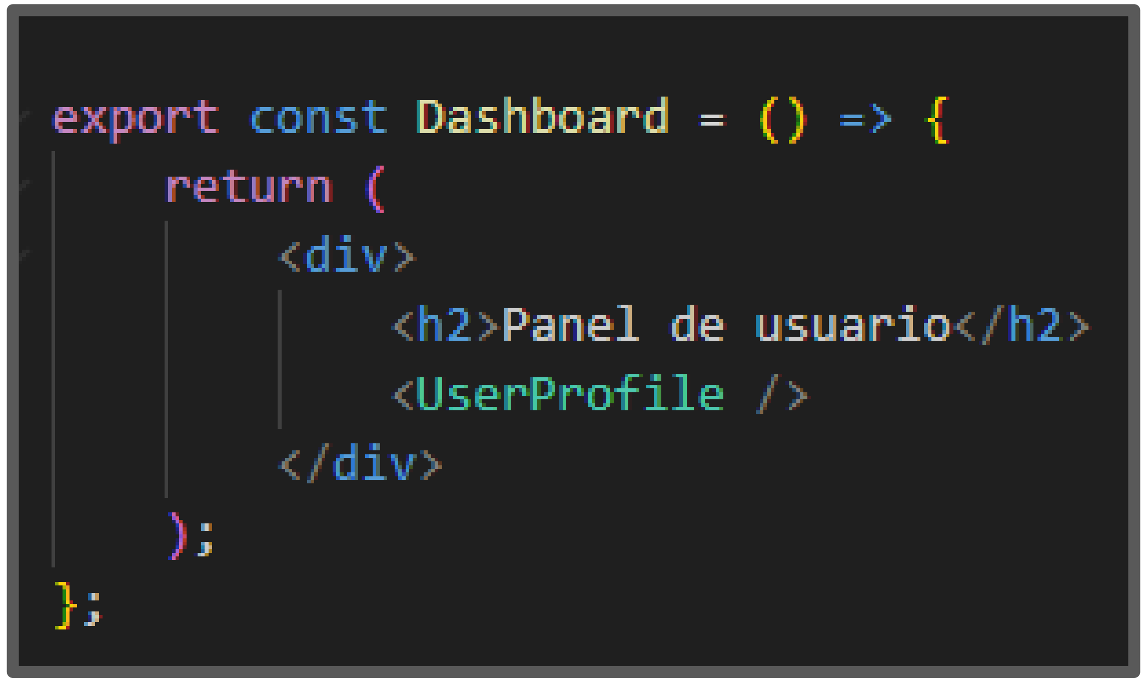
useContext: it's used to invoke data from previous created context.

Example

To invoke **user** from **MyContext:**

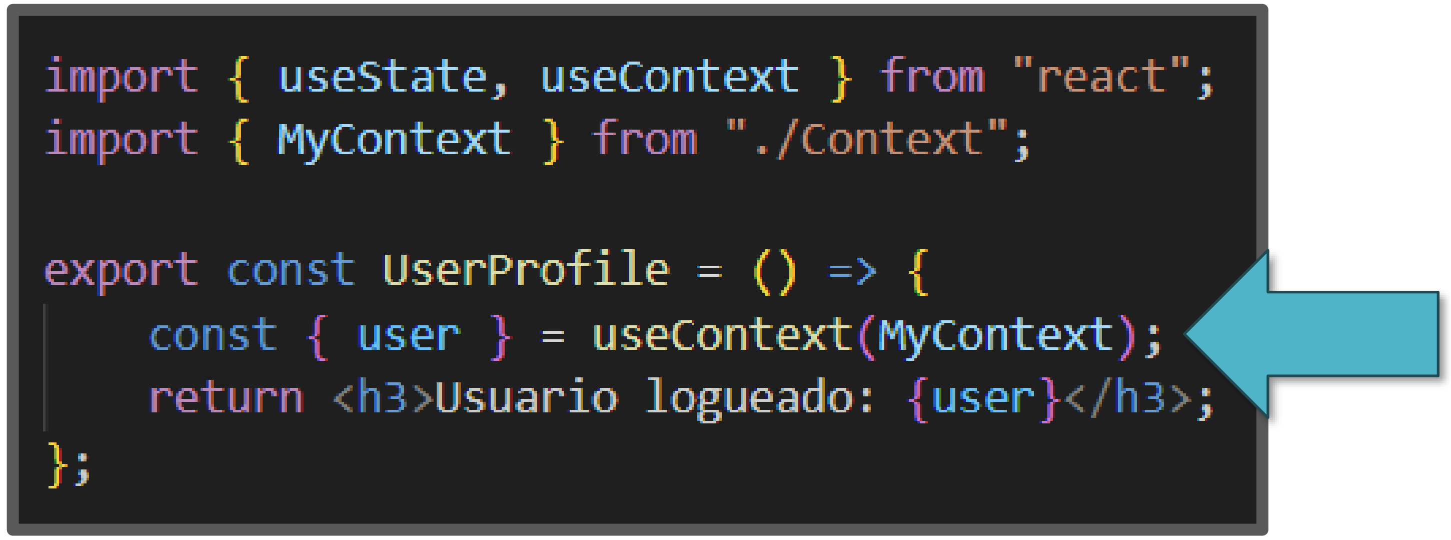
const { **user** } = **useContext**(**MyContext**);

/Dashboard.tsx



```
export const Dashboard = () => {
    return (
        <div>
            <h2>Panel de usuario</h2>
            <UserProfile />
        </div>
    );
};
```

/UserProfile.tsx



```
import { useState, useContext } from "react";
import { MyContext } from "./Context";

export const UserProfile = () => {
    const { user } = useContext(MyContext);
    return <h3>Usuario logueado: {user}</h3>;
};
```

