<https://www.linkedin.com/pulse/config-eslint-prettier-vs-code-react-js-anurag-kumar/>

https://www.ezfrontend.com/blog/lo-trinh-hoc-reactjs-co-ban-2020

<https://javascript.info/>

alt+shift+0 để đóng mấy cái không dùng trong jsx

linh react doc: https://react.dev/learn/tutorial-tic-tac-toe#completing-the-game

* Create project with reactjs: npx create-react-app yournameproject



{

  "name": "demo-app",

  "version": "0.1.0",

  "private": true,

  "dependencies": {

    "@testing-library/jest-dom": "^5.17.0",

    "@testing-library/react": "^13.4.0",

    "@testing-library/user-event": "^13.5.0",

    "classnames": "^2.5.1",

    "react": "^18.2.0",

    "react-dom": "^18.2.0",

    "react-router-dom": "^6.22.2",

    "react-scripts": "5.0.1",

    "web-vitals": "^2.1.4"

  },

  "scripts": {

    "start": "react-scripts start",

    "build": "react-scripts build",

    "test": "react-scripts test",

    "eject": "react-scripts eject"

  },

  "eslintConfig": {

    "extends": [

      "react-app",

      "react-app/jest"

    ]

  },

  "browserslist": {

    "production": [

      ">0.2%",

      "not dead",

      "not op\_mini all"

    ],

    "development": [

      "last 1 chrome version",

      "last 1 firefox version",

      "last 1 safari version"

    ]

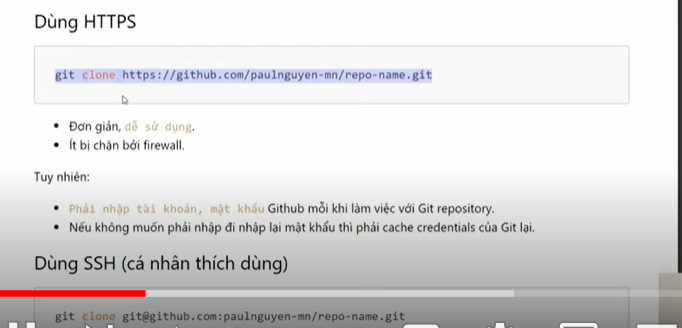
  },

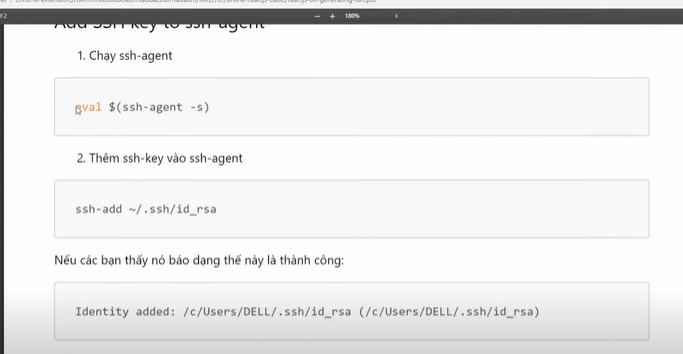
  "devDependencies": {

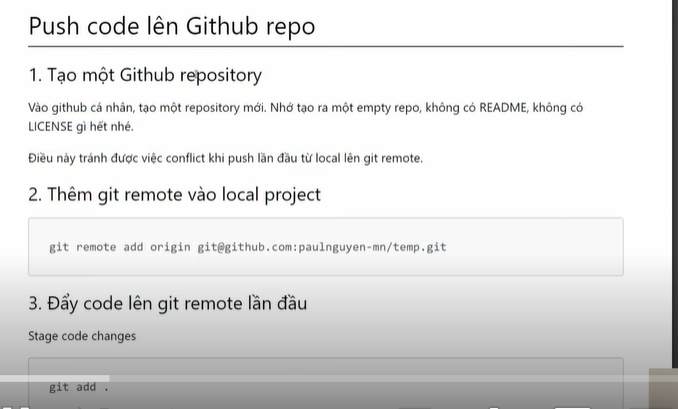
    "sass": "^1.71.1"

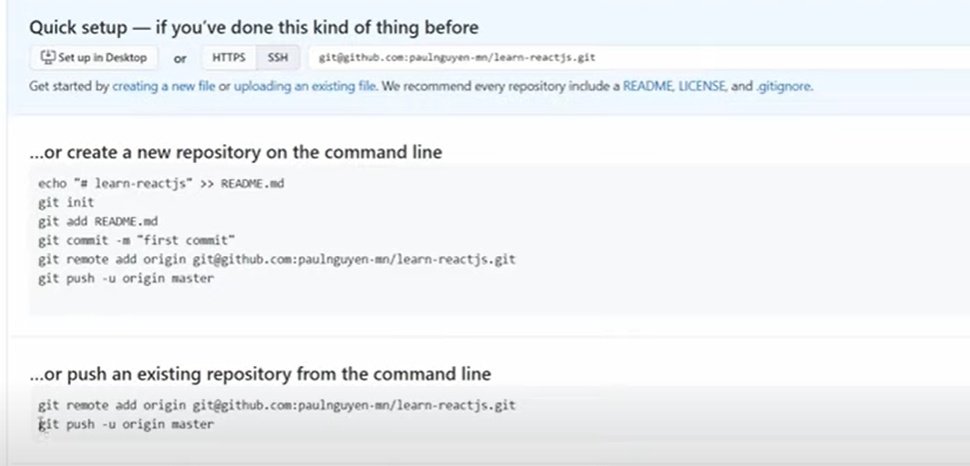
  }

}







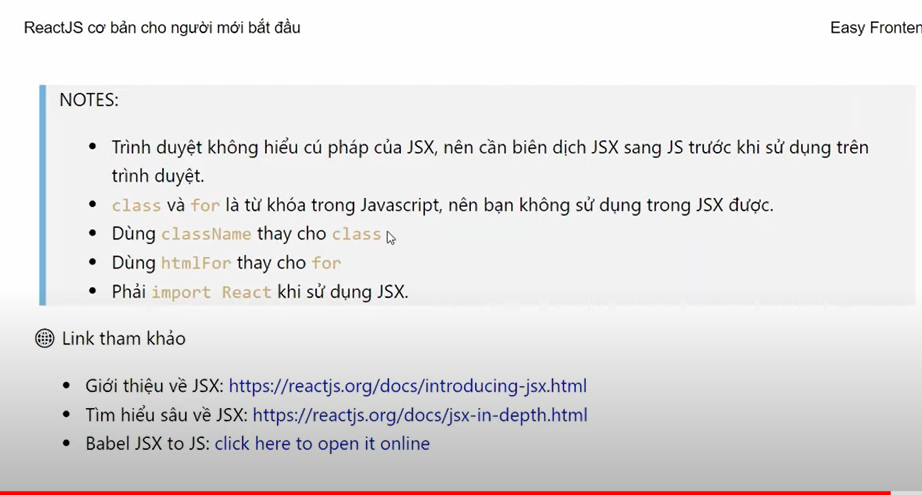


Đuôi .js

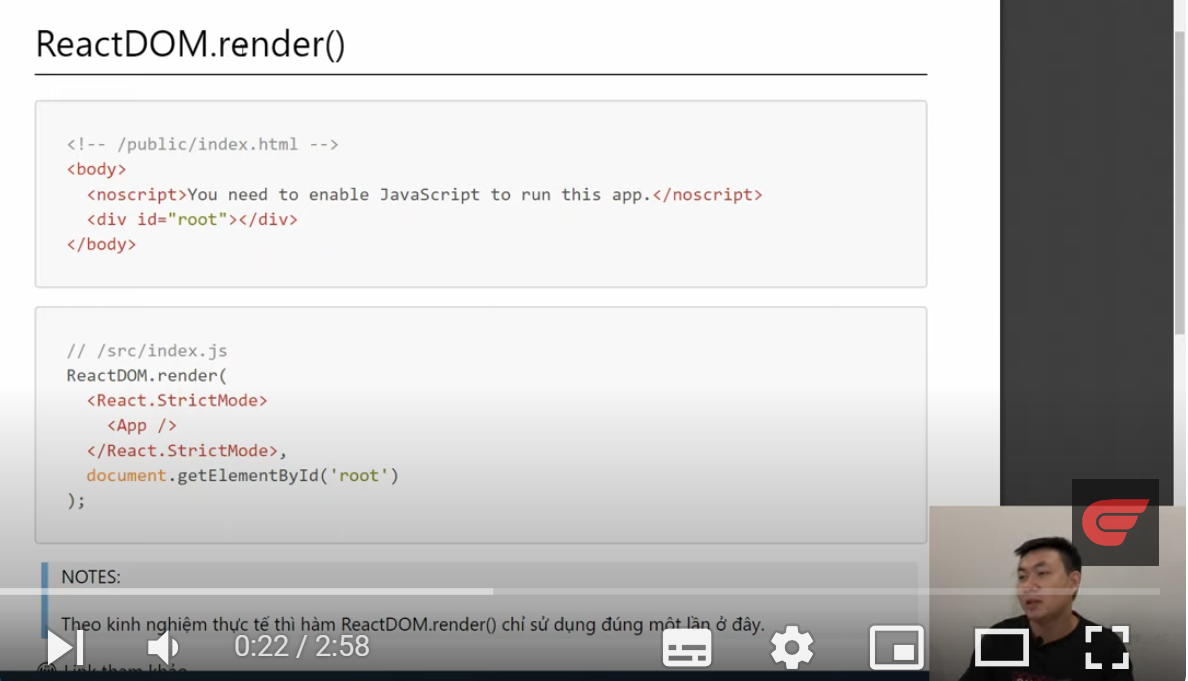
Phải import React vào khi sử dụng jsx

JSX viết tắt của chữ javascript XML

Clas trong React thì dùng className, còn for thì dùng htmlFor



Render code:



const root = ReactDOM.createRoot(document.getElementById('root'));

root.render(

  <React.StrictMode>

    <App />

  </React.StrictMode>

);

// If you want to start measuring performance in your app, pass a function

// to log results (for example: reportWebVitals(console.log))

