

Chào! CFD TEAM

1. Ai học?
2. Học cái gì?
3. Học như thế nào?
4. Học rồi cuối cùng được cái gì?

CFD REACT JS

1. Javascript nâng cao
2. React Js
3. Redux

Đạt được sau khóa học

1. 2 dự án sau khi hoàn thành khóa học.
2. Nâng cao trình độ Javascript, ES6.
3. Nâng cao trình độ tư duy logic.
4. Có khả năng phân tích, thiết kế và phát triển sản phẩm sử dụng React JS
5. Biết cách sử dụng các thư viện đi cùng: react-router-dom, redux...
6. Biết cách sử dụng Git.
7. Tự tin trả lời các câu hỏi phỏng vấn về React JS.

Yêu cầu trước khi học

1. Có thể coding web responsive từ bản thiết kế (CFD căn bản).
2. Kiến thức cơ bản về Javascript, Javascript DOM.
3. Tinh thần ham học hỏi, chịu khó và tham gia đầy đủ các buổi học.

Giới thiệu khóa học React JS

Ngày 1 - 7 - **CHUẨN BỊ**: Làm quen với React đơn giản:

1. Cài đặt git, cấu trúc thư mục, cài đặt giao diện, làm quen với Component, Props
2. Làm quen với State, Ref, Validate form, xử lý sự kiện. Portals
3. React-router-dom: Thư viện quản lý router làm SPA
4. Sử dụng Hook, làm quen với Lifecycle.
5. Xử lý dữ liệu, làm việc với server. Promise, async/Await, JSON. API
6. Giải bài tập
7. Giải bài tập, hoàn thành trang CFD

Giới thiệu khóa học Reactjs

Ngày 8- 10 - **CHẠY**: React nâng cao - Sử dụng thư viện, quản lý dữ liệu nâng cao:

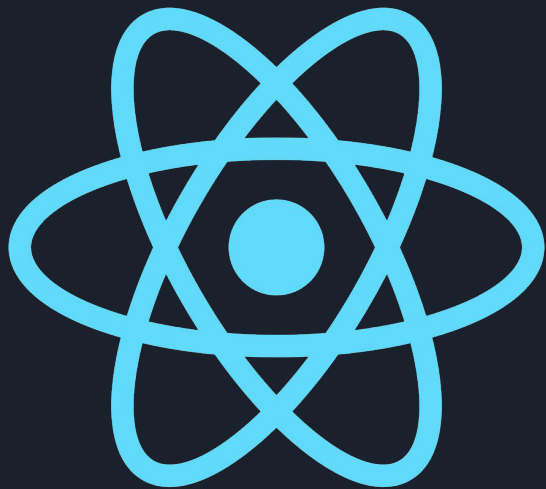
8. Cài đặt Redux, giới thiệu và thực hành (Giới thiệu Context, Reducer).
9. Redux thunk, redux saga, thực hành redux
10. Thực hành redux saga và thunk

Giới thiệu khóa học React JS

Ngày 11 - 16 - **VỀ ĐÍCH**: Thực hành dự án cuối khóa:

11. Giới thiệu, yêu cầu và hướng dẫn dự án cuối khóa, setup dự án: react-router-dom, redux, redux-saga
12. Làm việc về dữ liệu với backend
13. Sử dụng redux, redux-saga quản lý state
14. Hoàn thành đăng nhập, đăng ký/ Quản lý cart, thông tin cá nhân
15. Tối ưu dự án
16. Build dự án thành file và hướng dẫn bàn giao

Ngày 17 - 18 - **ANIMATION**: Hướng dẫn animation (Nghĩa):

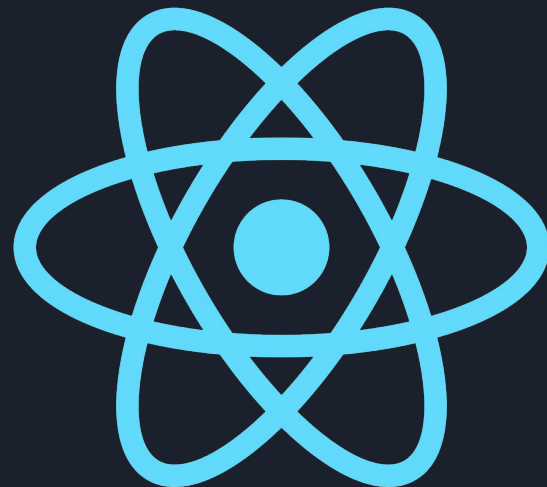


CHUẨN BỊ

CHUẨN BỊ

Ngày 1

1. Cài đặt git, hướng dẫn sử dụng git cơ bản
2. Cấu trúc thư mục dự án react
3. Cài đặt React
4. Cài đặt giao diện
5. Làm quen với Component, Props



Git

1. Git là gì?
2. Cài đặt Git
3. Các câu lệnh thường gặp trong git

Git là gì?