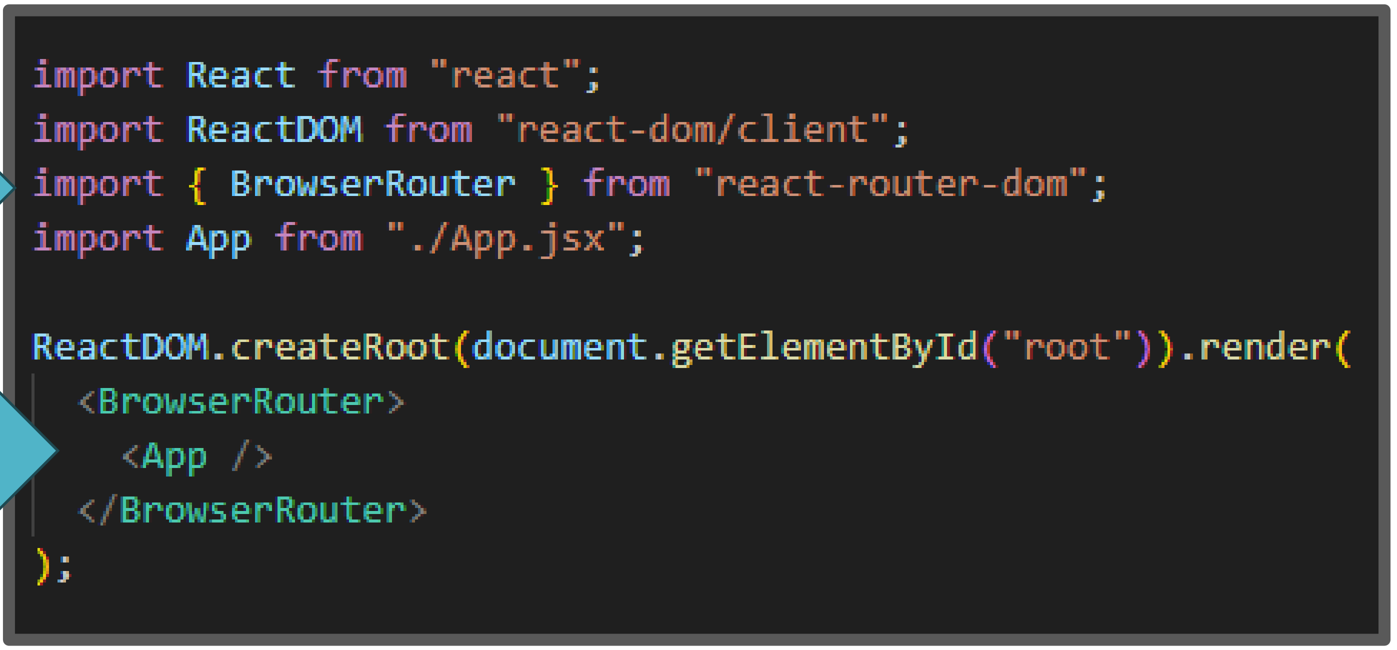
REACT ROUTER

It's a library that allows you to manage navigation in React applications in a declarative way, creating routes without the need to reload the page (SPA - Single Page Application).

npm install react-router-dom

To use React Router, must involve the App into the **BrowserRouter** Parent

https://reactrouter.com/en/main



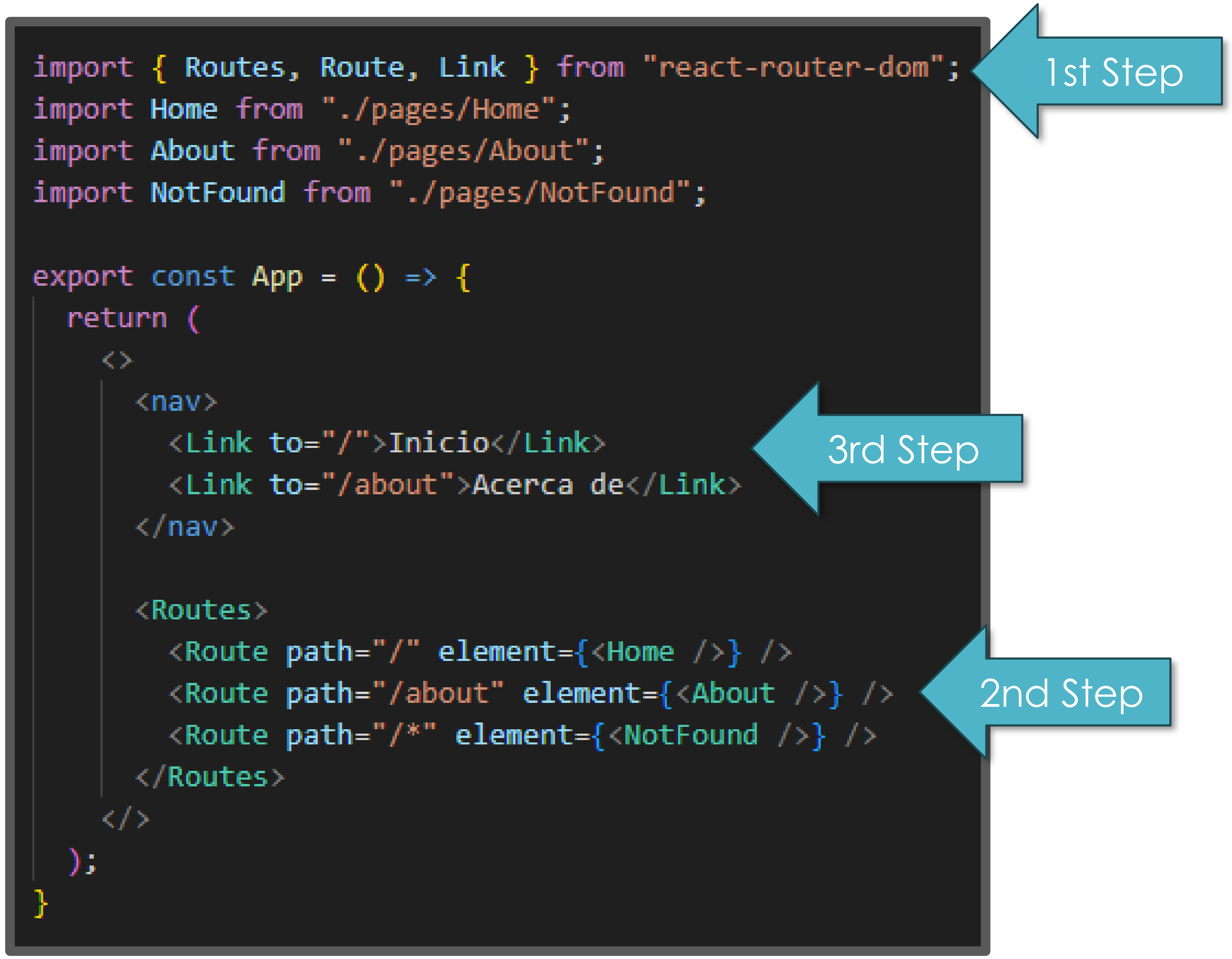
```
import React from "react";
import ReactDOM from "react-dom/client";
import { BrowserRouter } from "react-router-dom";
import App from "./App.jsx";

ReactDOM.createRoot(document.getElementById("root")).render(
  <BrowserRouter>
    <App />
  </BrowserRouter>
);
```

