ReactJS :

<https://reactjs.org/docs/hello-world.html>

chương 01 : Tìm hiểu RJ

1. Extention : react developer tool
2. Tích hợp bootstrap vào reactJs
3. Tìm hiểu về JSX : js + xml
4. Props : thuộc tính của 1 component
   1. Muốn lấy thuộc tính từ component cha truyền vào thì dung this.props.name
   2. Thuộc tính đặc biệt : this.props.children
   3. Khi truyền cẩn thận k nó hiểu nhầm là string mà lên dung {}
5. Event:
   1. Sự kiện click và khi click vào thì nó chạy 1 hàm và nếu truyền vào hàm đó thì dùng arrow function
6. Refs : chú ý khi sử dụng 1 input và lấy thuộc tính của nó thì đặt cho nó 1 cái ref và nhớ phải bind cho nó
7. State : trạng thái

Chương 02: ứng dụng Todolist

Chia component

--App

--Title

--Controll

--Search

--Sort

--Form

--List

--Item

a. sử dụng mockdata : để đưa dữ liệu ảo

b. npm uuid giúp sinh ra 1 chuỗi ramdom

c. chức năng ẩn hiện form

luồng dữ liệu chạy

app(state)=>props là 1 hàm => onclick

d. chức năng ấn vào nút cancel thì nó đóng lại form

e. chức năng tìm kiếm

: bản chất là phải thay đổi giá trị state gì đó ở app.js ấy

Khi click vào button go thì lấy ra được giá trị từ ô input đã

Giờ làm sao đẩy giá trị state ở ô search về cho tk app là ok

Lại phải từ tk app đẩy 1 cái props sang => dựa vào cái prop đó mà thay đổi lại cái state từ app

Thư viện mới npm lodash

js check contain string

f. chức năng sort ( sap xep)

cần biết sắp xếp theo loại nào ( name hay lever) : state : orderBy : ‘name’

cần biết sắp xếp theo tang dần asc hay desc : state orderDir:’asc’

* Công việc là chọn vào đâu thì in ra nội dung ở đó ví dụ name-asc
* Lever asc …

Thực hiện click nó lấy ra giá trị theo từng ô

* Sử dụng lodash để sắp xếp

g. chức năng xóa

click vào nút đó thì phải lấy ra được cái id từ item gửi về cho tk app và tại đây tk app xóa nó đi

h. chức năng them mới

để lấy dữ liệu từ form ra thì coi lại phần form trong reactjs

* Lấy được dữ liệu ra thì chúng ra sẽ đẩy nó về cho tk app rồi add nó vào item là ok

k. chức năng edit

trong app cần có 1 state để cho biết phần tử được chọn là phần tử nào : itemSelected => đẩy nó vào cho tk form,

* Khi click vào thì lấy phần tử đó (item.js) đẩy về cho tk app cập nhập lại giá trị selected là xong

Lodash update obj in array

Khi click qua giá trị thì phải nhận được 1 cái lycyle gì đó

1. Sử dụng local store để lưu lại phía trình duyệt
   1. Local storega dữ liệu bị mất khi xóa cache của trình duyệt
   2. Session storage dữ liệu bị mất khi đóng trang web lại
   3. Local storage thì dữ liệu lưu trữ khoảng 5MB > cookie
   4. Nó lưu theo kiểu key : value
      1. Muốn thiết lập : localStorage.setItem(name, data)
      2. localStorage.getItem(name)
      3. localStorege.removeItem(name)
      4. trước khi render ra thi cập nhập dữ liệu cho tk item
      5. viết trong componentWillMount

public ứng dụng

reactjs buid

ch03. Redux

Note :

* store : chứa các state mặc định
* muốn lấy 1 giá trị gì đó dùng getState();
* muốn thay đổi gì đó trên cái state -> gọi những cái hành động truyền action vào dẫn đến action được gọi và truyền vào trong store -> thông qua appReducer để làm công việc action và x thì công việc x action là y thì công việc y -> rồi trả về state mới
* muốn thực hiện 1 hành động gọi đến cái store cần có phương thức dispatch()