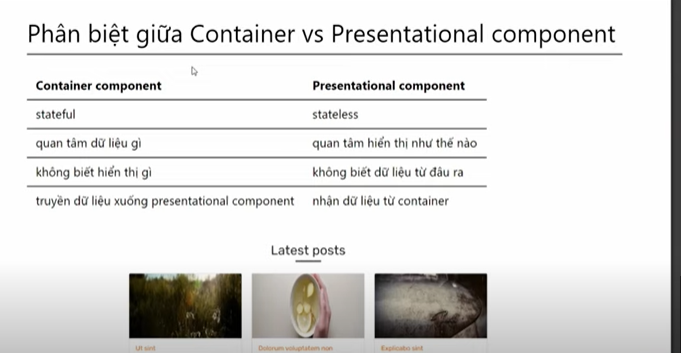
// or send to an analytics endpoint. Learn more: https://bit.ly/CRA-vitals

reportWebVitals();

Lý giải khi render: Render cái phần app của mình ở chế độ strictmode vào id root -> sau đó bỏ vào html của mình -> chỉ mount 1 lần vào trong ông root

Class component: trong class nó có hàm render sẽ trả về jsx

Function component: trong hàm viết gì đó cuối cùng , nó nó có return cho 1 jsx



Phân tích 1 trang filter: Có 1 thằng cha ở trên

+ Thằng cha có dữ liệu gồm: joblist, 1 object chứa tất cả những cái filter ở đây

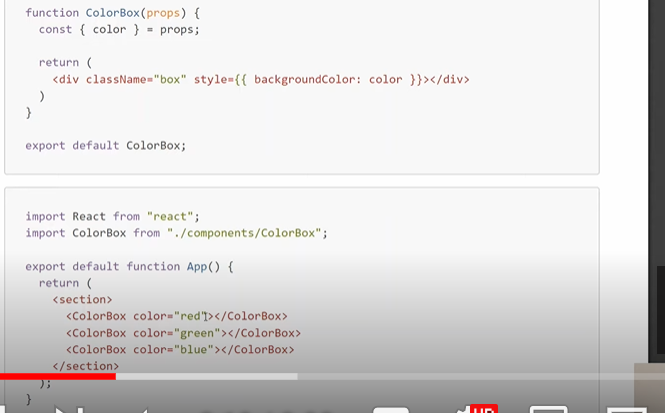
+ Chia ra những component bên dưới cho việc filter (mỗi lần filter thay đổi) -> chỉ cần thay đổi object filter -> đi gọi lại api (dữ liệu) -> bảo thằng con render lại là xong

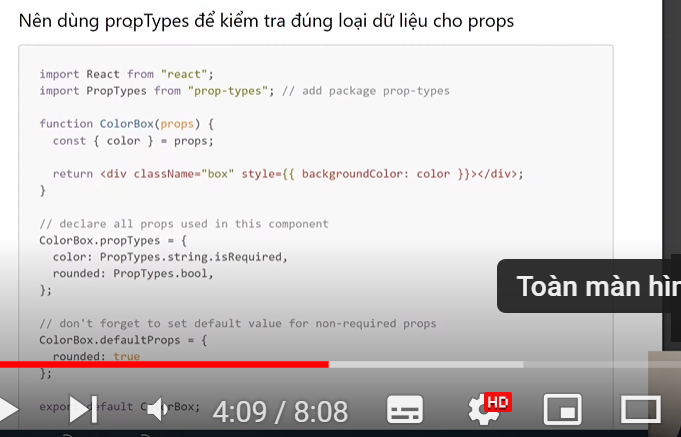


Prop là dữ liệu được truyền từ component cha xuống component con, nó không được thay đổi ở component con, muốn đổi thì nhờ component cha.

Props giúp tạo ra sự đa dạng cho component, cùng 1 component với props khác nhau thì render ra khác nhau







Nếu không bắt buộc thì không truyền isRequired

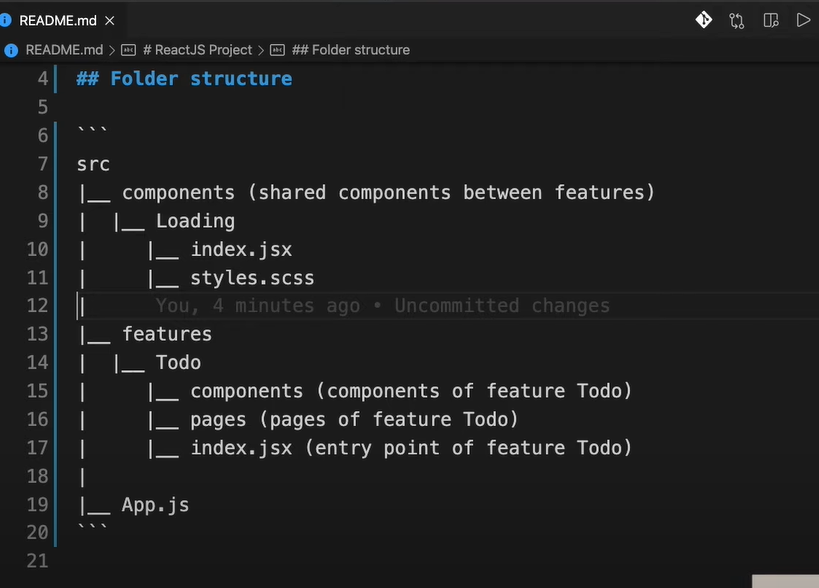
­­­<https://legacy.reactjs.org/docs/typechecking-with-proptypes.html>

(link kiểm tra điều kiện)

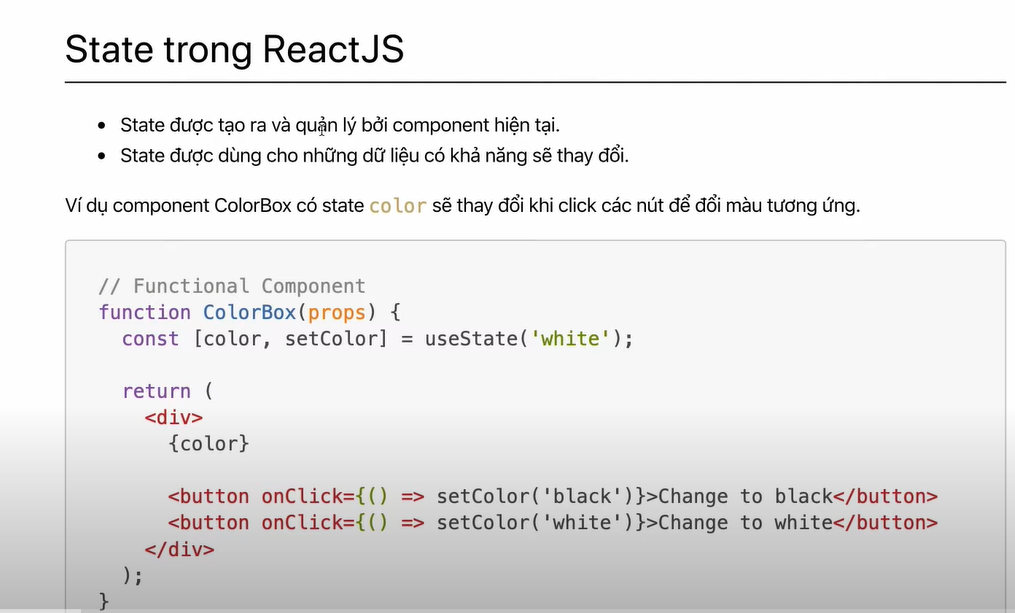
*Lưu ý:khi set điều kiện hoặc null, undefined thì nó sẽ không render lên màn hình gì*

**Trong cái mảng thì phải truyền vào cái key để nó biết đó là thằng con này**

**(mỗi thằng con trong cái list (mảng) là bạn phải chỉ định cho nó cái key, chứ không nó sẽ chửi ngay**



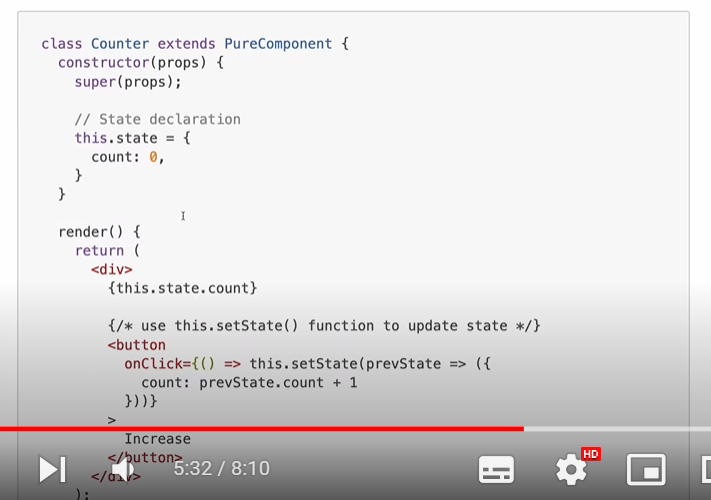
**State nó được tạo ra quản lý chính trong component của nó thay đổi được**