- ❖ Git là một "hệ thống điều khiển phiên bản phân tán" (Distributed version control system) miễn phí và mã nguồn mở, nó được thiết kế để một nhóm làm việc cùng nhau có thể chia sẻ và quản lý phiên bản các tập tin trong quá trình làm việc.
- ❖ Tiết kiệm thời gian
- ❖ Làm việc offline
- ❖ Khôi phục khi gặp lỗi, yên tâm
- ❖ Không trộn lẫn các công việc
- ❖ Cộng đồng người dùng lớn

Cài đặt Git

Tham khảo:

<https://kipalog.com/posts/Huong-dan-chi-tiet-tung-buoc-c-cai-dat-Git-cho-Windows>

Các câu lệnh thường gặp trong git

Tham khảo:

<https://viblo.asia/p/tap-hop-nhung-cau-lenh-git-huu-dung-dWrwwWr2vw38>

Câu lệnh thường gặp trong git

❖ **Git config:**

```
git config --global user.name "John Doe"
```

```
git config --global user.email "john@example.com"
```

❖ **Giúp Git bỏ qua folder node_modules:**

Tạo file `.gitignore` rồi thêm vào nội dung:

```
# dependencies
```

```
/node_modules
```

❖ **Khởi tạo Git repo cho code có sẵn:**

```
cd existing-project/
```

```
git init
```

❖ **Clone một remote repo:** `git clone https://github.com/user/repository.git`

Câu lệnh thường gặp trong git

- ❖ **Cập nhật thay đổi:**
git add .
git commit -m "Message"
- ❖ **Cập nhật lên server:**
git push origin <name_branch>
- ❖ **Xem lại lịch sử commit:** git log
- ❖ **Xem thay đổi trước khi push:** git diff
- ❖ **Pull từ remote repository:** git pull origin master

Các câu hỏi thường gặp khi phỏng vấn

1. Git fork là gì? Sự khác nhau giữa git fork, branch và clone?
2. Sự khác nhau giữa pull request và branch?
3. Sự khác nhau giữa git pull và git fetch?
4. Làm thế nào để revert previous commit trong git?
5. git cherry-pick là gì?
6. Giải thích những ưu điểm of Forking Workflow?
7. Sự khác nhau giữa HEAD, working tree và index?
8. Trình bày quy trình làm việc của Gitflow Workflow?
9. Khi nào nên sử dụng git stash?
10. Làm thế nào để loại bỏ một tập tin từ git mà không cần loại bỏ nó khỏi file system của bạn?
11. Khi nào nên sử dụng git rebase thay vì git merge?



Cài đặt React

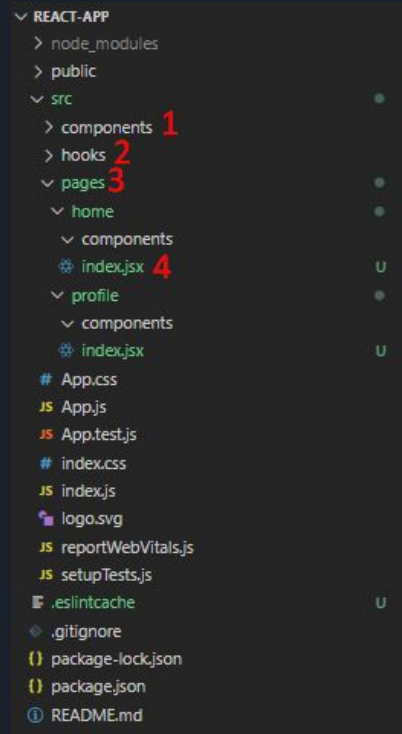
1. **Cài đặt Nodejs:** <https://nodejs.org/en/>
2. **Cài đặt react tool:** `npm install -g create-react-app`
3. **Tạo mới project react bằng react tool:** `create-react-app my-app`

Câu lệnh thường gặp khi làm React

1. **Run 1 project react:** `npm start`
2. **Build dự án thành sản phẩm production hoàn chỉnh:** `npm build`
3. **Cài thêm mới thư viện:** `npm install <tên thư viện> [--save-dev]`

Cấu trúc thư mục

1. components: Chứa các Components dùng chung
2. hooks: Chứa custom hook dùng chung
3. pages: Chứa các page, trong đó có index.jsx là file giao diện chính, components,... dùng riêng cho page
4. File giao diện chính cho từng page



Component, props

Component React là gì?