CREATING ROUTES

Now we can:

1. Import **Routes**, **Route** and **Link** from the library
2. Create the routes inside the App.jsx component using **Routes** and **Route.**
3. Create links using **Link** to navigate to the route



```jsx
import { Routes, Route, Link } from "react-router-dom";
import Home from "./pages/Home";
import About from "./pages/About";
import NotFound from "./pages/NotFound";

export const App = () => {
  return (
    <>
      <nav>
        <Link to="/">Inicio</Link>
        <Link to="/about">Acerca de</Link>
      </nav>

      <Routes>
        <Route path="/" element={<Home />} />
        <Route path="/about" element={<About />} />
        <Route path="/*" element={<NotFound />} />
      </Routes>
    </>
  );
}
```

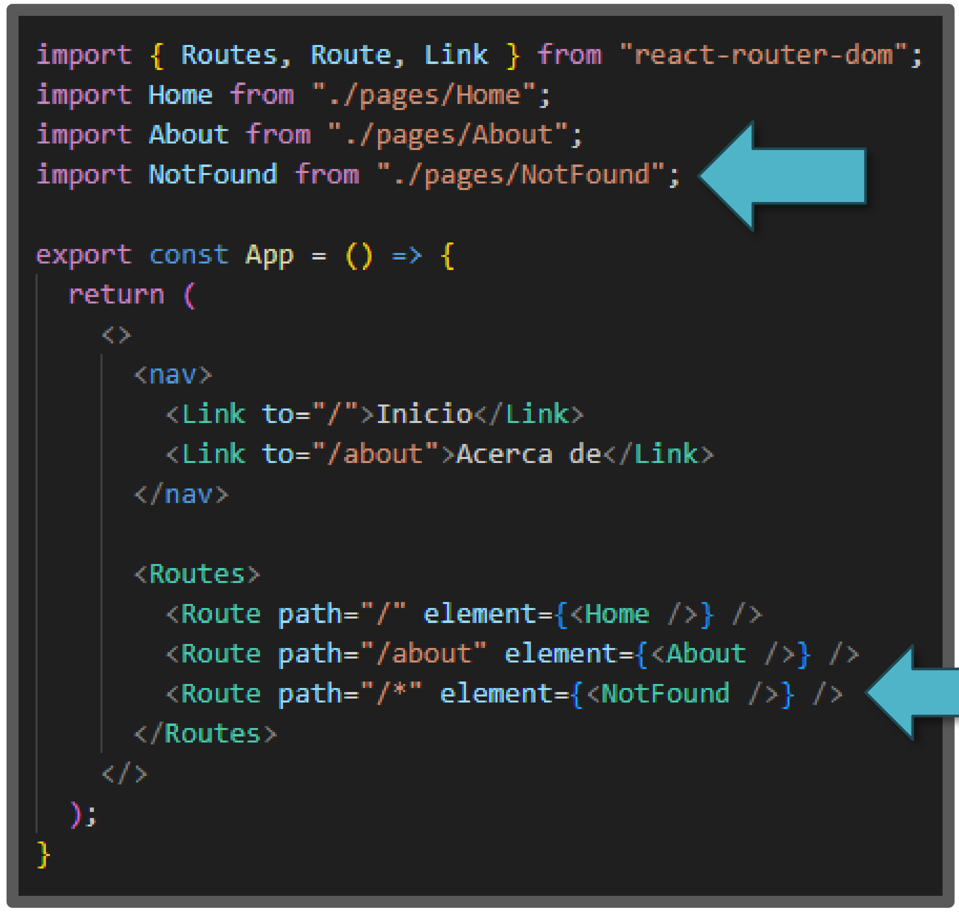
Handling 404 page

To handle 404 pages:

1. You can create a new **/*** route path and assign it to one component OR

2. Create a new **/*** route path and enable ' ***Navigate to*** ' to redirect to some other route.

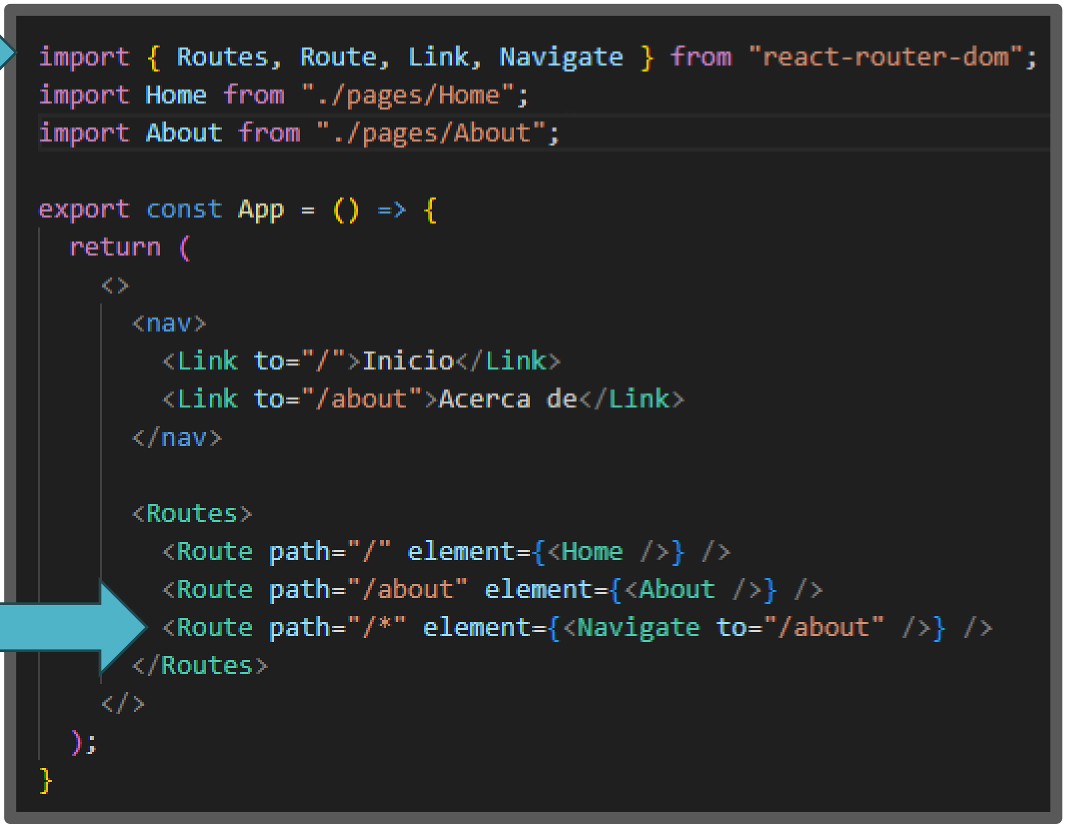
Using a Component



```
import { Routes, Route, Link } from "react-router-dom";
import Home from "./pages/Home";
import About from "./pages/About";
import NotFound from "./pages/NotFound";

export const App = () => {
  return (
    <>
      <nav>
        <Link to="/">Inicio</Link>
        <Link to="/about">Acerca de</Link>
      </nav>

      <Routes>
        <Route path="/" element={<Home />} />
        <Route path="/about" element={<About />} />
        <Route path="/*" element={<NotFound />} />
      </Routes>
    </>
  );
}
```



```
import { Routes, Route, Link, Navigate } from "react-router-dom";
import Home from "./pages/Home";
import About from "./pages/About";

export const App = () => {
  return (
    <>
      <nav>
        <Link to="/">Inicio</Link>
        <Link to="/about">Acerca de</Link>
      </nav>

      <Routes>
        <Route path="/" element={<Home />} />
        <Route path="/about" element={<About />} />
        <Route path="/*" element={<Navigate to="/about" />} />
      </Routes>
    </>
  );
}
```

Using a Route to Redirect

URL Segments and Query Params

Query Params:

http:// ….com/products**?product=10**

http:// ….com/stores**?store=23&name=apple**

http:// ….com / users**?user=14&role=3&month=march**

URL Segments

http:// ….com/products**/10**

http:// ….com/store**/23**

http:// ….com/user**/14**

URL Params

useSearchParams function is used to get and set params from URL:

Example:

http:// ….com/?**role=seller**

const [params, setParams] = useSearchParams()

To get:

*params.get(**role**)*

To set:

*setParams**({role: 'admin'})***

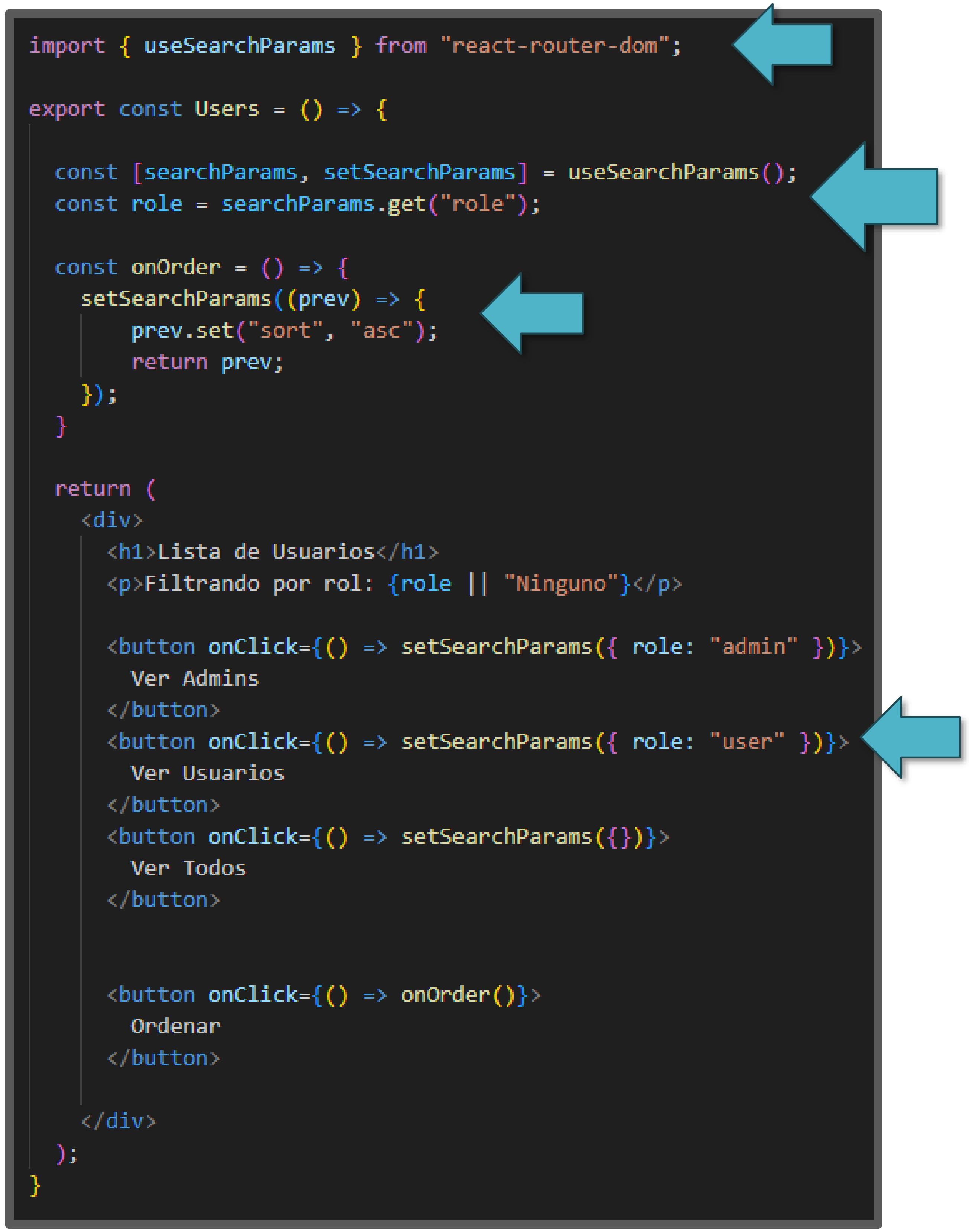
To add a new Param:

setParams((prev) => {

prev.set("sort", "asc");

return prev;

});



```
import { useSearchParams } from "react-router-dom";

export const Users = () => {

  const [searchParams, setSearchParams] = useSearchParams();
  const role = searchParams.get("role");

  const onOrder = () => {
    setSearchParams((prev) => {
      prev.set("sort", "asc");
      return prev;
    });
  }

  return (
    <div>
      <h1>Lista de Usuarios</h1>
      <p>Filtrando por rol: {role || "Ninguno"}</p>

      <button onClick={() => setSearchParams({ role: "admin" })}>
        Ver Admins
      </button>
      <button onClick={() => setSearchParams({ role: "user" })}>
        Ver Usuarios
      </button>
      <button onClick={() => setSearchParams({})}>
        Ver Todos
      </button>


      <button onClick={() => onOrder()}>
        Ordenar
      </button>

    </div>
  );
}
```

URL Segments

useParams function is used to get params from segment:

Example:

http:// ….com/users/**10**

http:// ….com/users/**20**

http:// ….com/users/**30**

1. Must create a new Child Route:

Example

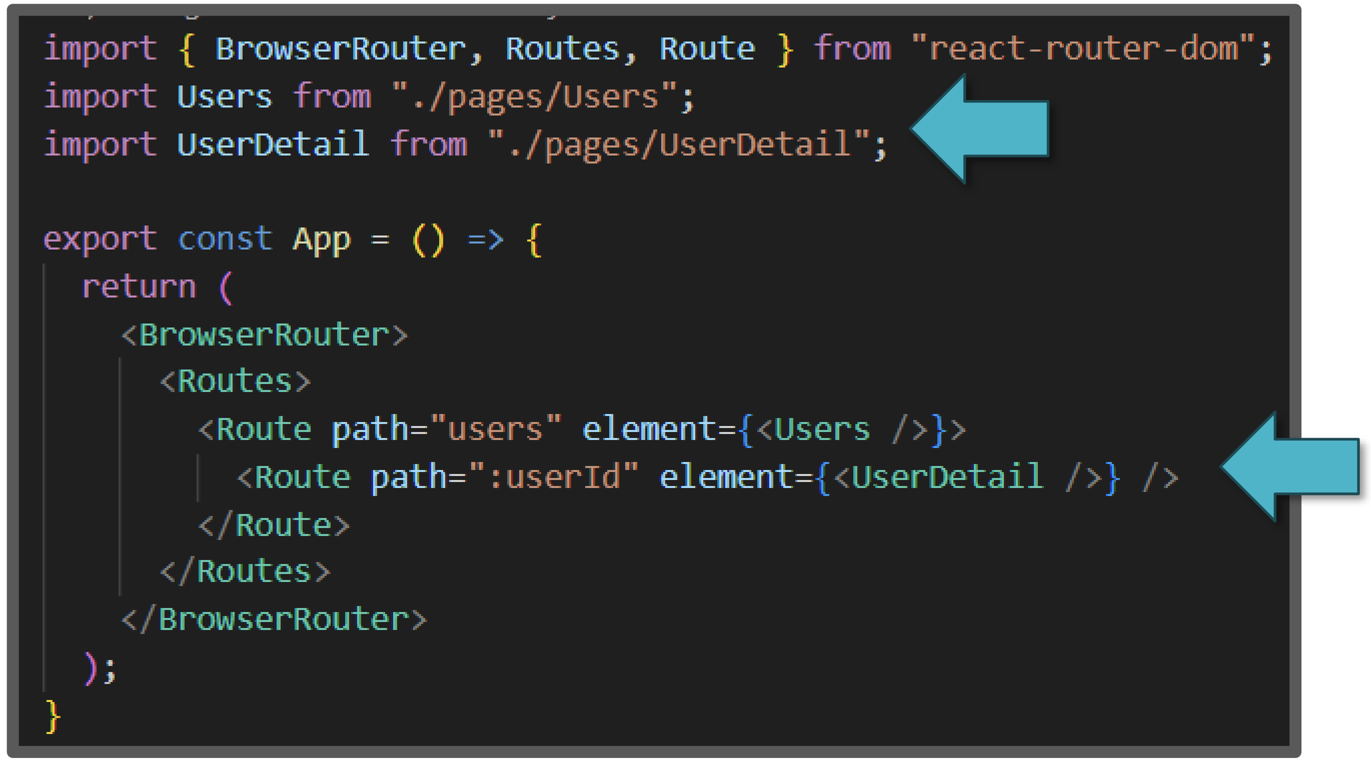
<Route path="**users**" element={<**Users** />}>