**Khi props, state thay đổi thì component nó sẽ rerender lại**

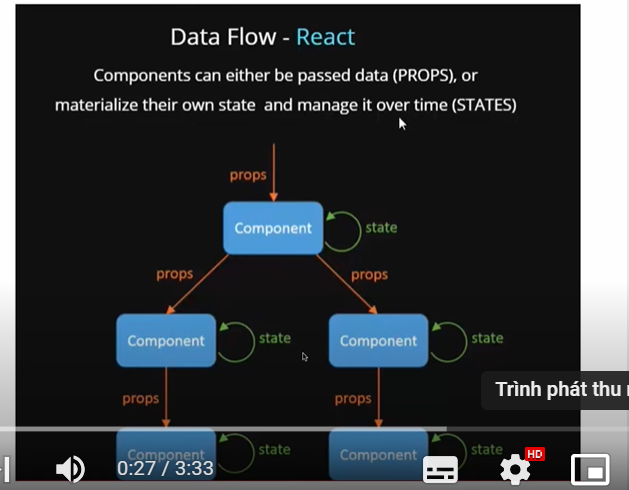


**Viết theo class:**



**Truoc khi update 1 state moi thi minh phai clone no ra (dung …listgiatrichuatruocdo)**



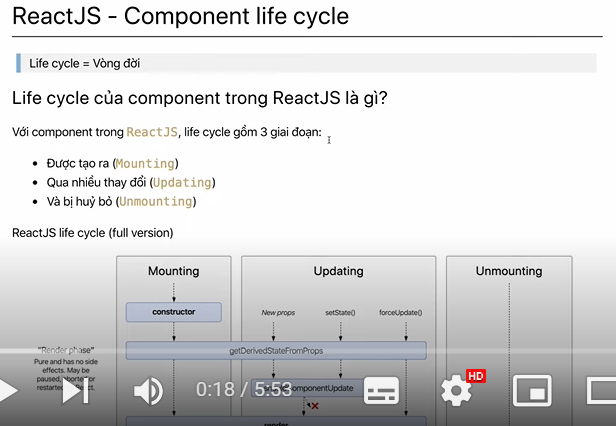


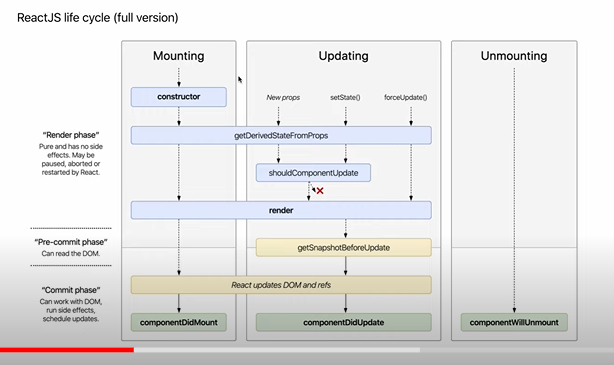
**Props từ cha truyền xuống không thay đổi được**

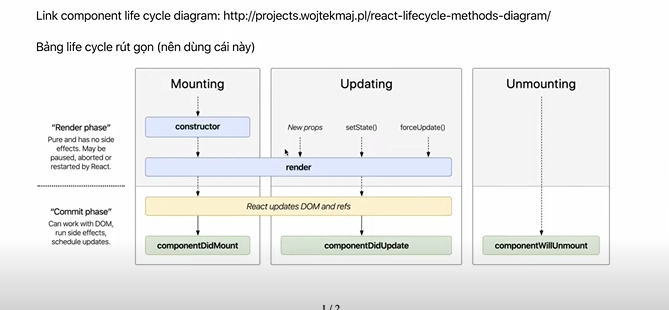
**State của bản thân component đó có thể thay đổi được**

**useState()** là một hook cơ bản giúp sử dụng state trong Functional Component. Hook này nhận vào input là một giá trị hoặc function và output là một mảng có hai phần tử tương ứng cho state và setState. Khi useState được gọi, nó khai báo một biến state. Đây là cách để "lưu giữ" các giá trị giữa các lần gọi hàm — useState là một cách mới để sử dụng như là cách this.state được dùng trong class. Thông thường, các biến này "biến mất" khi hàm kết thúc nhưng các biến state này được React giữ lại.

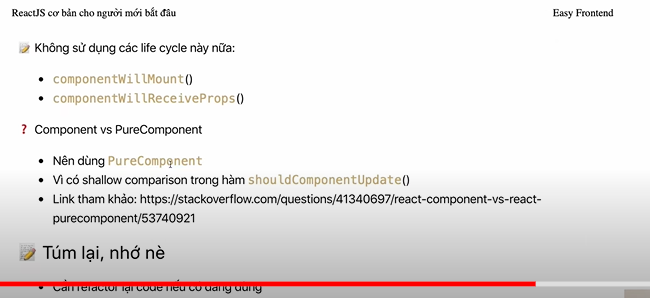
**Nó chỉ nằm trong Classcomponent thôi, còn functional component không có vòng dời**



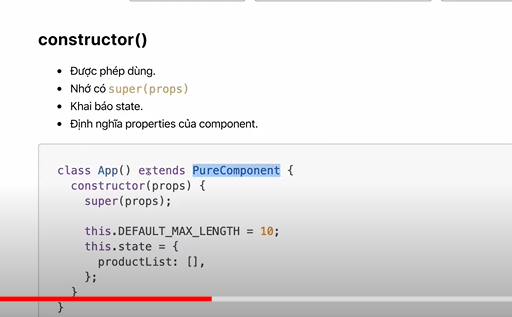




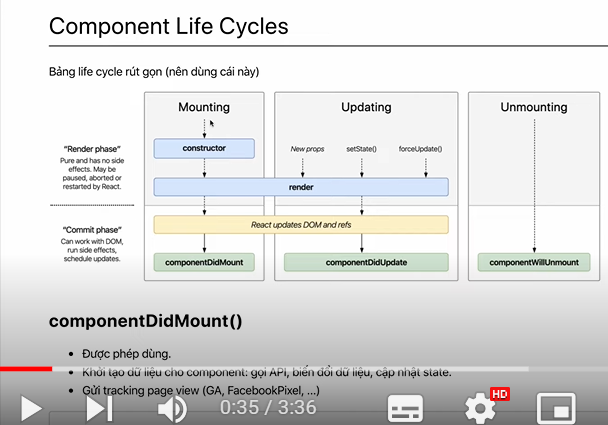
**Thường sử dụng constructor -> componentdidmount -> componentWillUnmount**



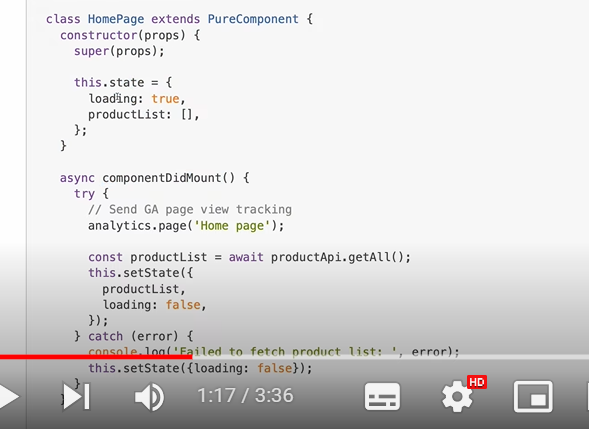
1. **Constructor()**
   1. **Được phép dùng**
   2. **Nhớ có super(props)**
   3. **Khai báo state**
   4. **Định nghĩa properties (thuộc tính) của component**



1. **ComponentDidMount()**
   1. **Được phép dùng**
   2. **Khởi tạo dữ liệu cho component: gọi API, biến đổi dữ liệu, cập nhật state**
   3. **Gửi tracking page view (GA, FacebokFixed)**



**Ví dụ nó chạy**



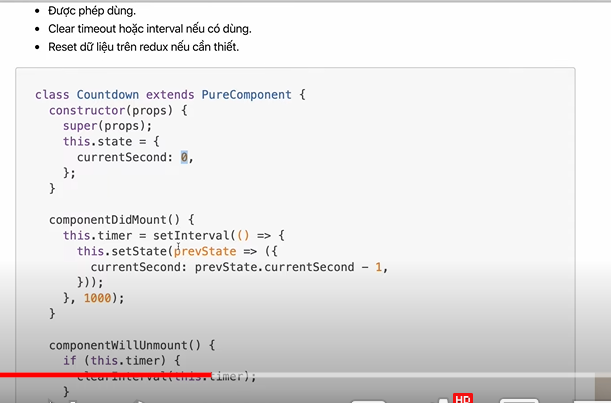
**Constructor nó chạy đầu tiên, sau đó nó render lần đầu xong rồi nó mới chạy lên componentDidMount (nó chỉ chạy 1 lần). (nếu như mình có setState thì nó rerender lại lần nữa)**

1. **ComponentWillUnmount() (chỉ chạy đúng 1 lần)**

**Được phép dùng**

**Clear timeout hoặc interval nếu có dùng**

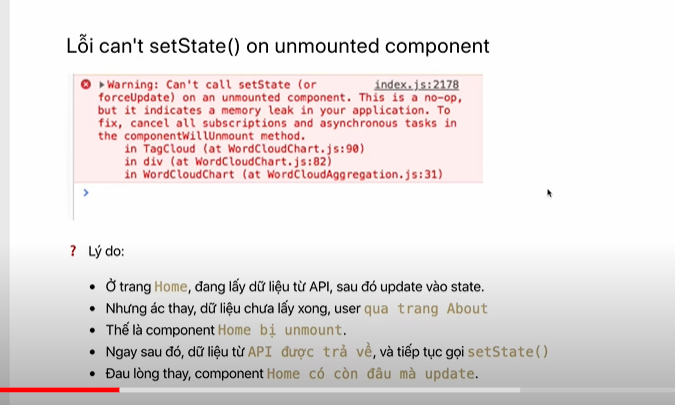
**Reset dữ liệu trên redux nếu cần thiết**

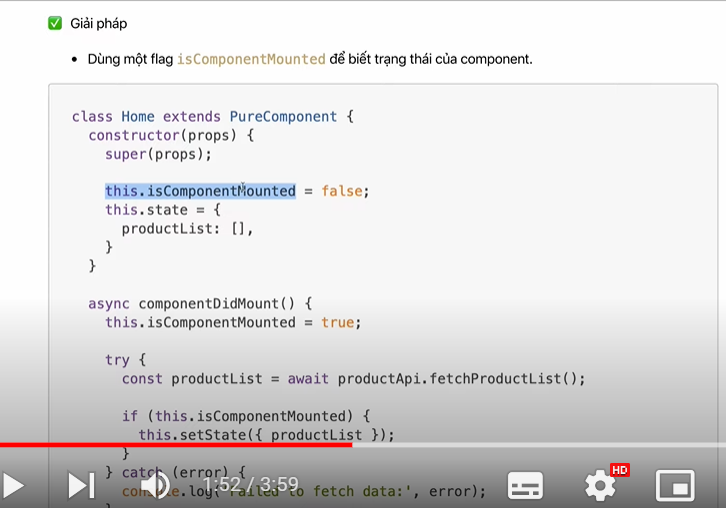


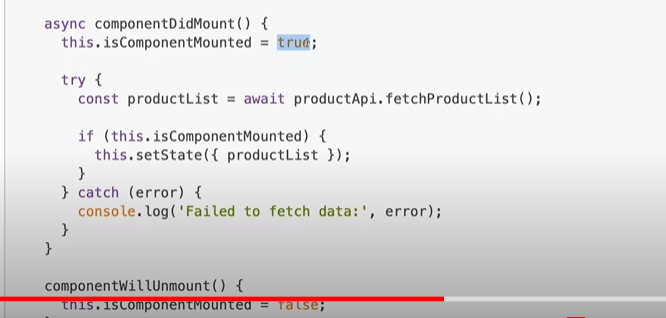
1. **ComponentDidUpdate() (được chạy lại nhiều lần, nếu có state hoặc props update)**