- Trong React, các Component hoạt động giống như các hàm trả về các thành phần HTML
- Các Component là các thành phần độc lập và có thể sử dụng lại.
- Các Component thực hiện công việc giống như các functions trong JavaScript nhưng chúng độc lập và nhiệm vụ chính là trả về HTML thông qua hàm render
- Có 2 loại Component là Function Component và Class Component.

```
import React, { useState } from 'react';

function Example() {
  // Declare a new state variable, which we'll call "count"
  const [count, setCount] = useState(0);

  return (
    <div>
      <p>You clicked {count} times</p>
      <button onClick={() => setCount(count + 1)}>
        Click me
      </button>
    </div>
  );
}
```

Component, props

Props là gì?

- Props thực chất là viết tắt của Properties, giữ thông tin về Component
- Props là những tham số bên ngoài truyền vào Component, bên trong sử dụng tham số đó để thực hiện logic
- Không nên thay đổi props trong quá trình thực hiện

* Khi khai báo:

```
function Welcome(props) {  
  return <h1>Hello, {props.name}</h1>;  
}
```

* Khi sử dụng:

```
const element = <Welcome name="Sara" />;
```

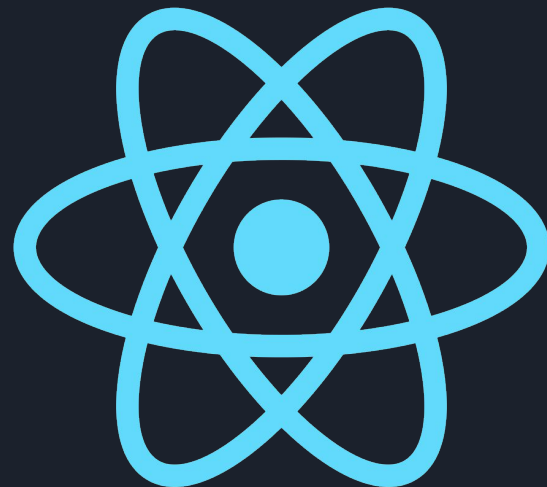
Thực hành

1. Cài đặt giao diện website CFD
2. Đưa lên Github

CHUẨN BỊ

Ngày 2

1. Làm quen với State
2. Ref
3. Portal
4. Xử lý sự kiện.
5. Validate form



Xử lý sự kiện

Sự kiện (Event) là gì?

- Là 1 hành động nào đó của người dùng trên website: onclick, dblclick, mousemove, mouseenter,.....
- React event sử dụng camelCase cho tên event

* Cách sử dụng:

```
<button onClick={activateLasers}>  
  Activate Lasers  
</button>
```


Component, props, state, Ref

State là gì?

- State cũng giống như props, cũng giữ thông tin về Component. Tuy nhiên, loại thông tin và cách xử lý khác nhau. State là 1 thành phần của Component, trong khi props được truyền từ ngoài vào.
- Cập nhật state bằng setState

```
import React, { useState } from 'react';

function Example() {
  // Declare a new state variable, which we'll call "count"
  const [count, setCount] = useState(0);

  return (
    <div>
      <p>You clicked {count} times</p>
      <button onClick={() => setCount(count + 1)}>
        Click me
      </button>
    </div>
  );
}
```

Ref (level 1)

Ref là gì?

