TÀI LIỆU KHÓA HỌC "React State Manager"

Tác giả: Hỏi Dân IT & Eric

Version: 1.0

Note cập nhật:

- Update tài liệu khóa học

Chapter 0: Giới thiệu
Giới thiệu và demo kết quả đạt được sau khi kết thúc khóa học này.

#0. Demo Kết quả đạt được
Link video demo: https://www.youtube.com/watch?v=6I0K-jICIU8&list=PLncHg6Kn2JT4-Wf0V_u5EbHGElgNQMjN3

Mục tiêu của khóa học:

- Nắm vững cách sử dụng Redux và Redux Toolkit
- Fetching data hiệu quả với React Query
- Hiểu rõ các middleware hay dùng với Redux như Redux Thunk và Redux Saga
- Sử dụng Redux với Next.js

#1. Hướng Dẫn Download Tài liệu khóa học

Tài liệu khóa học sẽ được update theo thời gian và được "đánh version".

Nếu trong quá trình khóa học, bạn thấy tài liệu không đầy đủ, thì check lại video này nhé :v

#2. Yêu cầu của khóa học

1. Biết HTML, CSS và cú pháp Javascript (tự học)

Về HTML, CSS => tự học

Về Javascript, tham khảo: (chỉ cần học để hiểu cú pháp khai báo của javascript)
https://www.w3schools.com/js/

Link khóa học Javascript miễn phí:
https://www.youtube.com/playlist?list=PLncHg6Kn2JT5dfQqpVtfNYvv3EBVHHVKo

2. Biết cú pháp typescript:
Link khóa học Typescript miễn phí:
https://www.youtube.com/playlist?list=PLncHg6Kn2JT5emvXmG6kgeGkrQjRqxsb4

3. Biết sử dụng Git để quản lý mã nguồn

Link khóa học Git miễn phí:
https://www.youtube.com/playlist?list=PLncHg6Kn2JT6nWS9MRjSnt6Z-9Rj0pAlo

4.Đã có kiến thức React cơ bản
Khóa học này là "quản lý React state", nên yêu cầu bắt buộc, bạn cần biết React trước khi theo học.

Nếu bạn chưa biết gì về React, theo học không phù hợp.

#3. Về khóa học này

Mục tiêu:

- Chỉ 1 khóa học duy nhất, học quản lý state của React với : Redux và React Query

=> giải quyết bài toán:

+ Sharing data giữa các component

+ fetching/caching data

=> phục vụ mục đích đi làm tại các công ty (giải quyết các bài toán thực tế)

Về chuyện leak khóa học và mua lậu

Mình biết rất nhiều bạn khi học khóa học này của mình, là mua lậu qua bên thứ 3. chuyện này là hoàn toàn bình thường, vì thương hiệu "Hỏi Dân IT" đang ngày càng khẳng định được vị thế của mình.
Nhiều bạn hỏi mình, sao mình không 'chặn việc mua lậu'. nói thật, nếu mình làm, là làm được đấy, cơ mà nó sẽ gây ra sự bất tiện cho học viên chân chính (con sâu làm rầu nồi canh). Với lại, ngay cả hệ điều hành windows, còn bị crack nữa là @@

Mình cũng có 1 bài post facebook về chuyện này:
https://www.facebook.com/askitwitheric/posts/pfbid02gyasktd3semgxat6nevnvwh4c8epzu3i7kpzhr7s7gmmfcvucyz96eb8avnvgwnhl
Với các bạn học viên chân chính, mình tin rằng, những cái các bạn nhận được từ mình khi đã chấp nhận đầu tư, nó sẽ hoàn toàn xứng đáng. vì đơn giản, với cá nhân mình, khách hàng là thượng đế.

VỚI CÁC BẠN MUA LẬU, MÌNH CHỈ MUỐN CHIA SẺ THẾ NÀY:

1. **TRÊN ĐỜI NÀY, CHẲNG CÓ GÌ CHẤT LƯỢNG MÀ MIỄN PHÍ CẢ.**
   VIỆC BẠN MUA LẬU QUA BÊN THỨ 3, LÀ GIÚP BỌN CHÚNG LÀM GIÀU VÀ GÂY THIỆT HẠI CHO TÁC GIẢ.
   NẾU NHÌN VỀ TƯƠNG LAI => CÀNG NGÀY CÀNG ÍT TÁC GIẢ LÀM KHÓA HỌC => NGƯỜI BỊ HẠI CUỐI CÙNG VẪN LÀ HỌC VIÊN

2. **HÃY HỌC THÓI QUEN TRÂN TRỌNG GIÁ TRỊ LAO ĐỘNG**
   NÓ LÀ THÓI QUEN, CŨNG NHƯ SẼ LÀ MỘT PHẦN TÍNH CÁCH CỦA BẠN.
   ĐỪNG VÌ NGHÈO QUÁ MÀ LÀM MẤT ĐI TÍNH CÁCH CỦA BẢN THÂN.
   NẾU KHÓ KHĂN, CỨ INBOX MÌNH, MÌNH HỖ TRỢ. VIỆC GÌ PHẢI LÀM VẬY =))

3. **MÌNH ĐÃ TỪNG LÀ SINH VIÊN GIỐNG BẠN**, MÌNH HIỂU TẠI SAO CÁC BẠN LÀM VẬY. **HÃY BIẾT CHO ĐI. SỐNG ÍCH KỶ, THÌ THEO LUẬT NHÂN QUẢ ĐẤY, CHẲNG CÓ GÌ LÀ NGẪU NHIÊN CẢ**

4. **NẾU BẠN THẤY KHÓA HỌC HAY, HÃY BIẾT DONATE ĐỂ ỦNG HỘ TÁC GIẢ. LINK DONATE: https://hoidanit.com.vn/donate**

Hành động nhỏ nhưng mang ý nghĩa lớn. Hãy vì 1 cộng đồng IT Việt Nam phát triển. Nếu làm như các bạn, có lẽ chúng ta đã không có Iphone, không có Apple như ngày nay rồi @@

#4. Về tác giả

Mọi thông tin về Tác giả Hỏi Dân IT, các bạn có thể tìm kiếm tại đây:

Website chính thức: https://hoidanit.com.vn/

Youtube "Hỏi Dân IT" : https://www.youtube.com/@hoidanit

Tiktok "Hỏi Dân IT" : https://www.tiktok.com/@hoidanit

Fanpage "Hỏi Dân IT" : https://www.facebook.com/askITwithERIC/

Udemy Hỏi Dân IT: https://www.udemy.com/user/eric-7039/

Chapter 1: Setup Environment
Cài đặt các công cụ cần thiết cho khóa học
#5. Công cụ code IDE
Cài đặt Visual Studio Code (VSCode):
Link download: https://code.visualstudio.com/download

- Setup VSCode: auto format on save

- Setup extension: (mục đích setup extension là "hỗ trợ" việc code, không phải là "phụ thuộc" vào extension)

=> chỉ setup extension "cần thiết", và hạn chế tối đa sự phụ thuộc vào extension

+ auto complete tag, auto close tag, auto rename tag
=> hỗ trợ việc code html nhanh hơn

+ code spell checker
=> hỗ trợ check tên biến/hàm có bị sai chính tả hay không ? (lưu ý là check tiếng anh, ko phải là check tiếng việt)

- Lưu ý về extension eslint, pretty (nếu cài: disabled)
Nếu bạn "từng học ở các nguồn khác", họ hướng dẫn cài đặt eslint, rồi pretty để "code cho đẹp hơn", thì vui lòng disabled những extension này, để tránh những "báo lỗi" không cần thiết.

Eslint, hay pretty, là cách bạn đặt ra quy tắc code, nhưng mà, mỗi cá nhân, mỗi công ty, có 1 quy định khác nhau

=> nếu bạn muốn setup eslint, hay pretty, thì setup trực tiếp trong project, không setup qua extension bạn nhé

Ở đây, chúng ta disabled, mục đích là dùng "những format" được quy định sẵn khi dùng VScode (không cần cài đặt extension).