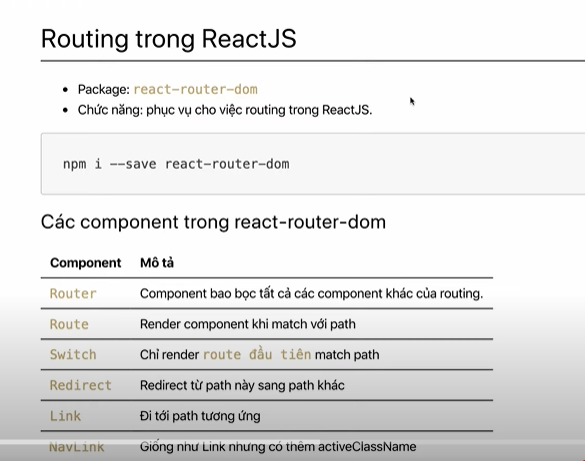
**Cực kỳ hạn chế dùng**

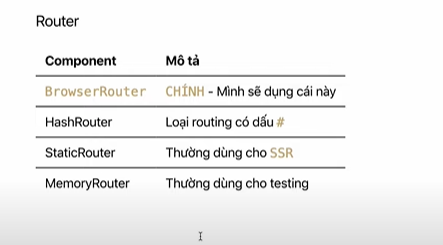
**ADVANCED chỉ dùng nếu muốn handle update component khi click nút back mà treenURL có query params**

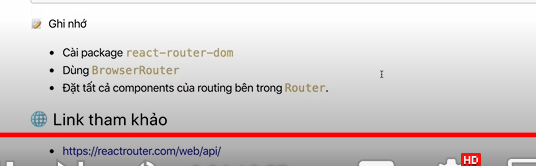












**/// Kinh nghiệm khi dùng Routing cho từng feature**

**Index;js**

import React from 'react';

import ReactDOM from 'react-dom/client';

import App from './App';

import './index.css';

import reportWebVitals from './reportWebVitals';

import {BrowserRouter} from 'react-router-dom';

const root = ReactDOM.createRoot(document.getElementById('root'));

root.render(

  <React.StrictMode>

    <BrowserRouter>

      <App />

    </BrowserRouter>

  </React.StrictMode>

);

// If you want to start measuring performance in your app, pass a function

// to log results (for example: reportWebVitals(console.log))

// or send to an analytics endpoint. Learn more: https://bit.ly/CRA-vitals

reportWebVitals();

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

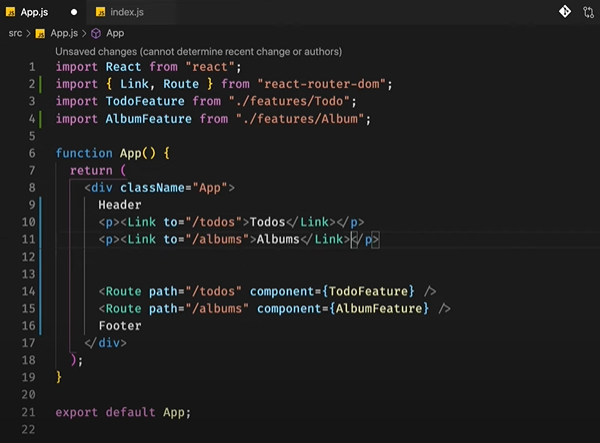
<Routes>

        <Route path="/todos" element={<TodoFeature />} />

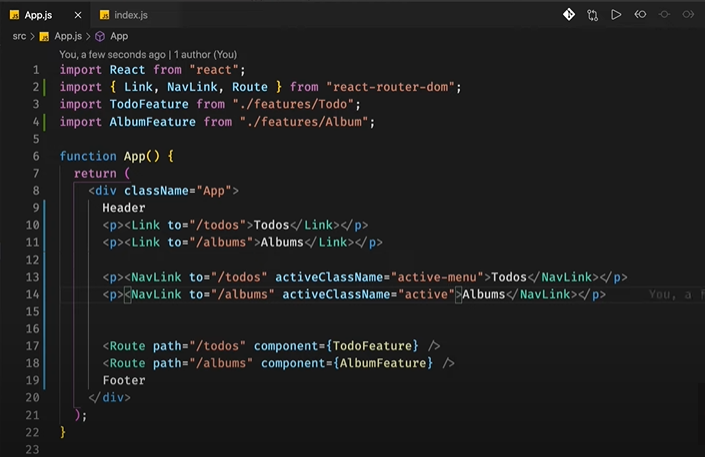
        <Route path="/albums" element={<AlbumFeature />} />

      </Routes>

**Dùng Link**



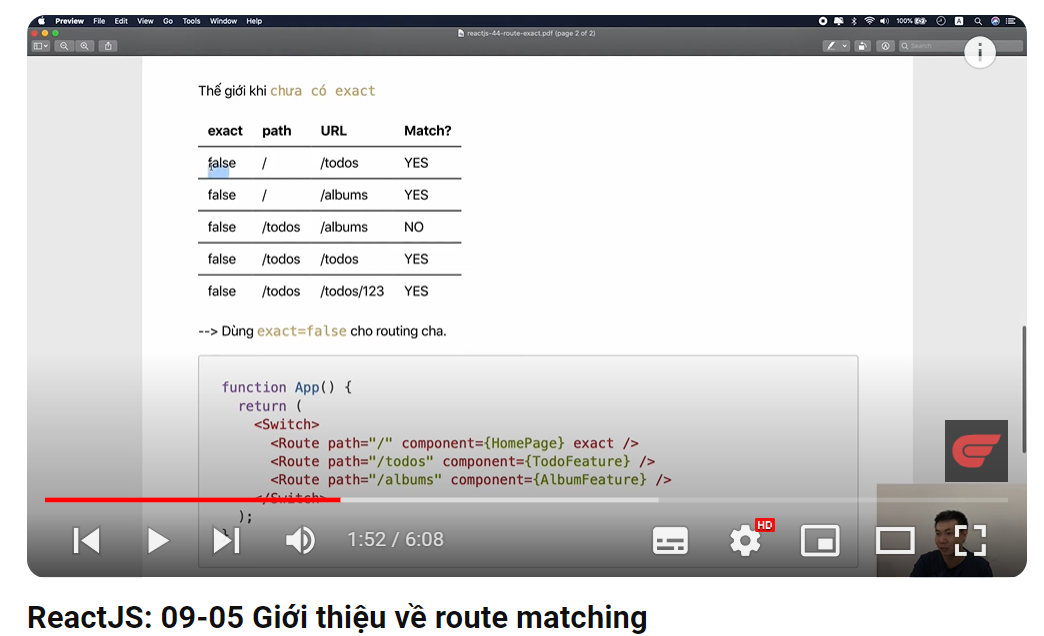
**Phân biệt Link và NavLink (thằng Navlink nó tự động gắn thêm có class=”active”)**



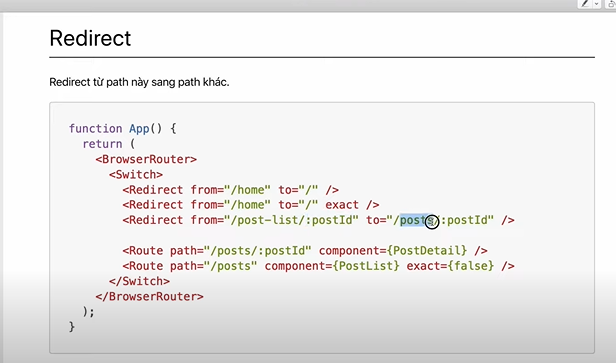
**Phàn biệt Route và switch**

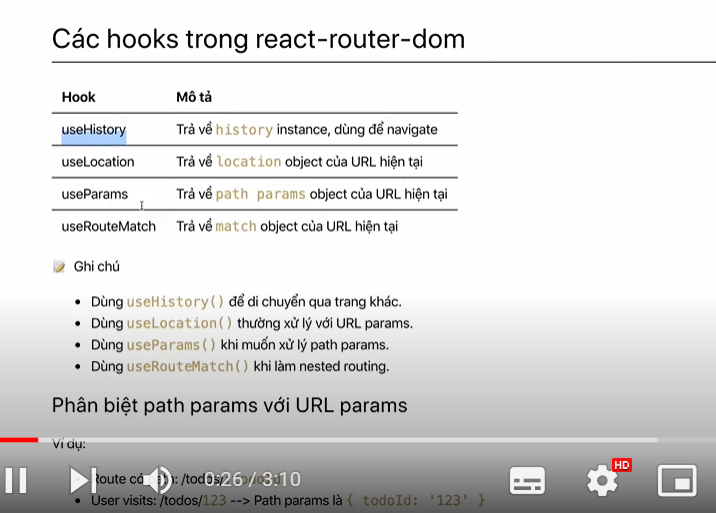
**Switch giúp mình nó sẽ chạy từ trên xuống dưới, nó sẽ check từng route, từng route hiện tại nếu nó map thì nó dừng tại component đó**