    <Route path="**:userId**" element={<**UserDetail** />} />

</Route>

userId is a dynamic parameter that changes depending on the URL.

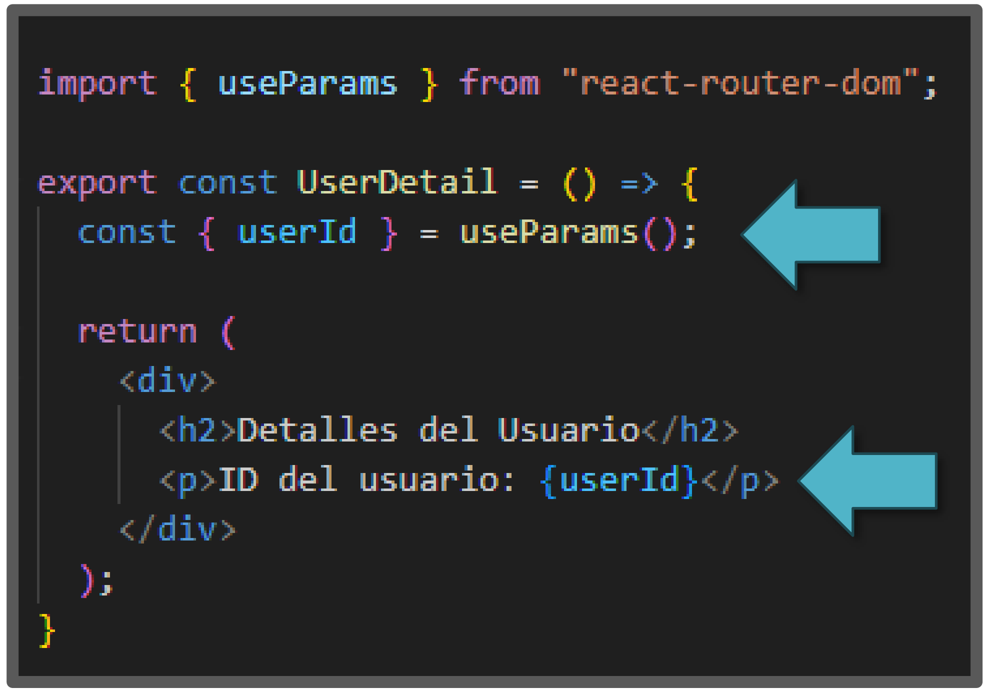
/App.tsx



```
import { BrowserRouter, Routes, Route } from "react-router-dom";
import Users from "./pages/Users";
import UserDetail from "./pages/UserDetail";

export const App = () => {
  return (
    <BrowserRouter>
      <Routes>
        <Route path="users" element={<Users />}>
          <Route path=":userId" element={<UserDetail />} />
        </Route>
      </Routes>
    </BrowserRouter>
  );
}
```

/pages/UserDetail.tsx



```
import { useParams } from "react-router-dom";

export const UserDetail = () => {
  const { userId } = useParams();

  return (
    <div>
      <h2>Detalles del Usuario</h2>
      <p>ID del usuario: {userId}</p>
    </div>
  );
}
```

URL Redirections

useNavigate() is used to redirect from current page to a new one through the code.

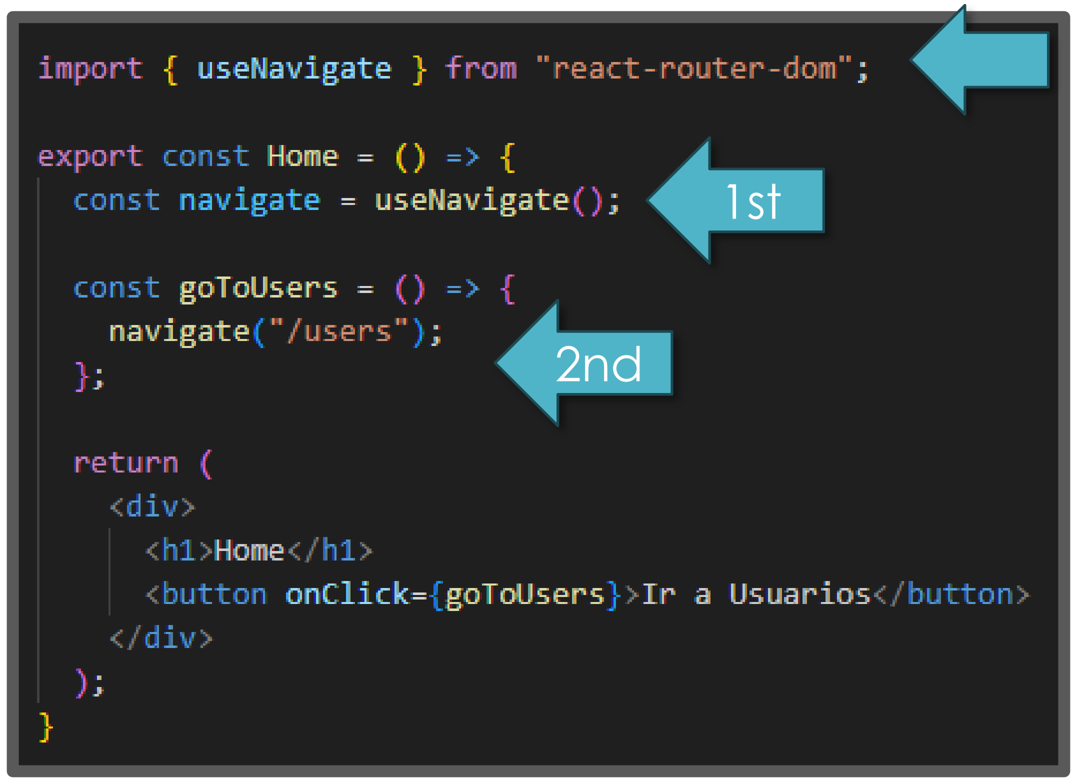
1. Invoke **useNavigate()** into a variable
2. Call variable as a function with the route to be redirected