#6. Browser

Về browser (trình duyệt web), bạn vui lòng tuân theo quy định bên dưới nhé :

1. Sử dụng Google Chrome (bắt buộc):

Download: https://www.google.com/chrome/

Lưu ý không dùng Cốc Cốc, Firefox hay Edge..., vì:

+ trong khóa học, mình sử dụng Google Chrome. Dùng giống nhau, sẽ có giao diện code giống nhau => hạn chế lỗi "không cần thiết"

+ **Google Chrome phổ biến nhất**, đi làm, chúng ta test cũng ưu tiên Google Chrome đầu tiên (cứ phổ biến mà xài)

Fact: Nếu bạn để ý, bạn sẽ thấy Firefox, hay Edge, đều bắt chước tính năng dành cho developer của Google Chrome :v

2. Chuyển ngôn ngữ Tiếng Anh (bắt buộc):

Bạn nào dùng ngôn ngữ tiếng việt => vui lòng chuyển qua tiếng anh, vì:

+ Giao diện ngôn ngữ là khác nhau. Các "thuật ngữ" ngành IT, dịch sang tiếng việt, nó không sát nghĩa (không đúng)

+ Đi làm thực tế, không ai dùng tiếng việt đâu bạn. sự thật đấy (nếu bạn không muốn được gọi là đứa nhà quê, hai lúa :v)

+ dùng tiếng anh, nó có lợi ích về lâu dài (giúp bạn đi nhanh hơn, vì tài liệu = tiếng anh luôn có sẵn)

+ Nếu bạn chưa biết gì về tiếng anh => dùng google translate.

Ở đây, chưa biết gì, thì code theo

code nhiều thành quen thôi. giống bạn chơi game ấy, toàn từ ngữ tiếng anh, sao bạn chơi được ? (học code nó tương tự bạn nhé)

3. Hướng dẫn chuyển đổi ngôn ngữ của Google Chrome

//todo (trong video đã hướng dẫn)

#7. Node.JS

1. Node.JS là gì ?

Bạn học React, tại sao lại cần Node.JS ?

Node.js là platform giúp bạn có thể thực thi ngôn ngữ Javascript trên máy tính (server)

Điều này tương tự với:
Bạn muốn học **Microsoft Words** => bạn cần cài đặt hệ điều hành **Windows**
(Bạn học **React** => cần cài đặt **Node.JS)**

như vậy, platform là "môi trường" giúp bạn chạy code.
chúng ta không học "node.js", mà học "react'.

Node.js là điều kiện "bắt buộc" để chúng ta chạy code các bạn nhé.

2. Cài đặt Node.JS

Lưu ý: không cài đặt version Node.js mới nhất, vì rất nhiều thư viện chưa kịp hỗ trợ Node.js latest

Việc cài chính xác 1 version của Node.js, sẽ giúp code của mình và bạn giống nhau 100% (hạn chế tối đa lỗi không cần thiết)

Điều này tương tự với việc, bạn "chơi 1 game trên windows 7 rất ok", nhưng điều này "không chắc chắn" khi bạn dùng windows 11, right ?

- Cài node v18.17.0: https://nodejs.org/download/release/v18.17.0/

- Muốn dùng nhiều version (tránh ảnh hưởng tới dự án cũ => nvm):
https://www.youtube.com/watch?v=ccjKHLyo4lM&list=PLncHg6Kn2JT6E38Z3kit9Hnif1xC_9VqI&index=40

3. Kiểm tra cài đặt Node.js

dùng câu lệnh: **node -v**

khi cài đặt node.js => đồng nghĩa với việc bạn cài đặt npm (node package manager) => công cụ giúp cài đặt thư viện code js
=> kiểm tra bằng cách: **npm -v**

#8. Quản lý code với Git

Nếu bạn chưa biết gì về Git, vui lòng xem nhanh tại đây:
https://www.youtube.com/playlist?list=PLncHg6Kn2JT6nWS9MRjSnt6Z-9Rj0pAlo

Mục đích sử dụng Git:

- "Kéo code" dự án thực hành

- Lưu trữ "backup" cho code. Tránh tình huống máy tính bị hư => mất hết code

- Git là "công cụ bắt buộc phải biết" khi đi làm thực tế (đi thực tập/fresher)

Chapter 2: Redux Overview

Góc nhìn tổng quan về thư viện Redux, giúp bạn trả lời câu hỏi What/Why/When to use Redux ?

#9. Redux là gì ?

1. What

- Redux là **1 thư viện javascript giúp quản lý/cập nhật "application state"** (tương tự như react state, app state chính là tất cả data muốn lưu trữ cho ứng dụng)

- Redux cung cấp pattern (quy luật/luật lệ) để code, thông qua sử dụng "events" (gọi là actions)
=> mô hình pub/sub (publisher/subscribers)

2. Why

- Redux giúp quản lý "global state - application state",
trả lời cho câu hỏi **when/where/why/how state đấy thay đổi/cập nhật ?**

- Code của redux có thể predictable/testable

3. When

Không phải tất cả ứng dụng dùng React, là "phải dùng Redux". Redux có hữu ích nhiều nhất khi:

- Có 1 lượng lớn data của "application state", cần được chia sẻ (sử dụng) tại nhiều nơi trong ứng dụng (dùng tại nhiều component)

- App state được "update" thường xuyên

- Logic để update "app state" dài dòng, phức tạp (nhiều code :v)

- App có kích thước vừa và lớn (có khối lượng code lớn), và team có nhiều người.

ví dụ minh họa về bài toán state này
- bảng giá chứng khoán
- //todo : add hình minh họa về share state between components

#10. Tại sao lại học Redux ?

- Đối với các dự án nhỏ (beginners), có lẽ bạn không cần dùng tới Redux
(vẫn code được đầy đủ tính năng).

- Khi đi làm tại các công ty, join các dự án lớn gồm nhiều người làm, có thể bạn sẽ bắt gặp "redux"

=> đây là lúc để học công nghệ mới, bạn vẫn có thể sử dụng Redux trong dự án nhỏ, chẳng qua là không dùng hết sức mạnh của nó

#11. Redux Die bởi React Context API

1. Phân biệt

- **Redux** là thư viện javascript, giúp quản lý "global state"
=> trả lời cho câu hỏi **when/where/why/how** state đấy thay đổi/cập nhật

- **React Context API** là tính năng tích hợp sẵn của React,
cung cấp phương thức "trông giống như Redux" (**useReducer**),
có nhiệm vụ passing (down) props từ cha xuống con

2. Redux có die ?

- Redux không die. Cách dễ nhất là xem lượng download từ npm
https://www.npmjs.com/package/react-redux

- **Redux và Context API phục vụ các mục đích khác nhau**:
Nếu bạn cần truyền props từ cha xuống con (mà không muốn viết quá nhiều code)
=> sử dụng Context api.
Context api giống như "thùng chứa data". bạn cần, bạn lấy ra dùng.

Bạn dùng Redux khi:
- cần tính năng như context api (thùng chứa data)

- muốn hiểu tại sao "state" update, nơi nào xảy ra (why/where/how)

- muốn có pattern để viết code (phong cách viết code). Code có thể testing được (kiểm thử)

#12. Lịch sử phát triển của Redux

Phân loại Redux:

1. Redux thuần

- Có thể sử dụng với **javascript**/typescript

- Sử dụng HOC (higher order component), với các function:

function **mapStateToProps**( )

function **mapDispatchToProps**( )

connect(mapStateToProps, mapDispatchToProps)(MyComponent)

=> đây là phong cách được sử dụng với React **class component**

2. Redux toolkit

- Có thể sử dụng với javascript/typescript

- Sử dụng hook (ứng với **react hook** thay vì class component)

- Các function nổi bật:
useDispath( )
useSelector( )

=> sử dụng cái này thay vì Redux thuần

3. RTK Query

- Quan tâm về **fetching/caching data**

- Tương tự như React Query, SWR..., tuy nhiên sử dụng cho Redux

Chapter 3: Redux Setup
Cài đặt và sử dụng Redux cho dự án React

#13. Run Hello World với React/Redux

Yêu cầu: đã cài đặt Git và có kiến thức cơ bản về React

Setup dự án:

Bước 1: Clone
https://gitlab.com/public-starter-projects1/02-redux-ultimate/react-redux-example

git clone ...