**Còn nếu ko dùng switch thì nó sẽ render ra hết luôn những component trong Route**



­­­­­­­­­­





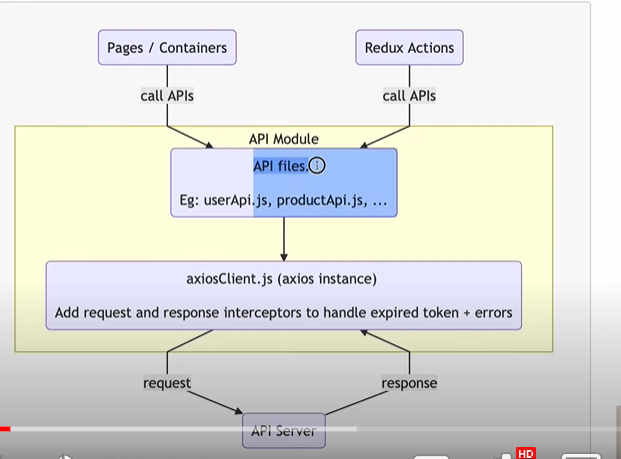


<https://reactrouter.com/en/main>

LEARN API WITH REACTJS

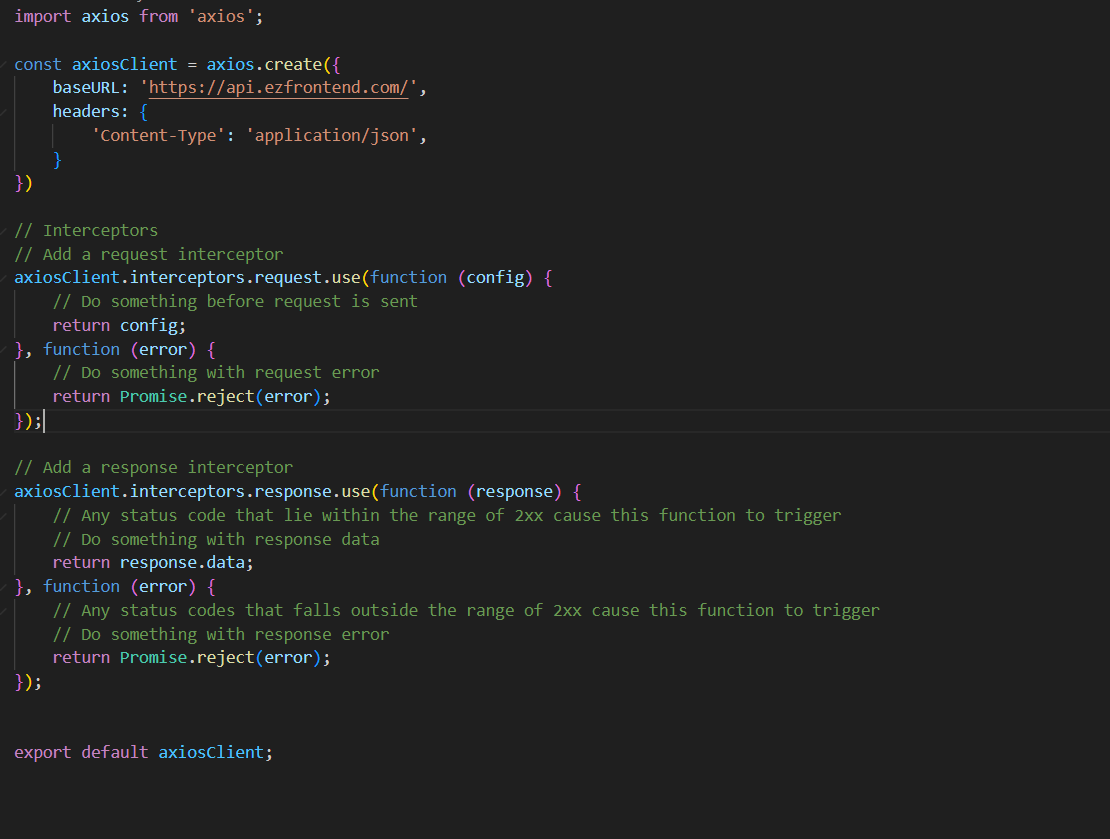
<https://github.com/axios/axios?tab=readme-ov-file#request-config>

<https://react-hook-form.com/>

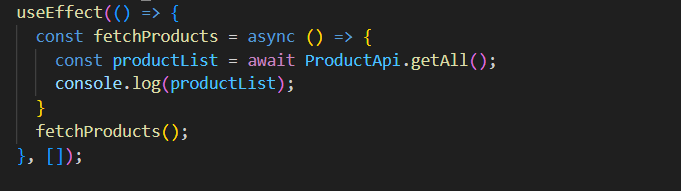


**Note: setup api module, goị được api từ pages/container của mình -> gọi xuông api file -> axiosClient.js -> gọi lên api server -> nó trả về response -> rồi nó trả ngược lên lại**

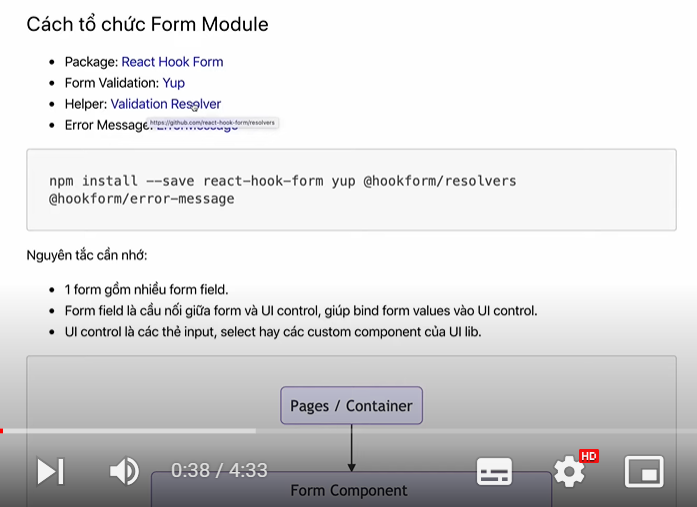


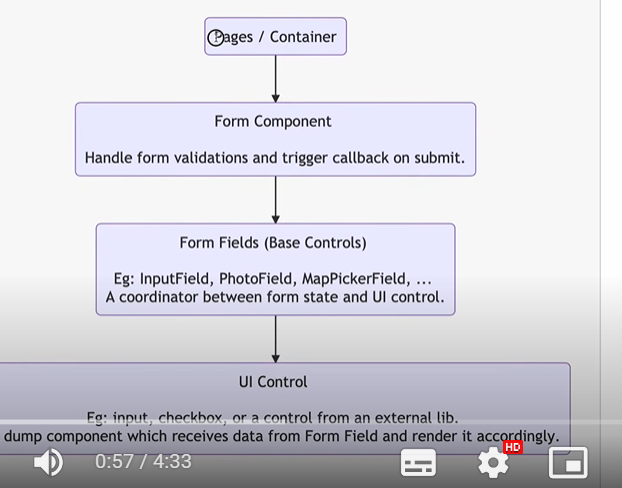


**Call api:**

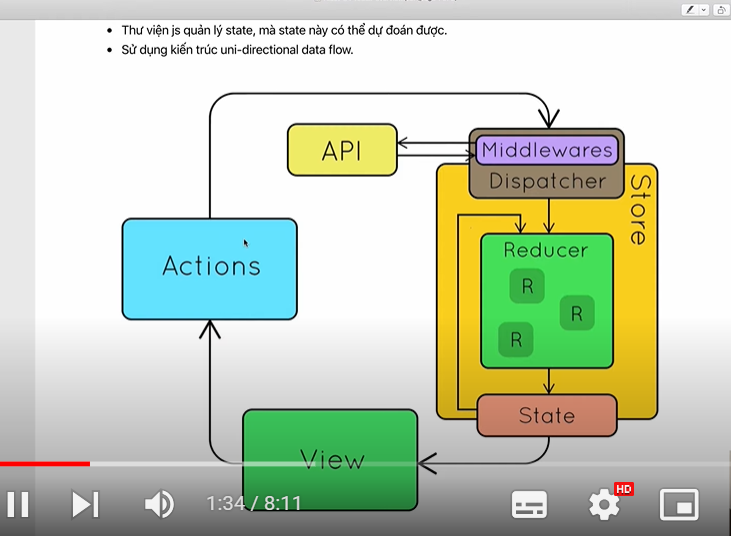




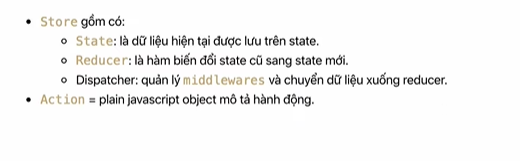


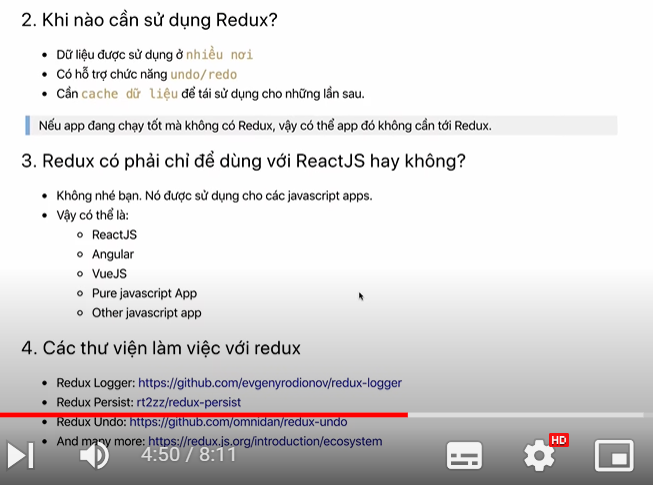


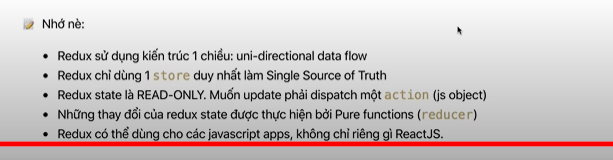
**npm i --save react-hook-form yup @hookform/resolvers @hook form/error-message**

