React4teachers

2022 edition

1. React4teachers: From Zero 2 Hero

2. Contenido del curso

- 1.- Introducció
- 2.- De VanillaJs a React
- 3.- React framework: Motivos de exito
- 4.- React framework: Class components vs Function components
- 5.- React framework: State management - Control de el estado de la aplication
- 6.- React framework: Routing
- 7.- Storybook
- 8.- Monorepo
- 9.- TailwindCSS y StyledComponents
- 10.-Despliegue de aplicaciones

3. React router + Helmet

Hasta ahora hemos usado React como SPA, pero eso tiene serios inconvenientes:

- La URL del navegador permanece en el mismo sitio

- No podemos enlazar externamente
- SEO se ve seriamente penalizado

Utilizaremos Router para enlazar y Helmet para inyectar atributos. Su uso es progresivo.

4. Objetivo / Ejercicio de la UD

En este caso, sin dar la solucion, deberas buscar como convertir la app que estamos haciendo con react router y sin dejar de utilizar las siguientes tecnicas:

- Componentes
- Uso de props
- Application State (Redux)
- Links (Router)
- SEO (Helmet)

5. React Router

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

https://reactrouter.com/

6. How to Route

La configuracion de React Router es muy sencilla

```
# create react app
npx create-react-app router-tutorial
npm install react-router-dom@6
```

Usaremos los siguientes elementos:

- BrowserRouter para englobar nuestra App
- Routes para "configurar" los enlaces
- Route que define el path y el componente

7. React router

```
import { render } from "react-dom";
import {
  BrowserRouter,
  Routes,
  Route,
} from "react-router-dom";
import App from "./App";
import ComponentA from "./pages/componentA";
import ComponentB from "./pages/componentB";

const rootElement = document.getElementById("root");
render(
  <BrowserRouter>
    <Routes>
      <Route path="/" element={<App />} />
      <Route path="/example" element={<ComponentA />} />
      <Route path="/other" element={<ComponentB />} />
    </Routes>
  </BrowserRouter>,
  rootElement
);
```

8. React router

El problema de esa aproximacion es el siguiente:

- Los componentes son independientes
- La comunicacion es limitada entre ellos
- Nos falta el concepto de layout
- Nos falta el envio de parametros por get

Para aplicar un layout tan solo tienes que incluir La Route de App como parent, e incluir Outlet y Link en App.

Veamos como se hace!

9. Lab 3 - p1:

En tu fichero index.js, engloba tu App component con BrowserRouter:

```
import React from 'react';
import ReactDOM from 'react-dom';
import './index.css';
import App from './App';
import reportWebVitals from './reportWebVitals';
import { BrowserRouter } from 'react-router-dom';
ReactDOM.render(
  <React.StrictMode>
      <BrowserRouter>
          <App />
      </BrowserRouter>
  </React.StrictMode>,
  document.getElementById('root')
);
```

10. Lab3 - p2:

Generemos los siguientes componentes:

- pages/Layout: Incluira la estructura de la pagina
- components/Menu: Incluira las acciones
- pages/Users: Componente listado
- pages/User: Componente detail, aceptara props

Para poder enlazar a las diferentes paginas, deberemos modificar index.js con lo siguiente:

```
ReactDOM.render(
  <React.StrictMode>
      <BrowserRouter>
          <Routes>
              <Route path="/" element={<App />}>
                  <Route path="users" element={<Users />} />
                  <Route path="user" element={<User />} />
              </Route>
          </Routes>
      </BrowserRouter>
  </React.StrictMode>,
  document.getElementById('root')
);
```

Conviene incluir las routas en un fichero separado

11. Lab3 - p3: App.js

Usaremos app.js como el layout principal y que engloba todo Para ello usaremos el component Outlet

```
import { Outlet} from "react-router-dom";
import './App.css';
import Menu from "./components/Menu";
function App() {
  return (
    <div className="App">
      <header className="App-header">
        <Menu/>
      </header>
      <Outlet />
    </div>
  );
}
export default App;
```

12. Lab3 - p4: components/Menu

Genera un componente Menu que incluya los enlaces que necesitas.

```
import { Outlet, Link } from "react-router-dom";

const MenuStyle = {
    display: 'flex',
    justifyContent: 'space-around',
    width: "100%",
    borderBottom: "solid 1px",
    paddingBottom: "1rem"
}

const Menu = () => (
    <nav style={MenuStyle}>
        <div>Aloha!</div>
        <Link to="/">Main</Link>
        <Link to="/users">Users</Link>
        <Link to="/user/1">User 1</Link>
    </nav>
```

```
)

export default Menu
```