Bước 2: cài đặt thư viện cần thiết
npm i

Bước 3: chạy dự án
npm run dev

#14. Setup Redux Devtool
- cài đặt extensions

- redux toolkit đã tích hợp sẵn devtools => cài đặt extension cho google chrome là xong :v
- test ứng dụng redux :v

#15. Setup React với Vite
Tài liệu:
https://vitejs.dev/guide/#scaffolding-your-first-vite-project

Link source code: https://gitlab.com/public-starter-projects1/02-redux-ultimate/react-vite-starter

Lưu ý:
- sử dụng source code được cung cấp để đảm bảo hạn chế lỗi tối đa (môi trường code giữa mình và bạn là giống nhau)

- sử dụng typescript để được gợi ý code (nó không khó đến vậy :v)

#16. Cài đặt thư viện Redux

Cài đặt thư viện cần thiết
npm install --save-exact @reduxjs/toolkit@1.9.7 react-redux@8.1.3

@reduxjs/toolkit => đây là thư viện giúp tạo ra "global state" của redux

react-redux => sử dụng redux với react component

Chapter 4: Redux Concepts
Hiểu rõ và nắm vững các kiến thức cốt lõi của Redux

#17. Yêu cầu bài toán
- Sharing data between components:
+ không muốn lift-up state (passing props)
+ các components không có quan hệ với nhau

- Code có khả năng debug và testing

Lưu ý: chỉ làm tính năng tăng (increase), tính năng giảm (decrease dùng để thực hành)

#18. Mô hình Oneway Binding
Tài liệu: https://redux.js.org/tutorials/essentials/part-1-overview-concepts#redux-terms-and-concepts

Khái niệm: state, view, actions

state: trạng thái của component
view: html
actions: hành động của user

=> mô hình one-way data flow:
//todo: add image

hành động => state thay đổi => view thay đổi

view thay đổi (ví dụ typing/submit...) => trigger hành đồng => state thay đổi ....

#19. Các keywords sử dụng với Redux

- actions: là 1 biến object

- dispatch: công cụ thực hiện hành động

- reducer: công cu update store

- store (state)

slide (reducer + action)

#20. Redux Store

Redux Store là nơi lưu trữ "global data" (store lưu trữ state application)

Bước 1: Create a redux store
https://redux-toolkit.js.org/tutorials/quick-start#create-a-redux-store

=> với redux toolkit đã auto setup redux devtool

Bước 2: Provide the Redux Store to React
=> nói cho ứng dụng React biết sự tồn tại của Redux Store (không gian lưu trữ Redux)
https://redux-toolkit.js.org/tutorials/quick-start#provide-the-redux-store-to-react

Bước 3: Test với Redux dev tool => empty

#21. Redux Slide

Slide là công cụ giúp cập nhật Redux Store

Bước 1: Khai báo slide
https://redux-toolkit.js.org/tutorials/quick-start#create-a-redux-state-slice

Bước 2: Nạp slide vào Store thông qua reducer
https://redux-toolkit.js.org/tutorials/quick-start#add-slice-reducers-to-the-store

Bước 3: test với Redux dev tool => hiển thị data trong store

#22. useSelector - Read Redux's State
Tài liệu: https://redux-toolkit.js.org/tutorials/quick-start#use-redux-state-and-actions-in-react-components

Mục đích:
- Sử dụng data của Redux bên trong React Component

=> sử dụng useSelector hook
Lưu ý: thay đổi Redux State => component auto render (tương tự props)

#23. useDispatch - Update Redux's State

Yêu cầu: cập nhật Redux State khi người dùng thực hành động trên giao diện (onClick, onChange...)

Why? Redux state được cập nhật => giao diện auto cập nhật theo (do sử dụng useSelector)

#24. Tổng kết cách sử dụng Redux với Redux Toolkit (React)
Tài liệu: https://redux-toolkit.js.org/tutorials/quick-start

Bước 1: Setup Redux Store (nơi lưu trữ data của Redux)
- File store.ts
- configStore làm mọi thứ:
+ Nạp "reducer"
+ Cung cấp "default settings", ví dụ như redux dev tool

Bước 2: Cấu hình React "tiếp nhận" Redux Store
- Update ứng dụng React
+ Sử dụng <Provider> bọc ngoài <App/>
+ Cung cấp store: <Provider store={store}>

Bước 3: Tạo Redux "slide"

- Thông qua createSlide, gồm có:
+ "name" của slide
+ "initial state"
+ reducer function

=> Xuất ra "reducer" và "action"
- reducer dùng để khai báo tại file "store.ts"
- actions được dùng tại view (React component)

Bước 4: Sử dụng Redux bên trong ứng dụng

- Read data với **useSelector** hook

- Update data với :
+ **useDispatch** và dispatch(actions)

#25. Bài tập thực hành tính năng decrease

//todo

#26. Sử dụng useAppDispatch, useAppSelector với Typescript

=> code nhanh hơn

https://redux-toolkit.js.org/tutorials/typescript#project-setup

#27. Kỹ năng Debug Xem Code Chạy

Console.log

Debugger

đặt break point

Chapter 5: Redux Practises
Thực hành dự án với Redux

#28. Setup dự án Backend
sử dụng backend với json web server:
https://gitlab.com/public-starter-projects1/02-redux-ultimate/fake-backend-crud

- Gồm 2 endpoints:
+ /blogs
+ /users

- Backup database: backup_db.json

#29. Setup Bootstrap
Tài liệu:
https://react-bootstrap.netlify.app/docs/getting-started/introduction

Cài đặt:
npm i --save-exact react-bootstrap@2.9.1 bootstrap@5.3.2

```
import 'bootstrap/dist/css/bootstrap.min.css';

import Button from 'react-bootstrap/Button';
```

#30. Design Base giao diện
Link download src video này: https://drive.google.com/file/d/1OxCfy8uv-Eg9LYRrNrSJfa8dkV9DPLfo/view?usp=sharing

Chia layout, gồm có:
- navbar
- chia 2 tabs: blogs và users
- tạo table (hardcode data)

https://react-bootstrap.netlify.app/docs/components/navbar#text-and-non-nav-links

https://react-bootstrap.netlify.app/docs/components/tabs#examples

https://react-bootstrap.netlify.app/docs/components/table#example

#31. Get List Users với React

- fetching data với useEffect
- render data với table

#32. Redux Thunk
Tài liệu: https://redux.js.org/tutorials/essentials/part-5-async-logic#thunks-and-async-logic

#33. Fetch List User với Redux
Part 1:
//fetch data with thunk
https://redux-toolkit.js.org/usage/usage-guide#async-requests-with-createasyncthunk

//display data with useSelector

Part 2:
//todo

Part 3:
Giải thích code

https://redux-toolkit.js.org/api/createAsyncThunk

createAsyncThunk nhận 2 tham số đầu tiên:
https://redux-toolkit.js.org/api/createAsyncThunk#parameters

- actions type: string
- a promise function

```
const fetchUserById = createAsyncThunk(
  'users/fetchByIdStatus',
  async (userId: number, thunkAPI) => {
    const response = await userAPI.fetchById(userId)
    return response.data
  }
)
```

=> action: users/fetchByIdStatus
a promise function: async () => {.. }

1. Action type

Với "type" truyền vào, redux tự động generate 3 actions ứng với promise: pending, fulfilled, rejected

Ví dụ, với type = users/fetchByIdStatus, redux tự động sinh ra 3 actions tương ứng:
pending: 'users/requestStatus/pending'
fulfilled: 'users/requestStatus/fulfilled'
rejected: 'users/requestStatus/rejected'

2. PayloadCreator
https://redux-toolkit.js.org/api/createAsyncThunk#payloadcreator

PayloadCreator gồm 2 tham số (arg, thunkAPI)
ví dụ: async (userId: number, thunkAPI) => {...}

dispatch(fetchUsers({status: 'active', sortBy: 'name'})).