- React refs là một tính năng hữu ích để chúng ta có thể để tham chiếu một element trong DOM hoặc từ một Component cha đến Component con. Điều này cho phép chúng ta đọc và chỉnh sửa các element đó. Hoặc lưu 1 biến giá trị.

```
function TextInputWithFocusButton() {  
  const inputEl = useRef(null);  
  const onButtonClick = () => {  
    // `current` points to the mounted text input element  
    inputEl.current.focus();  
  };  
  return (  
    <>  
      <input ref={inputEl} type="text" />  
      <button onClick={onButtonClick}>Focus the input</button>  
    </>  
  );  
}
```

Ref (level 2)

forwardRef:

- forwardRef là cách Component cha dùng ref lấy ra element của Component con

```
const FancyButton = React.forwardRef((props, ref) => (  
  <button ref={ref} className="FancyButton">  
    {props.children}  
  </button>  
));  
  
// You can now get a ref directly to the DOM button:  
const ref = React.createRef();  
<FancyButton ref={ref}>Click me!</FancyButton>;
```

Ref (level 3)

useImperativeHandle:

- useImperativeHandle là cách Component cha dùng ref lấy ra “1 thể hiện khác” của Component con
- Thường được sử dụng chung với forwardRef

Portals

Portals là gì?

- Cách Element được render
- Thường được sử dụng cho các element bắt buộc phải render ngoài root Element
- vd: Popup, Notification, tooltip,...

```
import React from 'react'
import ReactDOM from 'react-dom'

function Modal({ children }) {
  return ReactDOM.createPortal(
    <div id="modal-wrapper">
      {children}
    </div>,
    document.querySelector('body'),
  )
}

export default Modal
```

Validate Form

Validate Form là gì?

- Validate form là kiểm tra dữ liệu trước khi gửi lên server để xử lý xem có đúng chuẩn hay không rồi mới gửi.
- Validate form chỉ là bước nên có, giúp cải thiện UX ở giao diện người dùng.

* Các quy tắc validate form thông dụng:

- Email: phải đúng định dạng: example@gmail.com
- Số điện thoại: Tùy quốc gia mà có quy tắc khác nhau: Vd SDT Việt Nam là: 9 - 11 số, Có thể có hoặc không cần +84, số điện thoại phải nằm trong các đầu số quy định của VN.
- Username: không được để trống, loại bỏ khoảng trắng đầu và cuối
- Tuổi: không được dưới 18 tuổi
- Số thẻ visa, Mã Zipcode

** Trim, Không có 2 ký tự <space> cùng lúc.



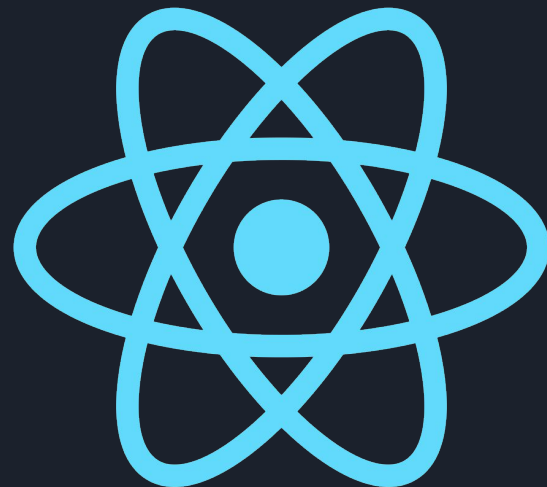
Thực hành

1. Validate form cho trang đăng ký.
2. Validate form cho trang hợp tác.
3. Validate form cho trang thông tin cá nhân.

CHUẨN BỊ

Ngày 3

1. React-router-dom: Thư viện quản lý router làm SPA
2. Chuyển trang không reload
3. Làm quen các khái niệm liên quan router






React-router-dom

React-router-dom là gì?

React-router-dom là thư viện quản lý route (các link url) của website. Dùng để xây dựng những trang Single Page Application (SPA) đơn giản hơn.



Các trường hợp thường gặp trong react-router-dom

1. Router đơn giản
2. Router lồng
3. Dynamic router (URL param)
4. Query Parameters
5. Redirect
6. No Match (404)
7. Animated Transitions

Các hook thường sử dụng

1. useHistory: Cho phép truy cập tới history của navigate
2. useLocation: Trả về location hiện tại khi url thay đổi, giống như useState thay đổi, hữu ích khi muốn bắt sự thay đổi của url
3. useParams: Trả về 1 object chứa các params
4. useRouteMatch:

Các Component thường sử dụng

1. `<BrowserRouter>`: Context API - bọc ngoài cùng toàn bộ dự án
2. `<HashRouter>`: Context API - bọc ngoài cùng toàn bộ dự án (url dạng hash)
3. `<Route>`: Render Component hoặc children khi url được match
4. `<Link>`: Dẫn link, thay cho thẻ a
5. `<NavLink>`: Dẫn link, thay cho thẻ a nhưng có thêm trạng thái active
6. `<Prompt>`: Dùng xuất hiện câu thông báo khi user rời khỏi trang hiện tại
7. `<Redirect>`: Redirect tới 1 trang nào đó.
8. `<Switch>`: Render element đầu tiên mà nó match

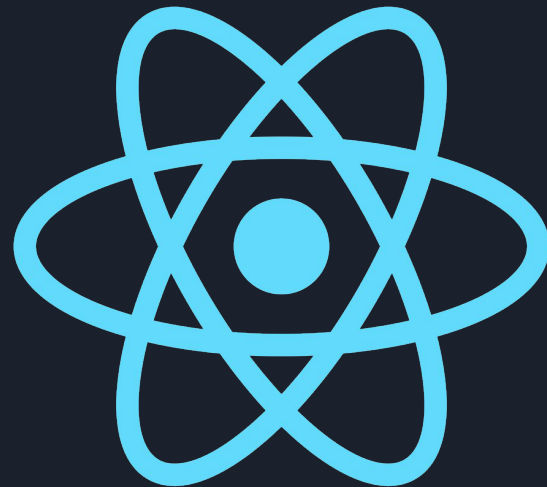
Thực hành

1. Áp dụng react-router-dom vào website:
 - Trang chủ, Khóa học, CFD Team, FAQ, Hợp tác
 - Thông tin tài khoản, Khóa học của tôi, Dự án của tôi, Lịch sử thanh toán, quản lý COIN
2. Thêm trang 404
3. Sử dụng URL Parameters cho Trang đăng ký khóa học, chi tiết khoá học

CHUẨN BỊ

Ngày 4

1. Sử dụng Hook
2. Làm quen với LifeCycle của hook.
3. Authentication
4. validate



Hook

Hook là gì?

- Hooks là một bổ sung mới trong React 16.8.
- Hooks là những hàm cho phép bạn “kết nối” React state và lifecycle vào các Components sử dụng hàm.
- Với Hooks bạn có thể sử dụng state và lifecycles mà không cần dùng ES6 Class.

Tại sao chúng ta cần React Hooks?

- Sự rắc rối của Lifecycles trong class
- Wrapper hell: các component được lồng (nested) vào nhau nhiều tạo ra một DOM tree phức tạp.
- Viết ngắn, giảm lượng code và nhanh hơn Class thông thường

Hook

Sử dụng Hook (Tên hook có tiền tố “use”)

```
import React, { useState } from 'react';

function Example() {
  // Declare a new state variable, which we'll call "count"
  const [count, setCount] = useState(0);

  return (
    <div>
      <p>You clicked {count} times</p>
      <button onClick={() => setCount(count + 1)}>
        Click me
      </button>
    </div>
  );
}
```


Hook

Cơ bản

1. `useState`: Tạo ra 1 state (biến giữ giá trị) đơn lẻ giữ 1 giá trị nào đó.
2. `useEffect`: Giúp tạo ra các “hiệu ứng, slide effect” trong các chức năng.
3. `useRef`: Thao tác với các phần tử DOM

Quản lý state

4. `useReducer`: Phiên bản nâng cao hơn state, dùng để quản lý tất cả các state trong một object
5. `useContext`: Quản lý state (Giống redux)

Hook

Optimization hook

7. `useMemo`: giúp ta kiểm soát việc được render dư thừa của các Component con
8. `useCallback`: có nhiệm vụ tương tự như `useMemo` nhưng khác ở chỗ function truyền vào `useCallback` bắt buộc phải ở trong quá trình render
9. `useLayoutEffect`: Cải thiện hiệu năng render

Tìm hiểu thêm

10. `useImperativeHandle`: Customizes lại thể hiện nhận được của parent khi sử dụng ref
11. `useDebugValue`: Kiểm tra và debug. Chỉ có giá trị ở môi trường develop

Custom Hook

Custom hook là gì?

- 1 hook do mình tự định nghĩa, 1 function đặc biệt.
- Custom Hook là một hàm chứa logic được sử dụng lại. Có thể sử dụng các hook có sẵn bên trong custom Hook như useState, useEffect,... (Function bình thường không thể).
- Lập trình viên có thể tự tạo hook cho riêng mình, dùng để chia sẻ các logic sử dụng chung, tách biệt với UI. Đặt tên bắt đầu bằng từ khóa "use", vd: useClock, useParam, useLocalSave,....