navigate("/")

3. Also, can forget the current page to avoid going back from the navigator back button.

Example:

The user cannot go back to the login page using the browser's "Back" button.

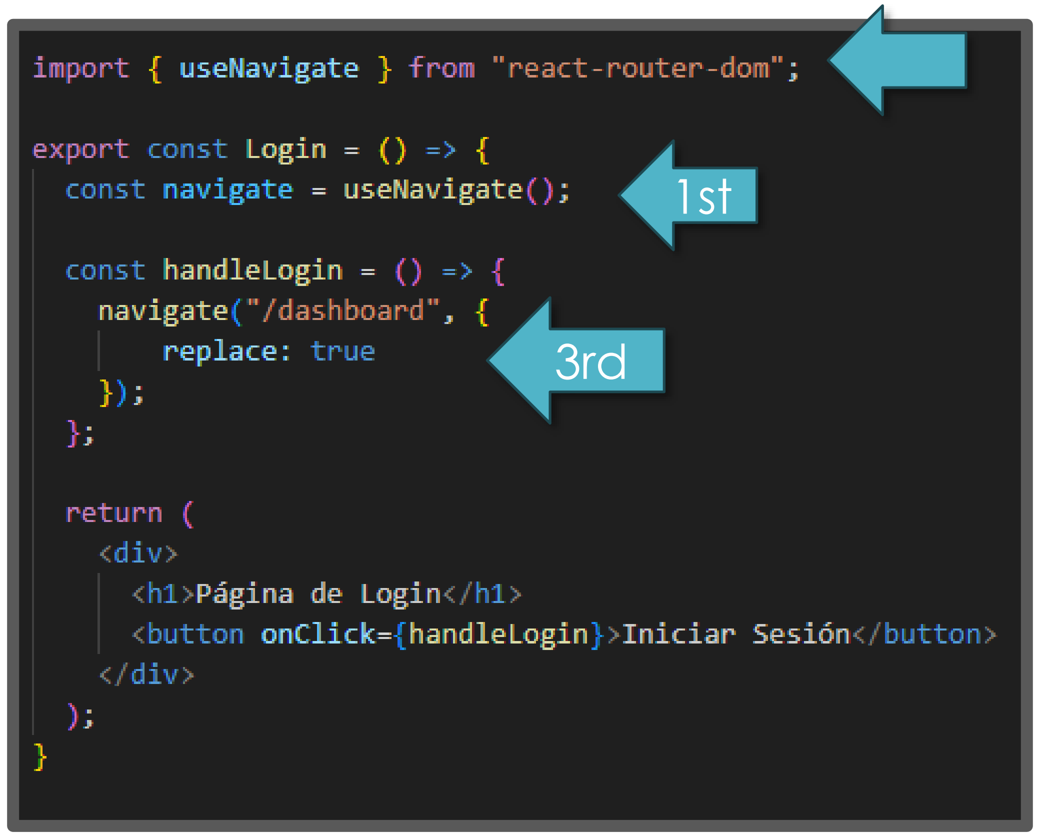


```
import { useNavigate } from "react-router-dom";

export const Home = () => {
  const navigate = useNavigate();

  const goToUsers = () => {
    navigate("/users");
  };

  return (
    <div>
      <h1>Home</h1>
      <button onClick={goToUsers}>Ir a Usuarios</button>
    </div>
  );
}
```