3. extraReducers

https://redux-toolkit.js.org/api/createSlice#extrareducers

Logic Action định nghĩa ngoài reducer của slide => viết tại extraReducer
ví dụ:
+ dùng async logic với thunk
+ dùng các action tại các slide khác

```
  builder.addCase(fetchUserById.fulfilled, (state, action) => {
    // Add user to the state array
    state.entities.push(action.payload)
  })
```

#34. Notification với React Toastify
//setup react toastify
npm i --save-exact react-toastify@9.1.3
https://fkhadra.github.io/react-toastify/introduction/

https://fkhadra.github.io/react-toastify/installation

Yêu cầu: setup global container => test xem hoạt động không

#35. Bài Tập Design Create/Update/Delete User
Tài liệu:
https://react-bootstrap.netlify.app/docs/components/buttons
https://react-bootstrap.netlify.app/docs/components/modal#vertically-centered
https://react-bootstrap.netlify.app/docs/forms/floating-labels
Source code video này:
https://drive.google.com/file/d/1miYlWFlQ9zuqbyM58KqnWBje5QCFVwye/view?usp=sharing

Lưu ý: nếu bạn "không thể" design được giao diện, nên cân nhắc việc dừng lại, học react cơ bản.

Vì: redux không liên quan tới react (phần design giao diện). Redux là phần mở rộng của React. Nếu kiến thức cơ bản "chưa nắm vững"
=> học mở rộng chỉ tốn time vô ích. Sự thật đấy.

Với typescript, bạn có thể dùng (props: any) để bỏ đi check type

Yêu cầu:
- thêm 3 buttons: Create/Edit/Delete
- Không cần truyền logic xử lý xuống modal, chỉ cần truyền props đóng/mở. Vì logic xử lý sẽ được làm với Redux

1. Tính năng create a new user
- Nhấn vào button create new' => mở ra modal create new

- thông tin trên modal gồm có: email, name

- khi nhấn submit, lấy được thông tin của người dùng nhập vào (console.log)

2. Tính năng edit a user
- Nhấn vào button 'edit' => mở ra modal edit

- Thông tin user được fill sẵn vào modal (không hiển thị id, chỉ hiển thị những trường cho phép edit)

- khi nhấn submit, cần lấy được thông tin người dùng edit, và "id của user" (console.log)

3. Tính năng delete a user
- Nhấn vào button 'delete' => mở ra modal delete
- khi nhấn submit => cần lấy được id của user (console.log)

#36. Create a New User với Redux

//if success => refetch data
thunkAPI.dispatch(actionLoading());

Tài liệu:
https://redux-toolkit.js.org/api/createAsyncThunk#payloadcreator

#37. Bài tập Update a User với Redux

Các bước đã làm:

Bước 1: Từ React dispatch Redux action (kèm theo data -> payload)

Bước 2: Tại Redux, viết logic gọi API (thunk)

Bước 3: Sau khi gọi API thành công:

- dispatch action để fetch lại data

- cập nhật trạng thái redux (gọi là trạng thái A). Mục đích cập nhật trạng thái này, là để cho React biết "sự thay đổi"

Bước 4: Tại React

- Lắng nghe sự thay đổi của "trạng thái A"
- Hiển thị kết quả tương ứng với "trạng thái A"

API update user:

```
const res = await fetch(`http://localhost:8000/users/${payload.id}`, {
        method: "PUT",
        body: JSON.stringify({
          email: payload.email,
          name: payload.name,
        }),
        headers: {
          "Content-Type": " application/json"
        }
      });
```

Source code video này:
https://drive.google.com/file/d/10i4CMpZqYesVN8yxmDak6v9XDJXMhFF8/view?usp=sharing

#38. Bài tập Delete a user với Redux

```
const res = await fetch(`http://localhost:8000/users/${payload.id}`, {
        method: "DELETE",
        headers: {
           "Content-Type": " application/json"
        }
    });
```

Source code video này:
https://drive.google.com/file/d/1FjaeEXMAgBjurX1Vm37wOdJEGeDtBM8s/view?usp=sharing

#39. Hỗ trợ dark/light mode
https://getbootstrap.com/docs/5.3/customize/color-modes/

Part 1: Quản lý data với local state
Design Header:
https://react-bootstrap.netlify.app/docs/forms/checks-radios/#switches

để id => click label

```
import Form from 'react-bootstrap/Form';

<Form.Check
                    type="switch"
                    id="custom-switch"
                    label="Light mode"
                  />
```

//todo: add local state

Sau khi add local state => data-bs-theme cho Navbar

Vấn đề tồn đọng (set mode data cho toàn bộ app), hiện tại chỉ set được cho header ?

Part 2: Sử dụng Redux

có thể thêm data-bs-theme vào tag <html> or tag <body> => nằm ngoài div root

--bs-body-bg vào tag <body>

Bước 1: Tạo appReducer

Bước 2: lưu thông tin light/dark vào redux

Bước 3: Tạo actions tương ứng

Bước 4: thay thế local state = redux

Bước 5: Sử dụng Javascript để thay đổi tag body
https://stackoverflow.com/a/63087710

```
useEffect(() => {
    const body = document.querySelector("body");
    if (body) body.setAttribute('data-bs-theme', mode);
}, [mode])
```

Vấn đề tồn động : F5 (refresh) => bị mất dark mode

#40. Redux persist

Why not Local Storage ? Bởi vì khi localStorage thay đổi => component ko re-render

Bản chất của thư viện Redux Persist:

Khi user f5 website:

- Đọc dữ liệu từ local storage
- Nạp dữ liệu vào redux
- Cho App render bình thường (sau khi quá trình nạp dữ liệu hoàn tất)

https://redux-toolkit.js.org/usage/usage-with-typescript#getting-the-state-type

https://redux-toolkit.js.org/usage/usage-guide#use-with-redux-persist

https://www.npmjs.com/package/redux-persist

npm i --save-exact redux-persist@6.0.0

hàm configureStore, ngoài config rootReducer => còn setup các tham số mặc định, ví dụ như devTool, thunk middleware...

Source code video này:
https://drive.google.com/file/d/1qi0usF-6f2wMXO89P6VEWyTywu9bQ5CJ/view?usp=sharing

#41. Bài tập CRUD blogs với Redux

Source code video này:
https://drive.google.com/file/d/1eD0qRTm0vXT55NQuAAO30EZxmtGow07X/view?usp=sharing

Các APIs sử dụng

1. Lấy tất cả blogs:

GET /blogs

2. Lấy blog theo id

GET /blogs/:id

ví dụ: GET /blogs/1

3. Tạo mới 1 blog

POST /blogs

body truyền lên object { title, author, content}

4. Cập nhật 1 blog

PUT /blogs/:id

body truyền lên object { title, author, content}

ví dụ: PUT /blogs/1

5. Xóa 1 blog

DELETE /blogs/:id

ví dụ: DELETE /blogs/1

#42. Tổng kết các kiến thức Redux đã học

1. Các kiến thức đã học

- Mô hình hoạt động của Redux:
Từ View (React) -> dispatch (**actions**). Action này được viết trong Reducer (**slide**) -> thay đổi state của Redux

State của Redux thay đổi, nơi nào sử dụng state này, sẽ tự động bị "re-render" (thông qua useAppSelector)

- Cấu hình Redux **Slide** :actions + reducer
- Cấu hình Redux Store
- Cấu hình Redux Persist : lưu data vào local Storage

=>
- Code cho chạy được ?
- Các khóa học khác của Hỏi dân IT là đã đủ dùng

2. Các kiến thức chưa học

Thực tế, mỗi công ty 1 yêu cầu => want more (dev như là siêu nhân)

- Fetching data (handle loading, error) => khóa react test fresher

- Handle Caching data ?