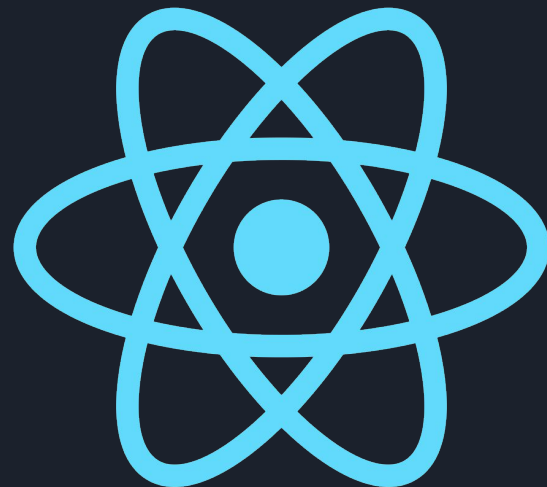
Thực hành

1. Sử dụng context API viết chức năng loading khi vào trang
2. viết 1 hook dùng để validate form
3. Authentication với Context, sử dụng custom hook
4. Viết 1 hook khi refresh lại trang thì không mất giá trị input đó
5. Viết Component PrivateRouter cho những trang profile, đăng ký
6. Đăng nhập

CHUẨN BỊ

Ngày 5

1. Bất đồng bộ, async/await, Promise, Try/Cache
2. Fetch, Axios, ajax, API
3. JSON
4. GET, POST, PUT, DELETE
5. Authentication, Authorization, Jsonwebtoken
6. Xử lý dữ liệu từ backend



Bất đồng bộ, async/await, Promise, Try/Cache

1. Bất đồng bộ là gì?
2. Promise là gì?
3. Async/Await là gì?
4. Khi nào cần sử dụng?
5. Try/Catch là gì?

JSON, Ajax, Fetch, API, Axios

1. JSON là gì?

JSON là một kiểu định dạng dữ liệu trong đó sử dụng văn bản thuần túy, định dạng JSON sử dụng các cặp key - value để lưu dữ liệu.

2. Ajax là gì?

AJAX là chữ viết tắt của Asynchronous JavaScript and XML. Nó là một bộ các kỹ thuật (GET, POST, PUT, DELETE,...) thiết kế web giúp cho các ứng dụng web hoạt động bất đồng bộ – xử lý mọi yêu cầu tới server.

3. HTTP Request là gì?

HTTP là giao thức được thiết kế theo kiểu client - server, giao tiếp giữa client và server dựa vào một cặp request - response, client đưa ra các request và server sẽ response trả lời các request này.

JSON, Ajax, Fetch, API, Axios

4. Fetch là gì?

Fetch API là một tính năng đơn giản cho việc gửi và nhận request bằng js. Với fetch thì việc thực hiện các yêu cầu web và xử lý phản hồi dễ dàng hơn so với XMLHttpRequest cũ.

5. Axios?

Axios là thư viện chuyên dùng để xử lý ajax, api



GET, POST, PUT, DELETE

1. GET: Lấy dữ liệu
2. POST: Tạo mới dữ liệu
3. PUT: Cập nhật dữ liệu
4. DELETE: Xóa dữ liệu

Authentication, Authorization, Jsonwebtoken

1. Authentication: Là quá trình xác thực người dùng đăng nhập như username/password. Có các loại xác thực phổ biến sau:
 - a. Single-Factor Authentication: Xác thực 1 lớp, dùng username và password
 - b. Two-Factor Authentication: Xác thực 2 lớp, dùng username, password và kèm với 1 mã số bí mật như mã PIN của ATM
 - c. Multi-Factor Authentication: Xác thực với nhiều lớp bảo mật. VD như ATM đăng nhập hoặc chuyển tiền phải có OTP. Đặc điểm của các lớp bảo mật phải độc lập với nhau.
2. Authorization: Là quá trình xác thực bạn có các quyền nào dựa trên thông tin đã đăng nhập ở trên.