13. Lab3 - p5: pages/Users

Genera un componente sencillo, tipo hola mundo

```
const Users =()=>{
    return <h2>Pagina de usuarios</h2>
}

export default Users
```

14. Lab3 - p6: pages/User

Queremos que User acepte un parametro. Si te fijas en el ejemplo hasta ahora no lo tienes funcionando. Definiremos useParams y lo utilizaremos con desestructuracion.

```
import { useParams } from "react-router-dom";
const User =()=>{
    const { userId } = useParams();
    return <p>Esto es el usuario {userId}</p>;
}
export default User
```

Ahora nos falta enlazarlo adecuadamente con nuestros routers. Anadimos un elemento de fail over para rutas nos definidas.

```
  <React.StrictMode>
      <BrowserRouter>
          <Routes>
              <Route path="/" element={<App />}>
                  <Route path="users" element={<Users />}/>
                  <Route path="user/:userId" element={<User />} />
              </Route>
              <Route
                  path="*"
                  element={
                      <main style={{ padding: "1rem" }}>
```

```
                        <p>There's nothing here!</p>
                    </main>
                }
            />
        </Routes>
    </BrowserRouter>
  </React.StrictMode>,
```

15. Otras funcionalidades de Router

- UseParams > p6
- UseLocation
- ActiveLinks
- Search Params
- useNavigate

16. UseLocation

UseLocation, permite conocer la url actual

```
import { useLocation } from 'react-router-dom';
function Product() {
  const location = useLocation();
  useEffect(() => {
    const currentPath = location.pathname;
    const searchParams = new URLSearchParams(location.search);
  }, [location]);
  return <p>Product</p>;
}
```

17. ActiveLinks

Si cambias Link por NavLink tienes la opcion de hacer lo siguiente para saber si el link esta activo:

```
<NavLink
   style={({ isActive }) => {
     return {
       display: "block",
```

```
        margin: "1rem 0",
        color: isActive ? "red" : "",
      };
    }}
    to={`/invoices/${invoice.number}`}
    key={invoice.number}
  >
    {invoice.name}
  </NavLink>
```

18. Searchparams

Tambien permite la captura de informacion en la url por metodo GET (ejemplo "/login?success=1")

```
import {
  NavLink,
  Outlet,
  useSearchParams,
} from "react-router-dom";
....
let [searchParams, setSearchParams] = useSearchParams();
....
 let filter = searchParams.get("filter");
....
```

https://reactrouter.com/docs/en/v6/getting-started/tutorial#search-params

19. useNavigate

Si quieres enviar el usuario a una parte de la pagina programaticamente es sencillo

```
import {
  useParams,
  useNavigate,
  useLocation,
} from "react-router-dom";
....
 <button
          onClick={() => {
            deleteInvoice(invoice.number);
```

```
            navigate("/invoices" + location.search);
          }}
        >
          Delete
        </button>
```

20. Helmet

Para poder atender a requerimientos SEO, puedes utilizar esta libreria si te bases en React*. Curiosamente, esta generada por la NFL, de ahi el nombre.

https://www.npmjs.com/package/react-helmet

- Soporta todas las etiquetas head válidas: title, base, meta, link, script, noscript y style.
- Admite atributos para las etiquetas body, html y title.
- Soporta el renderizado del lado del servidor.
- Los componentes anidados anulan los cambios de cabecera duplicados.
- Los cambios de cabecera duplicados se conservan cuando se especifican en el mismo componente (soporte para etiquetas como "apple-touch-icon").
- Llamada de retorno para el seguimiento de los cambios en el DOM.

21. Helmet

```
import React from "react";
import {Helmet} from "react-helmet";
class Application extends React.Component {
  render () {
    return (
        <div className="application">
            <Helmet>
                <meta charSet="utf-8" />
                <title>My Title</title>
                <link rel="canonical" href="http://mysite.com/example" /
>
            </Helmet>
            ...
        </div>
    );
  }
```

```
};
```

22. Recursos de aprendizaje

- https://stackblitz.com/
- https://www.positronx.io/react-router-hooks-examples/
- https://www.freecodecamp.org/news/react-router-in-5-minutes/
- https://reactrouter.com/docs/en/v6/getting-started/tutorial