Chapter 6: React Query
Quản lý "server state" với React Query

#43. Vấn đề tồn đọng
1. Chuyện code
Code logic CRUD dài
Lập trình viên thường lười => muốn viết ngắn gọn

2. So sánh React Query và Redux
Về React Query: https://tanstack.com/query/v5/docs/react/overview

- **React Query để quản lý 'server state'**
=> fetching/caching data và "làm ngắn gọn cú pháp code"

- **Redux để quản lý 'client state'**.
Nếu muốn fetching/caching data -> **redux toolkit query (RTQ)** -> cú pháp phức tạp hơn react query

- Vấn đề tồn đọng của fetching data: cache + handle loading/error
Phân biệt redux (redux toolkit query) và react-query
https://tanstack.com/query/v5/docs/react/comparison

+ caching data with redux toolkit query
https://redux-toolkit.js.org/rtk-query/usage/examples

#44. Cài đặt React Query

1. Setup new project

Bước 1: Clone
https://gitlab.com/public-starter-projects1/02-redux-ultimate/react-query-starter

Bước 2: Run dự án
npm i
npm run dev

Mặc định, project sẽ chạy tại: localhost:5173

Các việc đã làm đối với dự án mẫu:
- Setup Bootstrap, React Toastify
- Không sử dụng Redux (giảm độ phức tạp cho dự án)
- Design base giao diện CRUD (chưa tích hợp API, đang hardcode dữ liệu)

2. Setup React Query
https://www.npmjs.com/package/@tanstack/react-query

Cài đặt:
npm i --save-exact @tanstack/react-query@5.4.3

Lưu ý: với các version mới (v4, v5...) => @tanstack/react-query@5.4.3
với version cũ <= v3: https://www.npmjs.com/package/react-query

3. Add delay ở backend
//todo: update file server.js của backend

#45. Fetch Data

Handle Loading/Error State

Với async logic, chúng ta luôn luôn tốn time để chờ đợi api làm nhiệm vụ

=> add loading state để tăng trải nghiệm của người dùng

Tài liệu: https://tanstack.com/query/v5/docs/react/overview#enough-talk-show-me-some-code-already

Bước 1: Setup Client

=> Nói cho React biết sự tồn tại của 'React Query' (tương tự Redux)

<QueryClientProvider client={queryClient}/>

Bước 2: Handle Fetch data

```
const { isPending, error, data: users } = useQuery({
      queryKey: ['fetch-users'],
      queryFn: () =>
        fetch('http://localhost:8000/users').then(
          (res) => res.json(),
        ),
  })

 if (isPending) return 'Loading...'

 if (error) return 'An error has occurred: ' + error.message
```

#46. useQuery Hook

- Sử dụng useQuery để fetch data

Dạng đơn giản nhất, bao gồm 2 thành phần, là query-key và query-function

1. Query keys

https://tanstack.com/query/v5/docs/react/guides/query-keys

- unique
- tương tự như id

2. Query function

https://tanstack.com/query/v5/docs/react/guides/query-functions

```
//dùng với typescript
    queryFn: (): Promise<IUser[]> =>
        fetch('http://localhost:8000/users').then(
            (res) => res.json(),
        ),
```

#47. React Query Devtool
Tài liệu:
https://tanstack.com/query/v5/docs/react/devtools

Cài đặt:
npm i --save-exact @tanstack/react-query-devtools@5.4.3

Mặc định, react-query-devtool chỉ bao gồm trong "bundle" với node_env = "development"

#48. Query parameter
https://tanstack.com/query/v5/docs/react/guides/query-keys#if-your-query-function-depends-on-a-variable-include-it-in-your-query-key

https://github.com/typicode/json-server#plural-routes

API get user by id: http://localhost:8000/users/:id

#49. Khái niệm Stale & Cache

Tài liệu:
https://tanstack.com/query/v5/docs/react/overview#motivation

1.Trạng thái của data

Data thường được lưu trữ tại backend (dưới database) và frontend "không kiểm soát data"
=> không biết data lưu trữ như thế nào và nó chứa gì, phụ thuộc hoàn toàn vào backend để có data

Có 2 trạng thái chính: mới (new) và cũ (old).
Tại thời điểm A, frontend đang sở hữu **data A mới nhất.**
tuy nhiên, tại thời điểm A + 1, backend thực hiện thao tác Create/Update/Delete
=> data A ấy (thứ frontend đang sở hữu) trở thành **data cũ**

Với fetching data, chúng ta dùng 2 từ chuyên nghiệp: **fresh** (= new) **và stale** (= old)

Stale: ám chỉ data "đã bị cũ"

Fresh: data mới nhất

//todo: thay đổi **tham số staleTime**
staleTime: 30 * 1000 //30s

2. Khái niệm cache

Cách dễ nhất để "tăng hiệu năng", là **chỉ gọi backend "khi thực sự cần thiết"**

Ví dụ: nếu đã gọi data rồi, đã có kết quả rồi, thì sẽ không gọi lại data đấy nữa
=> khái niệm cache ra đời

Với React Query, nó sẽ tự động tạo cache để lưu trữ data, hạn chế tối đa việc gọi lại backend khi không cần thiết

Minh họa: https://stackoverflow.com/a/74557893

#50. Design Pagination

Tài liệu:
https://react-bootstrap.netlify.app/docs/components/pagination

API phân trang:
https://github.com/typicode/json-server#paginate

Ví dụ: http://localhost:8000/users?_page=1&_limit=1

Cần truyền vào header 2 tham số:
_page: trang kết quả muốn lấy
_limit: số lượng bản ghi tối đa/lần

#51. Phân trang với React Query

Tài liệu:
https://tanstack.com/query/v5/docs/react/guides/paginated-queries

```
 const total_items = +(res.headers?.get("X-Total-Count") ?? 0)
 const page_size = PAGE_SIZE;
 const total_pages = total_items == 0 ? 0 : (total_items - 1) / page_size + 1;
setTotalPages(total_pages)
```

Source code video này:
https://drive.google.com/file/d/1J7gkcmXnPUUv4oAFcwSJFI7TfHhfODYk/view?usp=sharing

#52. Mutation Data

Với CRUD:

useQuery: fetching data (Read data)
https://tanstack.com/query/v5/docs/react/guides/queries

useMutation: mutate data (Create/Update/Delete data)
https://tanstack.com/query/v5/docs/react/guides/mutations

API create new user:

```
const res = await fetch("http://localhost:8000/users", {
        method: "POST",
        body: JSON.stringify({
            email: payload.email,
            name: payload.name
        }),
        headers: {
            "Content-Type": " application/json"
        }
    });
```

#53. Create New User

Tài liệu: https://tanstack.com/query/v5/docs/react/guides/mutations
https://react-bootstrap.netlify.app/docs/components/spinners/#buttons

```
const calculatePagesCount = (pageSize, totalCount) => {
   // we suppose that if we have 0 items we want 1 empty page
   return totalCount < pageSize ? 1 : Math.ceil(totalCount / pageSize);
};


// Example usage:

const pageSize = 10;
const totalCount = 21;
const pagesCount = calculatePagesCount(pageSize, totalCount)
```

#54. Revalidate Data

Tài liệu: https://tanstack.com/query/v5/docs/react/guides/invalidations-from-mutations

#55. Bài tập Update User

API Update User:

```
const res = await fetch(`http://localhost:8000/users/${payload.id}`, {
        method: "PUT",
        body: JSON.stringify({
          email: payload.email,
          name: payload.name,
        }),
        headers: {
          "Content-Type": " application/json"
        }
      });
```

Source code video này:
https://drive.google.com/file/d/16a6C-DTrLxvcIOMzEl2bSrIlMNSpmHSw/view?usp=sharing

#56. Bài tập Delete User

API Delete User:

```
const res = await fetch(`http://localhost:8000/users/${payload.id}`, {
        method: "DELETE",
        headers: {
          "Content-Type": " application/json"
        }
      });
```

Source code video này:
https://drive.google.com/file/d/1xV4Z014K80lGRRpxTPLvyWUmJdJ47Jcu/view?usp=sharing

#57. Sharing Data Between Components

lấy list user from cache => hiển thị số lượng ở header và tab blogs

Tài liệu:
https://tanstack.com/query/v5/docs/react/reference/QueryClient#queryclientgetquerydata

https://stackoverflow.com/a/68337967

#58. Bài tập CRUD Blogs

Source code : checkout branch: **final**
https://gitlab.com/public-starter-projects1/02-redux-ultimate/react-query-starter/-/tree/final?ref_type=heads

Các APIs sử dụng
1. Lấy tất cả blogs:
GET /blogs

2. Lấy blog theo id
GET /blogs/:id

ví dụ: GET /blogs/1

3. Tạo mới 1 blog
POST /blogs
body truyền lên object { title, author, content}

4. Cập nhật 1 blog
PUT /blogs/:id
body truyền lên object { title, author, content}

ví dụ: PUT /blogs/1

5. Xóa 1 blog
DELETE /blogs/:id

ví dụ: DELETE /blogs/1

#59. So sánh Redux và React Query

Cả Redux và React Query đều có thể:
- Fetching data, thực hiện CRUD
- Sharing data between component

Lợi thế của React Query, là code ngắn hơn, thư viện này cung cấp "magic" code, phần khó thư viện đã làm hết.