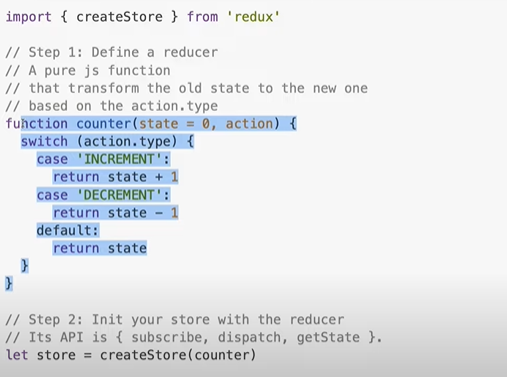


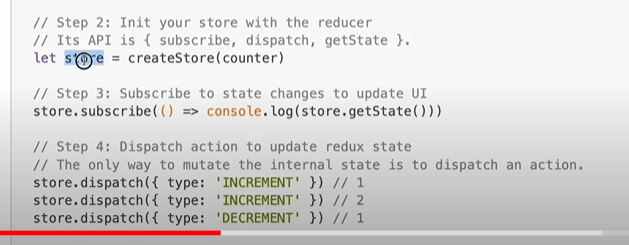
**View gửi action (trigger) lên store -> ông reducer sẽ chỉnh sửa cập nhật xuống state**