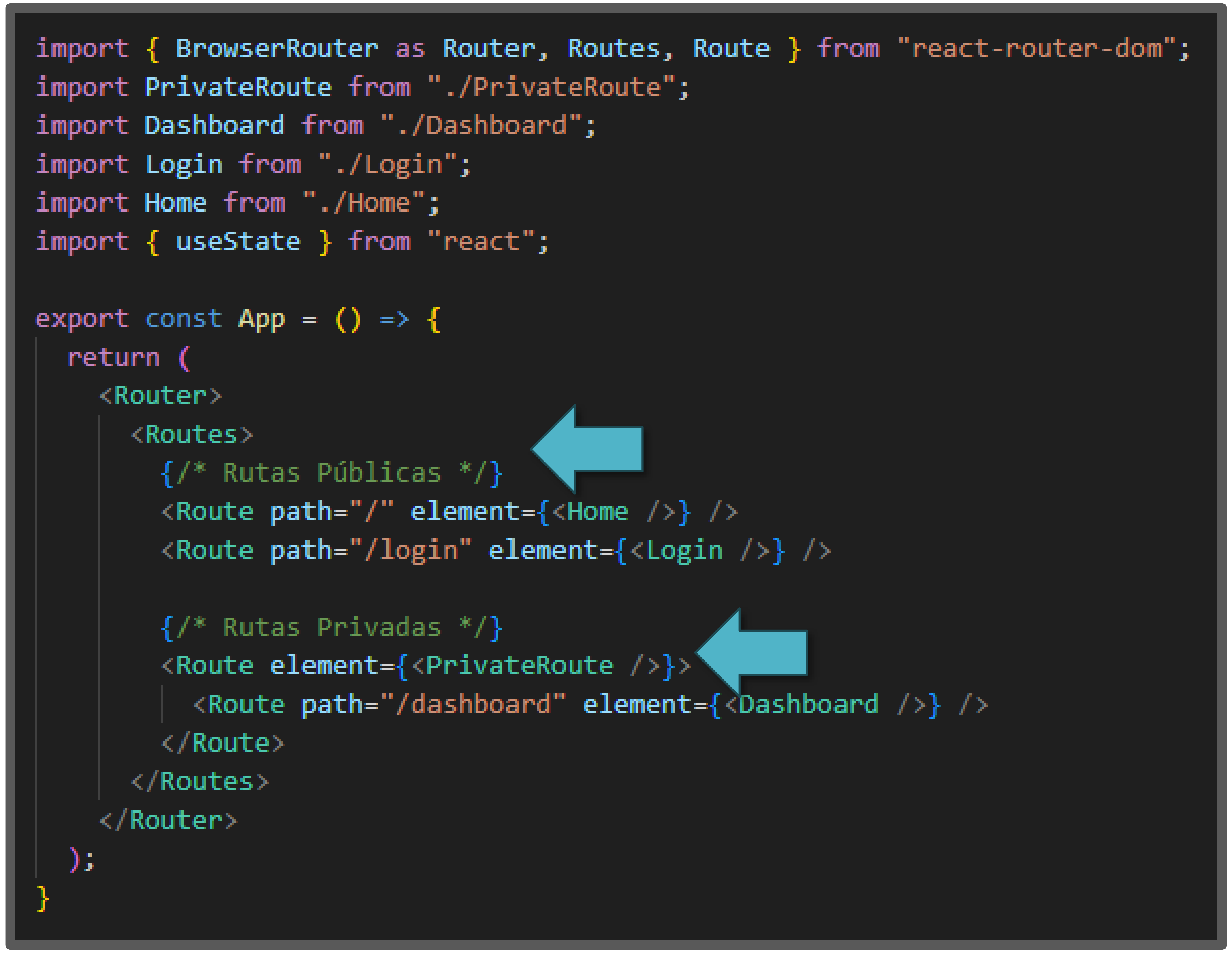
```
import { useNavigate } from "react-router-dom";

export const Login = () => {
  const navigate = useNavigate();

  const handleLogin = () => {
    navigate("/dashboard", {
      replace: true
    });
  };

  return (
    <div>
      <h1>Página de Login</h1>
      <button onClick={handleLogin}>Iniciar Sesión</button>
    </div>
  );
}
```

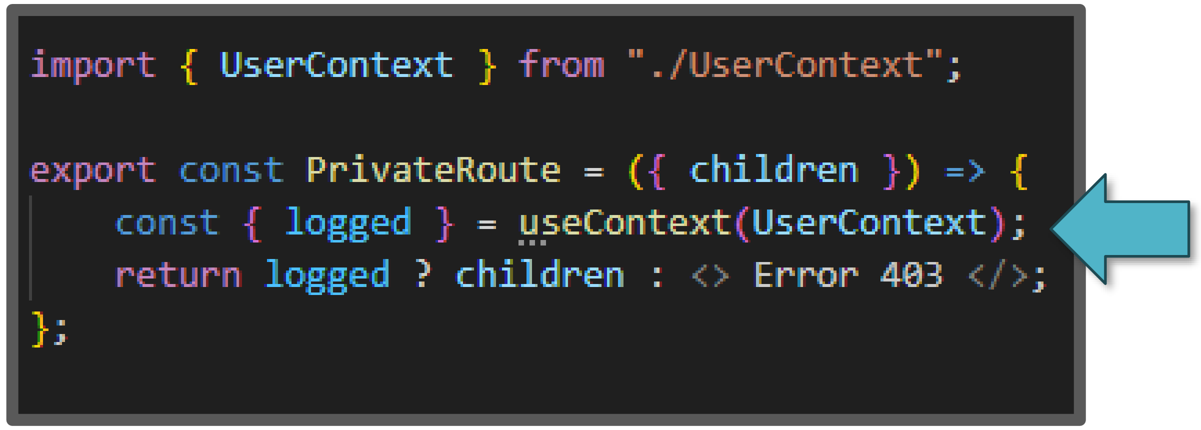
Private and public routes



```
import { BrowserRouter as Router, Routes, Route } from "react-router-dom";
import PrivateRoute from "./PrivateRoute";
import Dashboard from "./Dashboard";
import Login from "./Login";
import Home from "./Home";
import { useState } from "react";

export const App = () => {
  return (
    <Router>
      <Routes>
        {/* Rutas Públicas */}
        <Route path="/" element={<Home />} />
        <Route path="/login" element={<Login />} />

        {/* Rutas Privadas */}
        <Route element={<PrivateRoute />}>
          <Route path="/dashboard" element={<Dashboard />} />
        </Route>
      </Routes>
    </Router>
  );
}
```



```
import { UserContext } from "./UserContext";

export const PrivateRoute = ({ children }) => {
    const { logged } = useContext(UserContext);
    return logged ? children : <> Error 403 </>;
};
```

CHALLENGE 06

1. Implement FAKE Login and Logout in a new project. Keep in mind to use Context, Providers and States
2. Enable some public and private routes in your current project.
3. Test your private routes by login and logout
4. Show current username when the user is logged in.