Vậy câu hỏi đặt ra là, React Query có thay thế Redux ?

=> **không thay thế.** Vì nếu Redux bị thay thế, thì số lượng download trên npm sẽ không khủng như vậy :v

Chi tiết: https://tanstack.com/query/v5/docs/react/guides/does-this-replace-client-state

1. Khi nào nên sử dụng React Query
- Muốn viết code ngắn
- Muốn cache data
- Muốn quản lý "server state", tức là **combo fetch + cache data từ backend**

những điểm không nên sử dụng React Query, chính là việc sử dụng các thư viện khác, ví dụ Redux:
- Muốn viết code theo "hình thức/pattern" nhất định
- Muốn data là mới nhất (chưa quan tâm tới cache). Điều này "rất quan trọng" trong ứng dụng real-time
- Muốn quản lý "client state"

React Query là con dao 2 lưỡi. Dùng cache là tốt, tuy nhiên, nếu data thay đổi thường xuyên => cache không có tác dụng => dùng Redux :v
và, khi sử dụng cache, cần hiểu rõ cơ chế hoạt động của nó :v

2. Các ứng dụng trong thực tế

Ứng dụng yêu cầu:
+ code đơn giản (ngắn)
+ cache data

=> sử dụng React Query

Ứng dụng yêu cầu:
+ code theo pattern
+ data thay đổi thường xuyên (real-time)
=> sử dụng Redux

Ngoài ra, bạn vẫn có thể sử dụng song song React Query và Redux trong cùng 1 ứng dụng mà không lo bị mâu thuẫn :v

#60. Các tham số mặc định của React Query
Tài liệu:
https://tanstack.com/query/v5/docs/react/guides/important-defaults

Một vài tham số mặc định:
stateTime = 0
gcTime = 300000 (5 phút)

#61. Reuse React Query
Tài liệu: https://tkdodo.eu/blog/practical-react-query

- Reuse Hook:
https://tkdodo.eu/blog/react-query-as-a-state-manager
Fix bugs ở header và phân trang

- Reuse key
https://tkdodo.eu/blog/effective-react-query-keys#use-query-key-factories

Download source code video này:
https://drive.google.com/file/d/1Kf8E3GV_OPUGgezqcGH4B0hd56OK23Oh/view?usp=sharing

#62. Tổng kết về React Query

1.Các kiến thức đã học

- Fetching data với **useQuery**

- Create/Update/Delete với **useMutation,** đồng thời **revalidate data**

Tham khảo tài liệu: https://tkdodo.eu/blog/practical-react-query

2.Các kiến thức chưa học

Tham khảo:
https://tanstack.com/query/v5/docs/react/guides/important-defaults

Chapter 7: Redux/React Query với Nextjs
Sử dụng Redux và React Query với framework Nextjs

#63. Sơ lược về Next.js

1. Tổng quan

- Next.js là framework tích hợp sẵn React, giúp việc phát triển website trở nên dễ dàng hơn.
- Next.js chuyên làm frontend (mặc dù "có thể" làm được backend)
- Được team React gợi ý nên dùng cho dự án thực tế:
https://react.dev/learn/start-a-new-react-project#production-grade-react-frameworks

Chi tiết về Next.js, tham khảo:
https://hoidanit.com.vn/khoa-hoc/react-pro-max-voi-nextjs-lam-chu-toan-dien-reactjs-hien-dai-65198100e6bafa8caad417a6.html

2. Tại sao dùng Next.js ?

- Hỗ trợ SEO
- Render React trên server (server component) => giúp load page nhanh hơn
- Và nhiều điều Next.js làm sẵn, ví dụ Routing, Images...

=> xuất hiện nhu cầu sử dụng Redux, React Query ... (các thư viện hay dùng với React 'client') với framework Next.js

=> câu hỏi đặt ra: tích hợp như thế nào ?

#64. Setup Redux Toolkit cho Next.js
https://redux-toolkit.js.org/introduction/getting-started#create-a-react-redux-app

https://download-directory.github.io/

Lưu ý: không nên dùng nextjs:latest

//todo: giải thích code cách setup dự án Redux cho Next.js

#65. Vai trò của Redux với Next.js

1. Khái niệm server và client

Server : code chạy tại máy chủ server, ví dụ như VPS (nơi hosting code backend của bạn)

Client: chính là browser duyệt web của bạn

với Next.js <=12, có thể sử dụng package sau để hạn chế bugs:
https://github.com/kirill-konshin/next-redux-wrapper

Tuy nhiên, từ Next.js 13 trở đi => package trên ném vào sọt rác, khi xuất hiện khái niệm "server component" :v
https://github.com/kirill-konshin/next-redux-wrapper/issues/494#issuecomment-1366851492

Nhắc về các mốc thời gian quan trọng:
Từ năm 2011 tới tháng 4/2022: trending là CSR (client side rendering)

Từ tháng 4/2022 -> nay, react 18 ra đời hỗ trợ mạnh mẽ cho việc render react ở Server => trending là Nex.tjs

Redux được sinh ra vào 2015 => phục vụ rất tốt cho CSR (quản lý client state), không phải là SSR (server state)

2. Có nên dùng Redux với Nextjs không ?

Tương tự như câu hỏi là có nên dùng thư viện Redux không => câu trả lời là phụ thuộc vào nhu cầu của bạn.

Tuy nhiên, **với Next.js, đa phần câu trả lời sẽ là KHÔNG**, chỉ có số ít trường hợp là nên dùng.

Các lý do bạn nên cân nhắc:
- Nên cân nhắc react Context API (với small data)

- Do nextjs là server => có thể sử dụng session, ví dụ như khi sử dụng next-auth

- persist data => not work well. chỉ sử dụng ở Client Component (dùng thêm tag 'use client')

=> Redux vốn dĩ sinh ra là quản lý "client state" => not for "server state".

Chỉ nên dùng Redux cho Next.js khi:
- Bạn không quan tâm nhiều về "caching data"
- Dữ liệu website thay đổi liên tục "real-time" => Next.js sẽ sử dụng client-component

#66. Sử dụng React Query với Next.js

Mục đích bạn dùng React Query:
- Code viết ngắn hơn
- Cache data

=> điều này đúng cho Next.js <= 12

Với Next.js >=13, Next.js "built-in"/tích hợp sẵn việc caching data với fetch

=> các thư viện như React-Query/SWR... "không cần thiết" phải dùng với Next.js.

Bạn vẫn có thể sử dụng, tuy nhiên, cần phải cấu hình (phức tạp/có bugs) và sử dụng "client component"
=> không dùng được với "server component"

=> không nhất thiết dùng react query (hoặc redux toolkit query) với Nextjs
https://tkdodo.eu/blog/you-might-not-need-react-query#you-might-not-need-it

Fact:
Chi tiết về Next.js, tham khảo:
https://hoidanit.com.vn/khoa-hoc/react-pro-max-voi-nextjs-lam-chu-toan-dien-reactjs-hien-dai-65198100e6bafa8caad417a6.html

Nextjs có các tính năng nổi bật tích hợp sẵn:
- caching data (khi fetching)

- revalidate data (khi create/update/delete)

Chapter 8: Redux Saga
Làm quen về Redux Saga Middleware

#67. Yêu cầu trước khi học Redux Saga

- Cần có hiểu biết về Redux (recommend Redux Toolkit) (**required**)
- Cần có hiểu biết về middleware (ví dụ redux thunk)

Trong đa số các trường hợp, **mình recommend redux-thunk**.