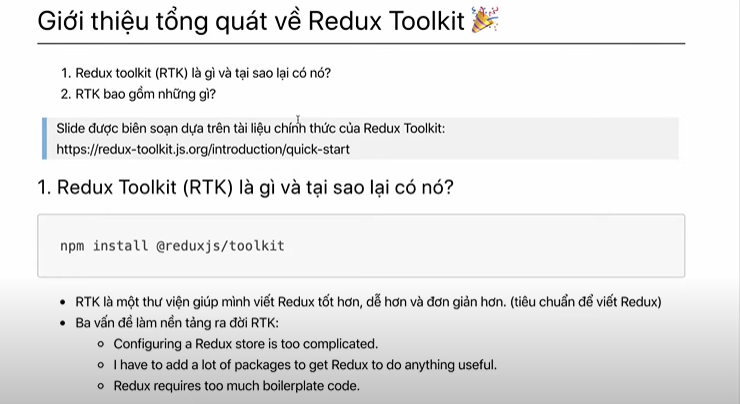


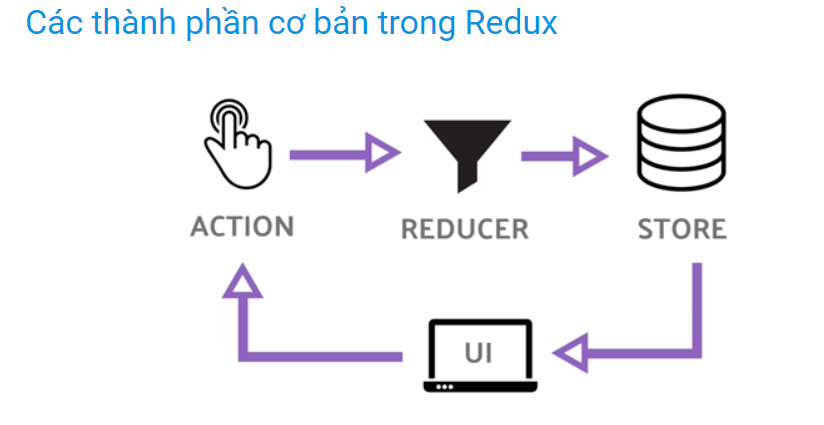


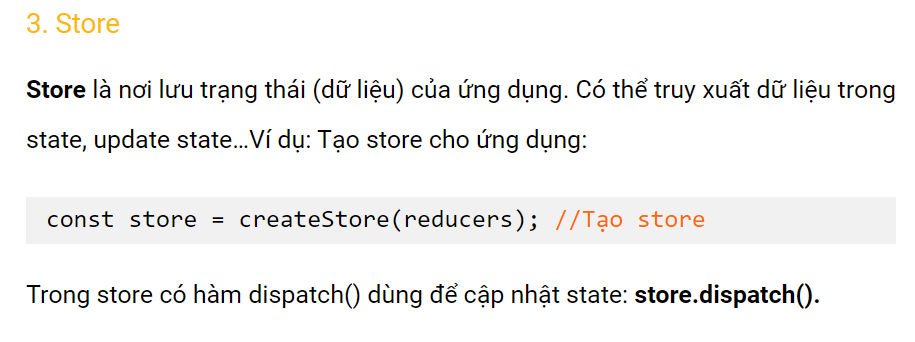


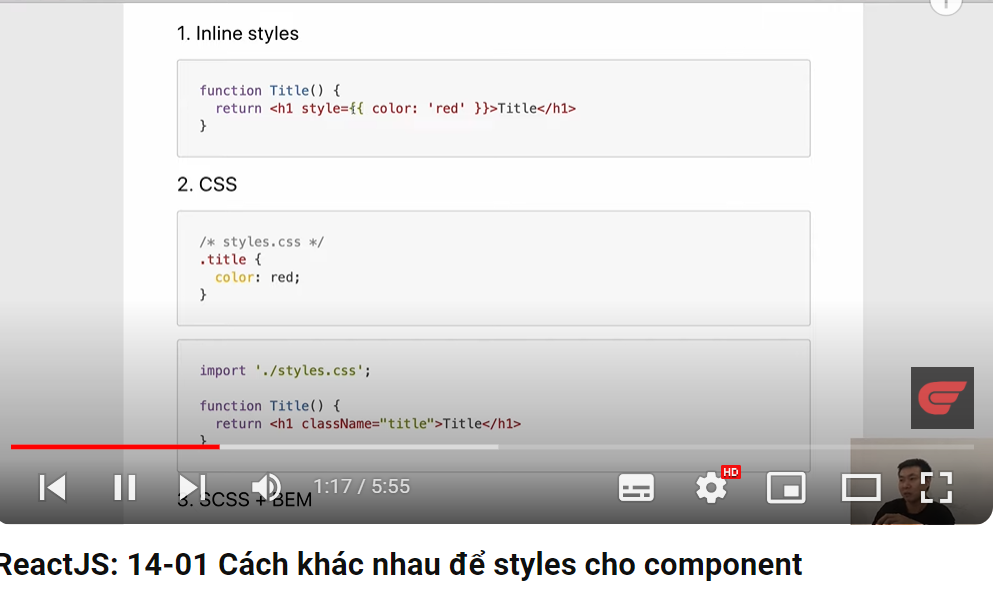


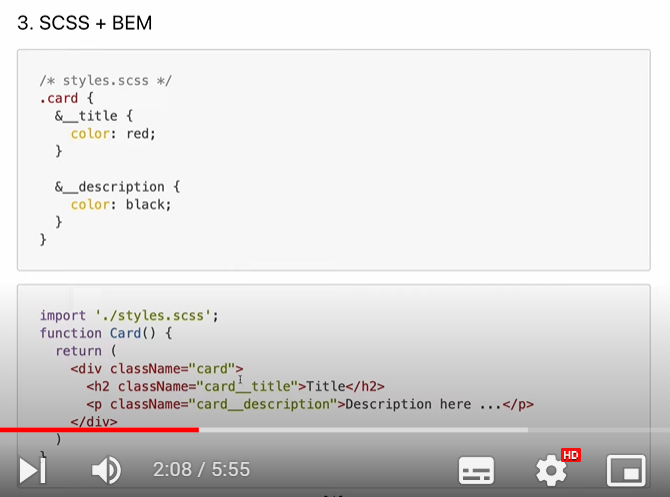




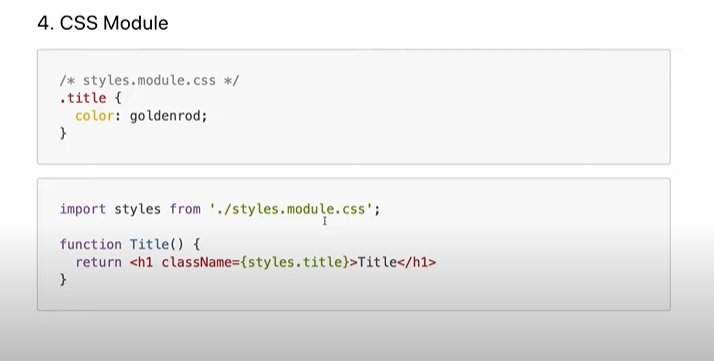


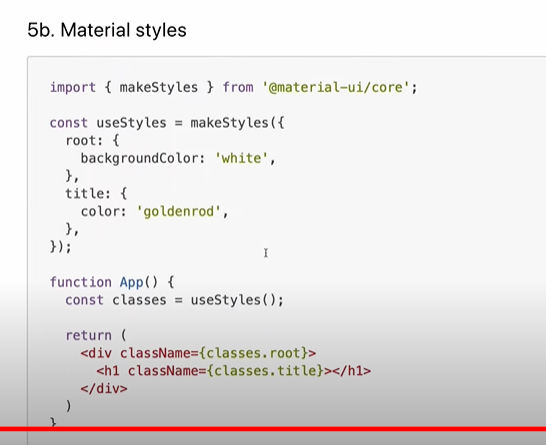


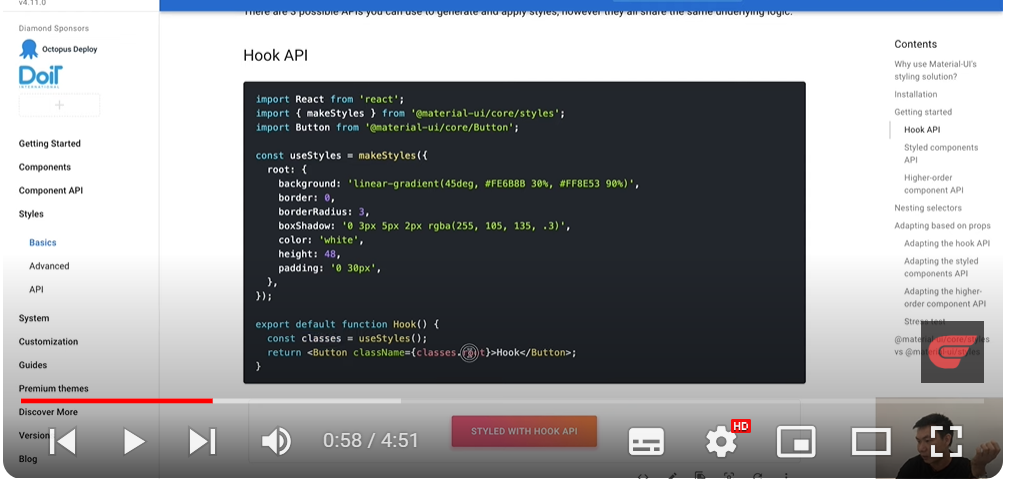


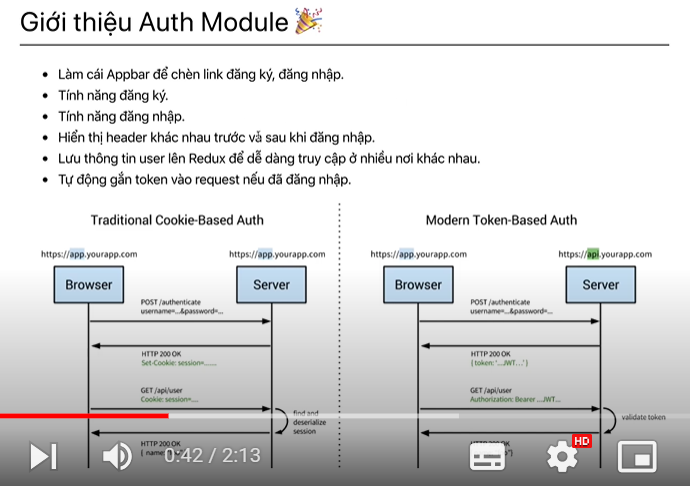


**Cách đặt css theo bema album\_\_name(nó chỉ cha với con), album\_\_name—active (là nó chỉ trạng thái của thằng name)**



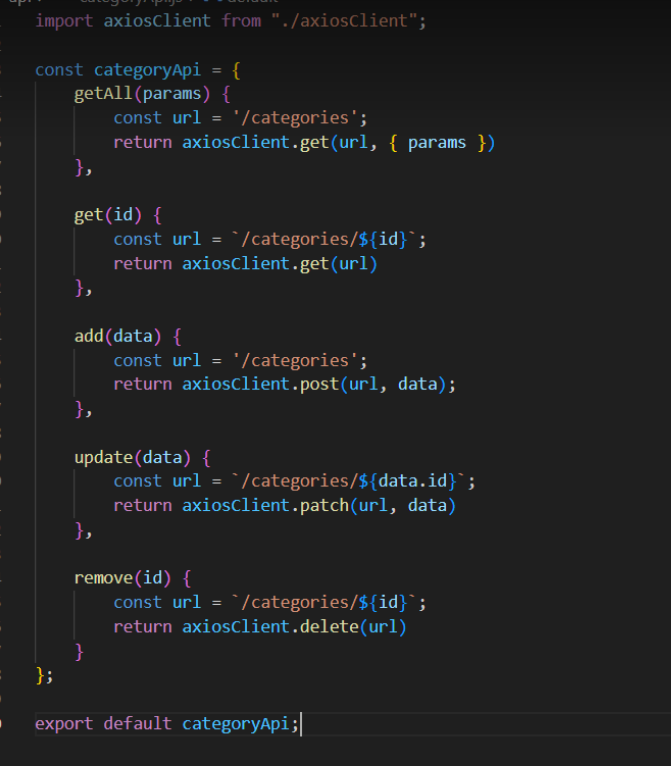


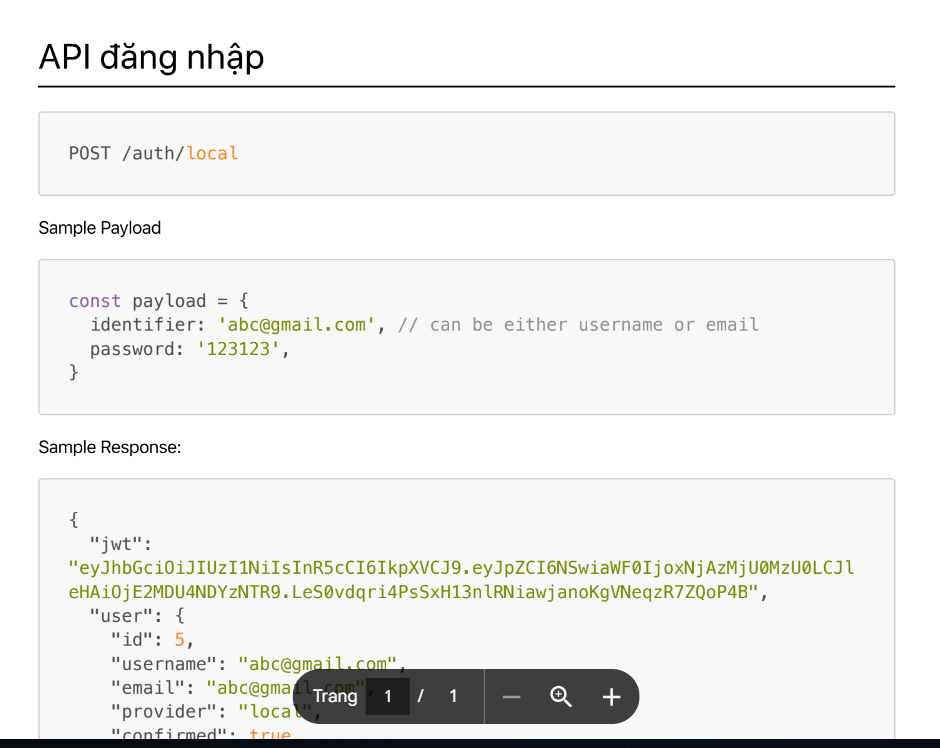


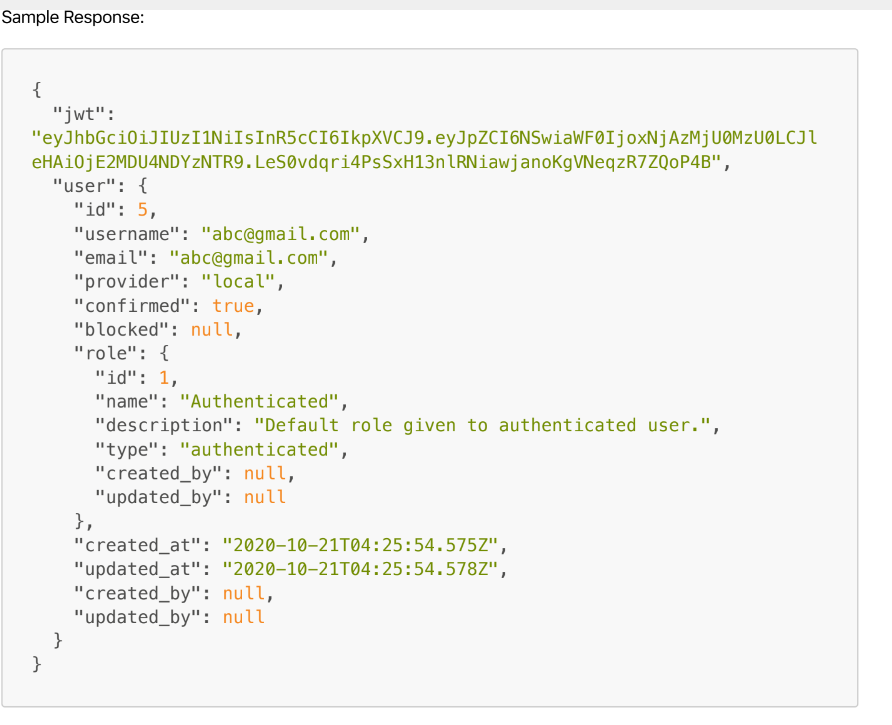


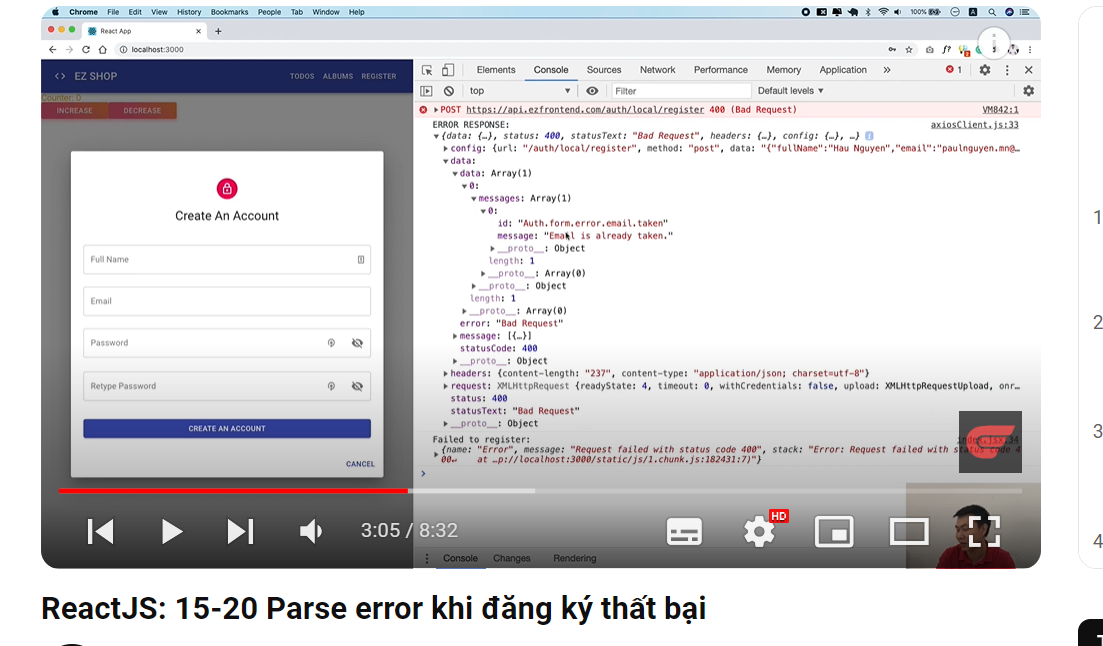
**Gọi api đăng ký => tạo 1 user slice-> gắn nó vào store để nó nhận diện được userReducer=>tạo action(a singe action, phải gọi api và lấy kết quả trả về, mới update vô UserSate)**