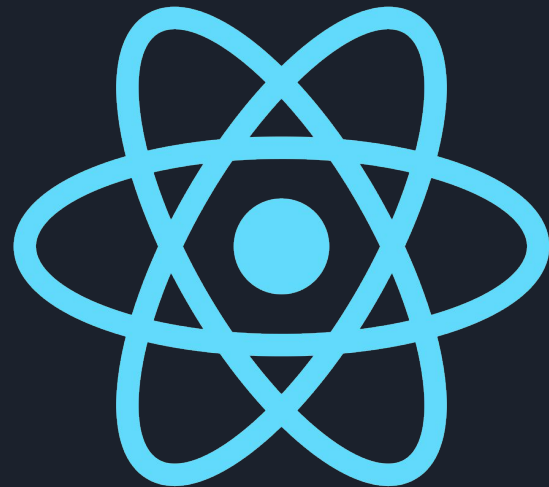
Thực hành

1. Gửi request lấy danh sách khóa học, render ra giao diện
2. Gửi contact lên server thông qua HTTP POST
3. Cập nhật thông tin cá nhân (POST)
4. Đăng ký khóa học (POST)
5. Lấy khóa học của học viên đó

CHUẨN BỊ

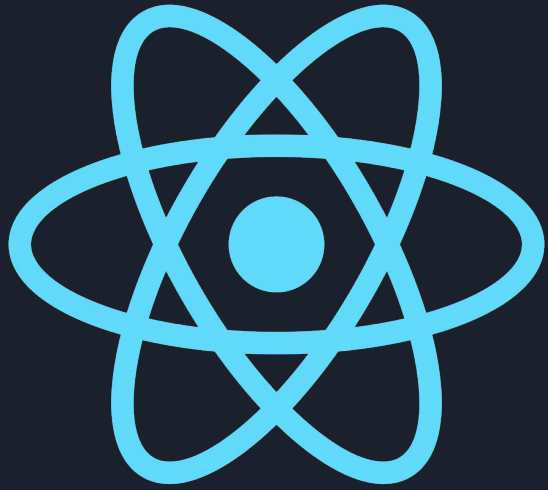
Ngày 7

1. Thực hành



Thực hành

1. Viết một hook dùng để validate form
2. Authentication với Context, sử dụng custom hook
3. Viết Component PrivateRouter cho những trang profile, đăng ký
4. Đăng nhập
5. Gửi request lấy danh sách khóa học, render ra giao diện
6. Gửi contact lên server thông qua HTTP POST
7. Cập nhật thông tin cá nhân (POST)
8. Đăng ký khóa học (POST)
9. Lấy khóa học của học viên đó

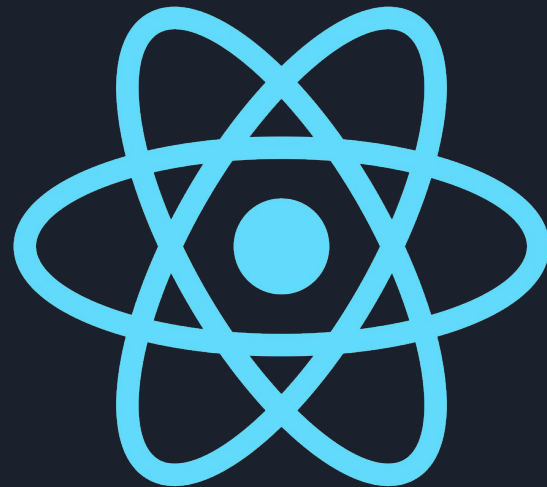


CHẠY

CHẠY

Ngày 8

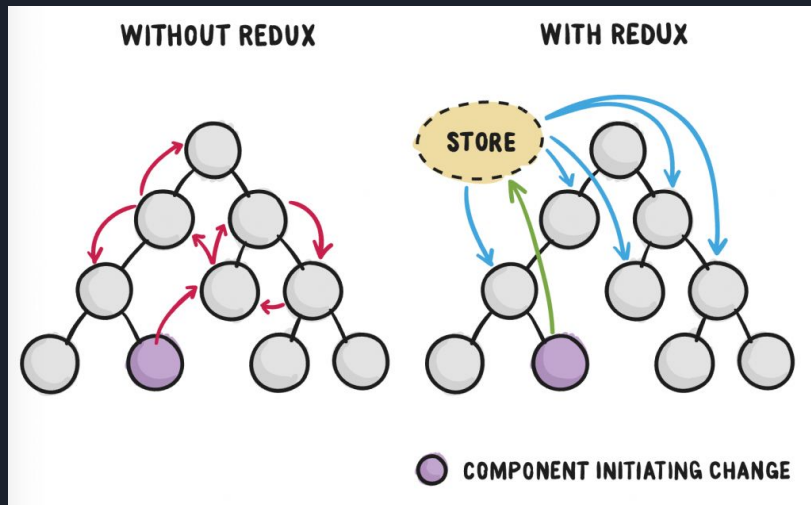
1. Concept về Context API và Redux
2. Giới thiệu Context, Reducer
3. Giới thiệu và cài đặt Redux
4. Cách sử dụng trong dự án (Login,...)