Nếu khi nào mà redux-thunk không đáp ứng được nhu cầu của bạn => chọn saga

Ở đây là học cho biết (phòng trường hợp sau này đi làm công ty nó dùng). biết (hiểu) thì dễ chém gió :v

Hiểu đơn giản về Saga:

- **Là Redux middleware** => bạn cần biết Redux trước khi học saga
- Đã hiểu middleware => xử lý logic bất đồng bộ

- Redux Saga sử dụng **Javascript Generator Functions.**

#68. Javascript Generator

Tài liệu:

- The "yield" operator: https://developer.mozilla.org/en-US/docs/Web/JavaScript/Reference/Operators/yield

- function* : https://developer.mozilla.org/en-US/docs/Web/JavaScript/Reference/Statements/function*

- Generator: https://developer.mozilla.org/en-US/docs/Web/JavaScript/Reference/Global_Objects/Generator

https://www.digitalocean.com/community/tutorials/understanding-generators-in-javascript

1. Generator Function

https://developer.mozilla.org/en-US/docs/Web/JavaScript/Reference/Statements/function*

Generator Function là hàm có dấu * bên cạnh function
=> mục để là tạo ra Generator Object

Cú pháp:

```
function* name(param0) {
  statements
}
function* name(param0, param1) {
  statements
}
function* name(param0, param1, /* …, */ paramN) {
  statements
}
```

hoặc

```
const my_var = function* (param0, param1, /* …, */ paramN) {
  statements
}
```

Lưu ý: viết function truyền thống và KHÔNG SỬ DỤNG arrow function

https://stackoverflow.com/questions/27661306/can-i-use-es6s-arrow-function-syntax-with-generators-arrow-notation

2. Generator
https://developer.mozilla.org/en-US/docs/Web/JavaScript/Reference/Global_Objects/Generator

Generator object được trả về bởi generator function.
Mục đích:
Generator function tạo ra Generator object

Chúng ta sử dụng object này để có thể "truy cập" vào function (pause, resume...)

===
Với function truyền thống:
function sum(a,b) { return a + b}
sum(6, 9) // calling function = 15
//nếu ko có keyword return , nó sẽ trả ra undefined

Với Generator function
// Declare a generator function with a single return value
function* generatorFunction() {
  return 'Hello, Generator!'
}
// Assign the Generator object to generator
const generator = generatorFunction();
console.log(generator)

Output
generatorFunction {<suspended>}
  __proto__: Generator
  [[GeneratorLocation]]: VM272:1
  [[GeneratorStatus]]: "suspended"
  [[GeneratorFunction]]: ƒ* generatorFunction()
  [[GeneratorReceiver]]: Window
  [[Scopes]]: Scopes[3]

=> generator không trả ra kết quả ngay lập tức, thay vì vậy, trả ra Generator object

(ban đầu trạng thái là suspended)

Generator là một iterator:
https://developer.mozilla.org/en-US/docs/Web/JavaScript/Reference/Iteration_protocols#the_iterator_protocol

=> có method next(), thứ được dùng trong vòng lặp

// Call the next method on the Generator object
generator.next()

Output
{value: "Hello, Generator!", done: true}

console.log(generator)
Output
generatorFunction {<closed>}
=> vì có keyword return

==========
Làm sao để có thể pause và resume a function
3. yield Operators
https://www.digitalocean.com/community/tutorials/understanding-generators-in-javascript#yield-operators

#69. Setup project

1. Clone base dự án

https://gitlab.com/public-starter-projects1/02-redux-ultimate/redux-saga-starter

Lưu ý: thêm action "decrease" để test saga

2. Giải thích source code đã cài đặt

//todo

#70. Add redux saga

Source code video này:
https://gitlab.com/public-starter-projects1/02-redux-ultimate/redux-saga-starter/-/commit/0e7ee58581e4852cd4ea5cda880701726bb26044

Cài đặt:
npm i --save-exact redux-saga@1.2.3

Bước 1: Add saga middleware

```
import RootSaga from "../saga/root.saga";

// Create the saga middleware
const sagaMiddleware = createSagaMiddleware( )

export const store = configureStore({
   reducer: {
      counter: counterReducer,
   },
   middleware: (getDefaultMiddleware) =>
getDefaultMiddleware().concat(sagaMiddleware),
})

sagaMiddleware.run(RootSaga);
```

Bước 2: Add root saga

```
export default function* RootSaga() {
   console.log(">>> root saga")
}
```

#71. Saga Effects
https://redux-saga.js.org/docs/basics/DeclarativeEffects/

1. Nhắc lại Generator function
function* myGeneratorFunction( ){
yield...
}

Generator function cho phép chúng ta kiểm soát function đấy: **stop** (dừng lại), **resume** (tiếp tục chạy), **destroy**/cancel
yield là công cụ để làm điều trên.

Về bản chất, saga là middleware "gián tiếp làm hộ" chúng ta để thao tác với function trên.
Saga sẽ dùng các method của **generator object**, ví dụ:
generator.next()
generator.return()
generator.throw()

2. Khái niệm Effect
Với javascript, chúng ta có khái niệm "**side effect**" (nếu bạn dùng thuốc quá liều, chúng ta bị tác dụng phụ/side effect)
```
const isImpure = () => {
  return Math.random();
};
```
pure function/impure function

Với React, chúng ta có hook useEffect() :v
Effect ám chỉ hành động chúng ta thực thi 1 functions, và functions đấy không đoán trước được kết quả.
Ví dụ: async/await khi gọi API
=> đấy là lý do người ta gọi API trong useEffect :v

===

Với Saga, Effect là 1 object. Object này chứa thông tin để "ra lệnh" cho saga middleware thực hiện.
Ví dụ: thực hiện async function, dispatch action...

Để tạo ra Effects, chúng ta sử dụng các functions được cung cấp bởi redux-saga/effects

ví dụ:
```
import { takeEvery } from 'redux-saga/effects'
import Api from './path/to/api'

function* watchFetchProducts() {
  yield takeEvery('PRODUCTS_REQUESTED', fetchProducts)
}

function* fetchProducts() {
  const products = yield Api.fetch('/products')
  console.log(products)
}
```

#72. Root Saga
https://redux-saga.js.org/docs/advanced/RootSaga

Lắng nghe 1 actions cụ thể

```
yield takeEvery("*", sagaFunction)
```

```
///
import { PayloadAction } from '@reduxjs/toolkit';
import { takeEvery } from 'redux-saga/effects';

function* fetchData(action: PayloadAction) {
   console.log(">>> check action", action)
}


export default function* RootSaga() {
   console.log(">>> root saga")
   yield takeEvery("counter/decrement", fetchData)
}
```

//increasement().type
#73. Ví dụ increase button

Saga là middleware, và xử lý async actions
=> để xử các hành động như increase/decrease (không phải async) thì cần làm 2 bước:

- **Bước 1:** ACTION_START (saga lắng nghe)

- **Bước 2:** saga xử lý, và dispatch ACTION_FINISH

sau khi nhận được action từ saga, redux sẽ làm việc còn lại.