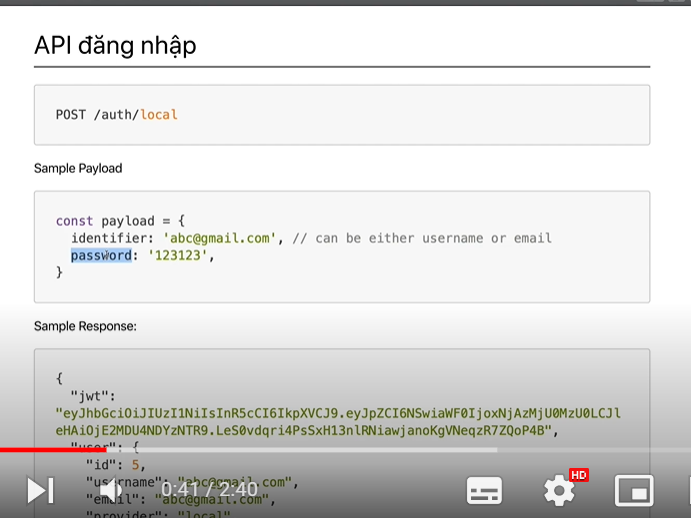
[**https://redux-toolkit.js.org/api/createAsyncThunk**](https://redux-toolkit.js.org/api/createAsyncThunk)

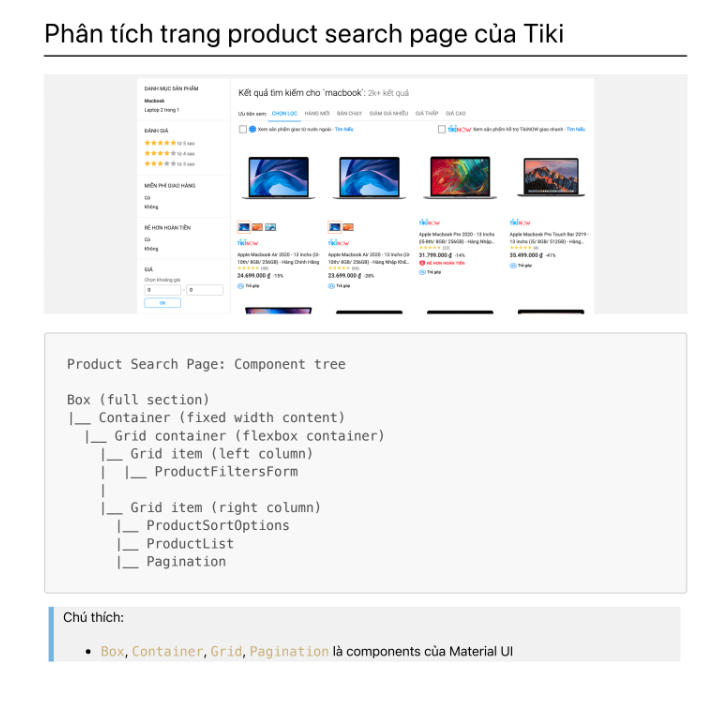


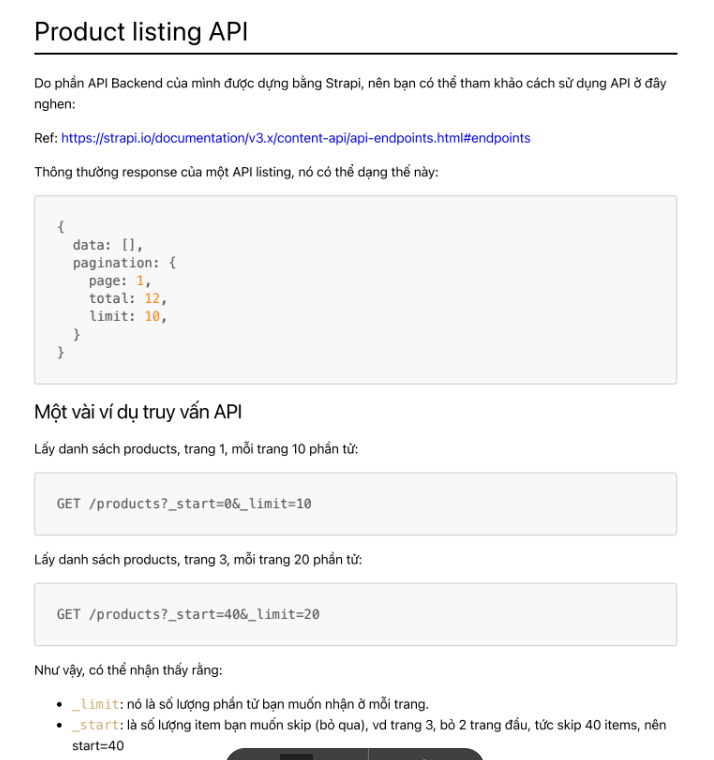


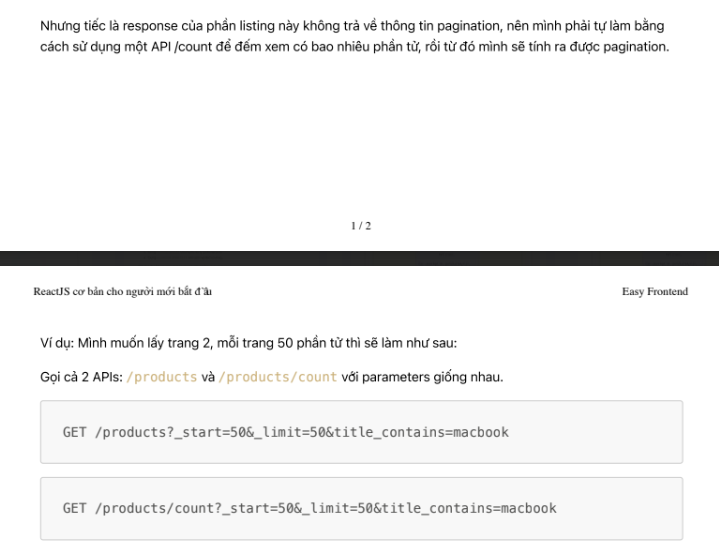


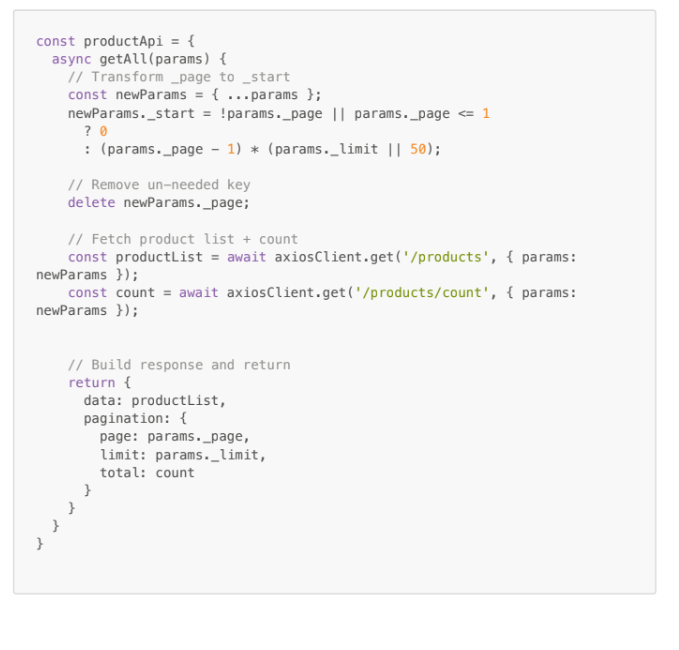












Từ phiên bản react-redux đã cho ra đời 2 em hooks thần thánh là useSelector và useDispatch. Trước đây thì ta có connect() – một Higher Order Component (HOC) giúp chúng ta nhận state và dispatch action từ store tại component thay thế cho hàm connect. Hiện tại một vài dự án của mình cũng đang sử dụng 2 em này.

**- useSelector là gì?**

Hook này cho phép chúng ta lấy state từ Redux store bằng cách sử dụng một selector function làm tham số đầu vào. Trong đoạn code phía trên bạn thấy thì mình có trả về mảng products từ store. Mặc dù nó thực hiện công việc như mapStateToProps nhưng nó vẫn có một số khác biệt mà bạn cần phải quan tâm.

* mapStateToProps chỉ return về 1 object, còn useSelector có thể return bất cứ giá trị nào
* Khi **dispatch một action**, useSelector sẽ thực hiện **so sánh tham chiếu** với giá trị được return trước đó và giá trị hiện tại. Nếu chúng khác nhau, component sẽ bị re-render. Nếu chúng giống nhau, component sẽ không re-render. mapStateToProps là một function sẽ luôn được chạy lại mỗi khi store có một sự thay đổi bất kì nào trong đó. Với mapStateToProps, tất cả các trường được return lại thành một dạng object kết hợp. Vậy nên mỗi khi mapStateToProps chạy thì nó sẽ return về một object với tham chiếu mới. Hàm connect() sẽ thực hiện so sánh nông với object mà mapStateToProps trả về, nếu khác nhau thì sẽ re-render lại component. Tức hiểu cặn kẽ hơn là so sánh tham chiếu (so sánh ===) các trường bên trong object mà mapStateToProps trả về, chỉ cần 1 trường khác nhau là sẽ bị coi là khác nhau.

Thoạt nhìn cách so sánh useSelector vs connect() có khác nhau 1 tẹo nhưng nếu ta khai báo nhiều useSelector cho mỗi state khác nhau thay vì gom lại một cục object duy nhất thì cách so sánh lại tương đương với connect(). useSelector có một nhược điểm là khi một component thay đổi, nó khiến cho toàn bộ cây component phải render lại và việc này gây ảnh hưởng đến performance. Để tránh việc re-render nhiều lần các bạn có thể dùng một HOC là React.memo().

**- useDispatch là gì?**

Hook này đơn giản chỉ là return về một tham chiếu đến **dispatch function** từ Redux store và được sử dụng để dispatch các action. Nhưng sẽ có vài điều mà mình cần cho các bạn biết.

## **Khác nhau giữa useSelector và mapState**

* Khi một action được dispatch, useSelector sẽ thực hiện so sánh giữa kết quả trước đó và kết quả hiện tại, nếu khác, component bị force để re-render.
* useSelector sử dụng so sánh === chứ không dùng phương pháp so sánh **shallow** (dịch vui là so sánh “nhẹ”)