Concept (Ý tưởng)

Vấn đề:

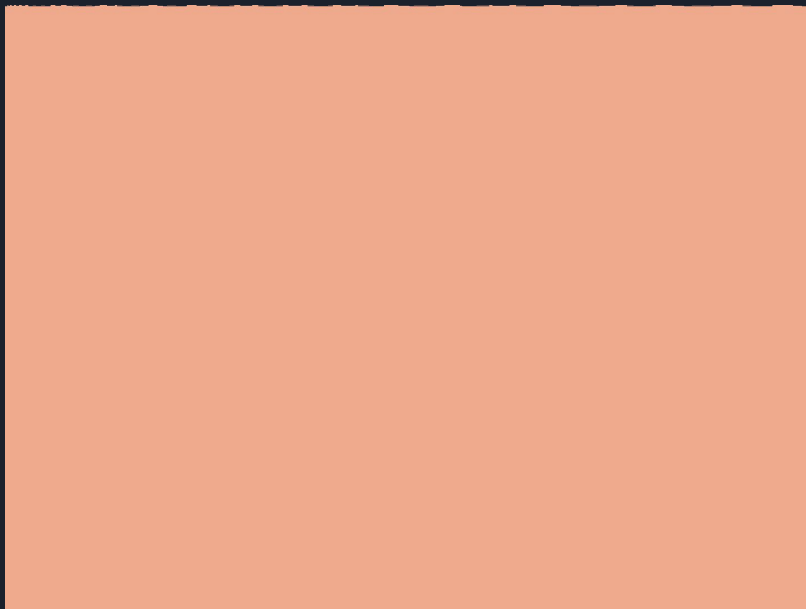
Khi ứng dụng càng lớn, sẽ có nhiều Component sử dụng chung state, vd: login chứa thông tin login. Sẽ có nhiều Component cần thông tin login để render view. Hoặc những Component không có chung cha cần chia sẻ thông tin với nhau. Trường hợp này state sẽ đi lòng vòng trên “tree” rất khó quản lý





Concept (Ý tưởng)

Cách thức hoạt động của Redux:



Giải thích các khái niệm:

Store: Nơi lưu trữ dữ liệu

Reducer: Nơi update state

State: Dữ liệu truyền qua view

View: Component React

Action: 1 hàm action thực hiện 1 hành động, return 1 object theo quy định

Event: 1 sự kiện do người dùng tác động hoặc được gọi tự động từ react

Cài đặt Redux

`npm install redux react-redux`

Tham khảo: <https://redux.js.org/introduction/getting-started>

Cài đặt Redux dev tool

Redux devtool là extension dùng để view log của redux dùng trong quá trình dev

Cài đặt: <https://github.com/zalmoxisus/redux-devtools-extension>

```
let store = createStore(counter, window.__REDUX_DEVTOOLS_EXTENSION__ && window.__REDUX_DEVTOOLS_EXTENSION__())
```

Các function thường sử dụng

createStore: Tạo store cho ứng dụng, 1 hệ thống nên chỉ có 1 store

combineReducers: Gộp nhiều reducer lại với nhau

Provider: React Context chứa value để truyền dữ liệu store đi khắp Component

useSelector: Lấy ra giá trị cần lấy

applyMiddleware: Gắn 1 middleware function vào giữa action và dispatcher

useDispatch: Lấy ra dispatch

useStore: Hook lấy ra toàn bộ store của redux



Thực hành

1. Sử dụng redux cho trang chủ, cache dữ liệu
2. Sử dụng redux cho trang chi tiết, cache dữ liệu
3. Sử dụng redux cho trang đăng ký
4. Sử dụng redux làm chức năng đăng nhập, authen

CHẠY

Ngày 9

1. Redux thunk
2. Redux saga
3. Thực hành redux

