https://codesandbox.io/s/lecture-15-sgzncm

React Hooks

- `useState`
- `useEffect`
- `useRef`
- `useMemo`
- `useCallback`
- `useContext`
- `useReducer`

useRef

Ref.jsx
RefDOM.jsx

- `useRef` returns a mutable ref object whose `.current` property is initialized to the passed argument (`initialValue`).
- The returned object will persist for the full lifetime of the component.

```
function useRef(initialValue) {
  const [ref, _] = useState({ current: initialValue });
  return ref;
}
```

refs vs state

refs	state
`useRef` returns `{current: initialValue}`	`useState` returns `[value, setValue]`
does not trigger re-render	triggers re-render
mutable	"immutable"

`useMemo`

- `useMemo` returns a memoized value.
- second argument is an array of dependencies, like `useEffect`.
- only recompute the memoized value when one of the dependencies has changed.
- usage
  - skip expensive calculations on every render
  - pass a callback to a child component that uses `useMemo` / `useCallback` to only re-render when the callback has changed

`useCallback`

- `useCallback` returns a memoized callback.
- second argument is an array of dependencies, like `useEffect`.
- usage
    - with `memo`, skip re-rendering a component if its props haven't changed

Routing in React

- Why do we need routing?
- How?

Why routing?

- Single Page Application (SPA)
- Multiple pages
- Different URLs for different contents
- Navigation

How?

- `react-router-dom`
- https://reactrouter.com/en/main

Installation and Setup

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

Component Router

```
import { BrowserRouter, Routes, Route } from 'react-router-dom';
```

Data Router

```
import { createBrowserRouter, RouterProvider } from 'react-router-dom';
```

Main Concepts

- Subscribing and manipulating the `history` stack
- Matching the URL to your routes
- Rendering a nested UI from the route matches

`history` stack

- Browsers maintain their own history stack as the user navigates around.

Action	Stack
clicks a link to `/home`	`/home`
clicks a link to `/about`	`/home`, `/about`
clicks the back button	`/home`
clicks the forward button	`/home`, `/about`

`window.location`

- `window.location` is an object that contains information about the current URL.
- `window.location.href = <url>` redirects the page to the specified URL.

Router

Router-v5.jsx Router-v6.jsx

```jsx
<Router>
  <Switch>
    <Route path="/about">
      <About />
    </Route>
    <Route path="/dashboard">
      <Dashboard />
    </Route>
    <Route path="/">
      <Home />
    </Route>
  </Switch>
</Router>
```

```jsx
<Routes>
  <Route path="/" element={<Home />}>
    <Route path="/about" element={<About />} />
    <Route path="/dashboard" element={<Dashboard />} />
  </Route>
</Routes>
```

`Link`

- `Link` is a component that renders an anchor tag (`<a>`) with a `href` to a location in the application.
- `reloadDocument` prop
- `state` prop

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

Route Matching

- `path` prop
- `element` prop
- `index` prop

```jsx
<Route path="/" element={<App />}>
  <Route index element={<Home />} />
  <Route path="users" element={<Users />}>
    <Route path=":userId" element={<User />} />
    <Route path=":userId/edit" element={<User mode="edit" />} />
    <Route path="signup" element={<SignUp />} />
  </Route>
</Route>
```

Programmatic Navigation

- `history.push(<url>)`
- `history.pop()`
- `history.replace(<url>)`
- `useNavigate` - `navigate(<url>)`

Private and Protected Routes

- `ProtectedRoute` to check if `user` exists
- if not, redirect to `/login`

React Router Hooks

- `useParams()`
- `useLocation()`
- `useNavigate()`

Data Router: `loader`

data-router/UserDetail.jsx

Why loader?

- Fetch data before rendering a route
- automatic revalidation
- integrates with error boundary

Performance

- `React.lazy()` and `Suspense`