- increase with saga
- phân biệt takeEvery và takeLatest (tương tự cho takeLeading)
dùng yield delay() để add delay

#74. Bài tập về decrease button

- Làm tương tự increase button
- gồm 2 action:
+ init decrease
+ decrease success

#75. createAction

https://redux-toolkit.js.org/api/createAction#usage-with-createreducer

Mục đích: tạo actions để reuse giữa redux toolkit và redux saga

#76. Hiển thị list user

Part 1: Định nghĩa actions

Phân tích actions:
- GET_LIST_USER_START
- GET_LIST_USER_PENDING
- GET_LIST_USER_SUCCESS
- GET_LIST_USER_FAILED

```
const initialState = {
  isPending: false,
  isError: false,
  data: [ ],
  errors: [ ],
}
```

- Các effect cần sử dụng khi fetch users:
+ send request (call)
https://redux-saga.js.org/docs/api/#callfn-args

Dùng call thay vì fetch api trực tiếp: https://redux-saga.js.org/docs/basics/DeclarativeEffects

+ dispatch action (put)
https://redux-saga.js.org/docs/api/#putaction

Part 2:
Source code video này:
https://gitlab.com/public-starter-projects1/02-redux-ultimate/redux-saga-starter/-/commit/6e69ff9ac2533923acae7e6069724b362ad797d7

#77. Thêm mới user
Source code video này:
https://gitlab.com/public-starter-projects1/02-redux-ultimate/redux-saga-starter/-/commit/81307a14a1c20dbd044983ac9ea6fd9e860b9611

API create new user:

```
const res = await fetch("http://localhost:8000/users", {
        method: "POST",
        body: JSON.stringify({
          email: payload.email,
          name: payload.name
        }),
        headers: {
          "Content-Type": " application/json"
        }
      });
```

#78. Bài tập Update User
Source code video này:
https://gitlab.com/public-starter-projects1/02-redux-ultimate/redux-saga-starter/-/commit/29bf803ddeeaeb8b447ab91e56fb2ebbdc768649

API Update user:

```
const res = await fetch(`http://localhost:8000/users/${payload.id}`, {
        method: "PUT",
        body: JSON.stringify({
          email: payload.email,
          name: payload.name,
        }),
        headers: {
          "Content-Type": " application/json"
        }
      });
```

#79. Bài tập Delete User
Source code video này:
https://gitlab.com/public-starter-projects1/02-redux-ultimate/redux-saga-starter/-/commit/29bf803ddeeaeb8b447ab91e56fb2ebbdc768649

API xóa user:

```
const res = await fetch(`http://localhost:8000/users/${payload.id}`, {
        method: "DELETE",
        headers: {
          "Content-Type": " application/json"
        }
      });
```

#80. Bài tập crud blogs

Source code video này:
https://gitlab.com/public-starter-projects1/02-redux-ultimate/redux-saga-starter/-/commit/fc0c2225fcc1f74b461e144e2a10abda99dfbff3

```
const fetchBlogs = async () => {
   const res = await fetch("http://localhost:8000/blogs");
   return res.json();
}

const createBlog = async (title: string, author: string, content: string) => {
   const res = await fetch("http://localhost:8000/blogs", {
      method: "POST",
      body: JSON.stringify({
         title: title,
         author: author,
         content: content
      }),
      headers: {
         "Content-Type": " application/json"
      }
   });
   return res.json()
}

const updateBlog = async (id: number, title: string, author: string, content: string) => {
   const res = await fetch(`http://localhost:8000/blogs/${id}`, {
      method: "PUT",
      body: JSON.stringify({
         title: title,
         author: author,
         content: content
      }),
      headers: {
         "Content-Type": " application/json"
      }
   });
   return res.json()
}
```

```
const deleteBlog = async (id: number) => {
   const res = await fetch(`http://localhost:8000/blogs/${id}`, {
      method: "DELETE",
      headers: {
         "Content-Type": " application/json"
      }
   });
   return res.json()
}
```

Chapter 9: Saga Flows
Flows chính là luồng chảy của dữ liệu khi chúng ta sử dụng tính năng phức tạp với Saga

#81. Blocking/Non-Blocking Effects
Tài liệu: https://redux-saga.js.org/docs/Glossary#blockingnon-blocking-call

Chúng ta đã làm quen một vài effect của Saga: takeEvery, put, call...
Về danh sách tất cả các Effect: https://redux-saga.js.org/docs/api

Effect của Saga được chia thành 2 loại: blocking và non-blocking
https://redux-saga.js.org/docs/api#blocking--non-blocking

1. Non-Blocking
Là các effect mà:
- Saga "yield' Effect mà không chờ kết quả của hành động, không quan tâm hành động đấy có thành công/thất bại hay khi nào fininsh.
- Saga 'yield' non-blocking effect đấy, sau đấy thực hiện các Effect tiếp theo.

Ví dụ:
```
yield call(ApiFn, ...args)      // Blocking: will wait for ApiFn (If ApiFn returns a Promise)
yield put(...)
yield call(otherSaga, ...args)  // Blocking: will wait for otherSaga to terminate
```

2. Blocking
Blocking là các Effect mà:
- Saga "yield" Effect này và đứng đó để chờ kết quả của hành động và không thực hiện "các yield" tiếp theo

#82. Watcher/Worker
https://github.com/redux-saga/redux-saga/issues/684

Tài liệu:
https://redux-saga.js.org/docs/Glossary#watcherworker

takeEvery = take + fork

```
function* watchFetch() {
  while (true) {
    const action = yield take(FETCH_POSTS)
    yield fork(fetchPosts, action)
  }
}
```

#83. Phân tích luồng hoạt động của login flow
//todo

#84. Tính năng Login/Logout
//todo

#85. Các kiến thức chưa đề cập
Tài liệu của Redux Saga:
https://redux-saga.js.org/

#86. Khi nào nên sử dụng Redux Saga ?

Nhược điểm của Redux Saga:
Generator tạo ra "steep learning curve" (khó học dành cho beginner)
=> nếu dùng, cần cân nhắc liệu rằng nó có đáng giá cho "team members" dành thời gian để học, vì nó ko giống như cách code javascript/nodejs thông thường, nên cần "thời gian" để thích nghi.

Redux-thunk middleware thì đơn giản hơn và đáp ứng hầu hết các trường hợp hay gặp.

Tuân theo nguyên tắc:
- redux-thunk để xử lý logic đơn giản
- redux-saga dành cho những trường hợp phức tạp

Chapter 9: Tổng kết các kiến thức đã học
Tổng kết các kiến thức về quản lý Global State của ứng dụng React

#87. Về sharing data between components
Nếu học về Redux thuần hoặc redux hook (không sử dụng Redux toolkit)
=>

Cần chia sẻ data giữa các components thì:

- Sử dụng Context API
- Sử dụng Redux

Lưu ý: không nên sử dụng Redux trên server (ví dụ dùng với Next.js)

#88. Về Fetching/Mutate Data

- Sử dụng React Query để fetching data
- Cần cache/mutate và revalidate data

Lưu ý: không nên sử dụng React Query trên server (ví dụ dùng với Next.js)

#89. Về các thư viện khác

Về sharing data, ngoài Redux, có thể dùng:

- Mobx: https://mobx.js.org/README.html
- Recoil: https://recoiljs.org/
- Zustand: https://docs.pmnd.rs/zustand/getting-started/introduction

Về fetching/mutate data, ngoài React Query, có thể dùng:

- RTK Query : https://redux-toolkit.js.org/rtk-query/overview
- SWR: https://swr.vercel.app/

Tham khảo:
https://tanstack.com/query/v5/docs/react/comparison

#90. What's next ?

Tự làm project thực hành

Nếu lựa chọn React Render Trên Server, lựa chọn Next.js:
https://hoidanit.com.vn/khoa-hoc/react-pro-max-voi-nextjs-lam-chu-toan-dien-reactjs-hien-dai-65198100e6bafa8caad417a6.html

Lời Kết

Như vậy là chúng ta đã cùng nhau trải qua hơn 90+ videos về sử dụng và quản lý Global State/Fetching/Mutate Data cho ứng dụng React.

Tất cả các kiến thức mình chia sẻ, đều được lấy từ kinh nghiệm đi làm của mình và các trang tài liệu:
https://redux-toolkit.js.org/
https://redux-saga.js.org/
https://tanstack.com/query/latest

Dĩ nhiên rằng, trong quá trình quá trình thực hiện khóa học này, mình sẽ không thể tránh khỏi những sai sót.

Vì vậy, nếu thấy sai sót, các bạn cứ thoải mái đóng góp qua Fanpage Hỏi Dân IT nhé.
https://www.facebook.com/askITwithERIC

Nếu bạn thấy khóa học này hữu ích, đừng quên Review đánh giá trên Udemy để nhận được ưu đãi (giảm giá) cho các khóa tiếp theo nhé ^^

Hẹn gặp lại các bạn ở các khóa học tiếp theo ….
Hỏi Dân IT (Eric)