1. . Sử dụng redux
   1. Gg redux ra
      1. Tạo store
      2. Muốn thực hiện hành động đến 1 cái store thì có method dispath
      3. Để nhập nhiều reducers thì chúng ta sử dụng combineReducers
      4. Tìm hiểu về phương thức subcribe
         1. Khi thay đổi gì đó ở tk store thì nó sẽ thực hiện thay đổi nó : nó bắt hành động
   2. Sử dụng redux cần import provider vào tập tin index.js
   3. Đổ danh sách dữ liệu ra item
      1. List kết nối tới kho chứa (store) rồi lấy dữ liệu đổ ra
         1. Import vào cái connect đã
         2. Export default connect(null)
   4. Thực hiện chức năng toggleForm
      1. Control lấy cái isshowForm
         1. Từ control kết nối tới soter
      2. Form Cũng lấy cái ishowForm
      3. Click vao thi mở và đóng form : thì chỉ cần khi clich vào thì nó thay đổi giá trị isShowForm => khi muốn thay đổi state trong store thì cần dùng dispatch
   5. Thực hiện lick vào edit thì hiển thị cái form trước đã
   6. Chức năng search thì trong redux tạo search rồi
      1. Tạo 1 actSearch truyền vào giá trị search
      2. Vậy khi search thì component phải lấy cái state sear
      3. Lấy giá trị ở ô search cập nhập lại nó trong cái reducers
      4. Khi click vào clear thì cho nó lại = rỗng
   7. Chu nang sort
      1. Hiển thị nó ở trong list component
      2. Chức năng hiển thị kiểu sort nằm trong Sort component
         1. Muốn lấy thông tin đó ra thì cần connect tới đó đã rồi lấy
         2. Ckick vào thì gọi đến store cập nhập lại giá trị
   8. Chức năng add
      1. Thì thực hiện add như bình thường
   9. Chức năng edit
      1. Cần biết giá trị lựa chọn là item nào => tạo 1 cái itemSelected trong cái store
      2. Từ form đến store lấy cái thành phần đang dc chọn ra

Ch04 – shopping cart

* Chia component
* Cài đặt redux npm install –save redux
* Cài đặt react redux : npm install --save react-redux
* Tạo các foder
  + Constanis : định nghĩa các hằng số
  + Action : các hành động
  + Reducers: chứa các state
* Phân tích ứng dụng có:
  + Danh sách các sản phẩm
  + Các sản phẩm trong giỏ hang
  + Câu thông báo
* ứng với nói là những reducers tương ứng
* Tạo 1 store
* Trong index.js nhớ import provider và store
  + - 1. Chức năng hiển thị danh sách item
         1. Connect ProductList với store để lấy ra state

Thực hiện đổ dữ liệu ra và viết thành 1 hàm

* + - * 1. Xử ký vs productItem

ProductList truyền giá trị cho ProductITem

Lấy nó ra rồi in thử ra xem rồi thực hiện đổ dữ kiệu

Có 1 vấn đề như này : sản phẩm giống hệt nhau nhưng có 1 cái cấu trúc nhỏ khác nhau

Là sản phẩm có thể mua và sản phẩm ko thể mua => tốt nhất là viết 1 hàm tạo ra phần đó

Hiển thị giá tiền thì xây dựng 1 cái helper để hiển thị 1 cách linh hoạt hơn

* + - 1. Chức năng hiển thị sản phẩm trong giỏ hang
         1. Cart.js kết nối tới store : lấy ra state cart chứa các sản phẩm trong giỏ hang
         2. Chức năng mua sản phẩm :

Khi click vào productItem thì mua nghĩa là tại thẻ a khi được click vào thì truyền vào ngay cái sản phẩm đó => lấy ra cái số lượng thông qua form : chú ý cần đặt name trùng tên vs state trong input

Xây dựng 1 lớp validate để kiểm tra các giá trị nhập vào có hợp lệ hay ko

Khi nhận thấy dữ liệu k hợp lệ thì gửi 1 cái dishpath (hành động) về store

ở notify component lấy cái state nó ra rồi gắn vâu thông báo vào

* + - * 1. Thêm sản phẩm vào giỏ hang

Khi mua thì dữ liệu cần gửi sang cart reducer => thực hiện dispath

Cách debug : log cái action đó ở reduce ra xem có thông tin gì ko

Lấy được thông tin đó ra thì lấy nó nhét vào state của nó là xong

Thực hiện lưu thông tin của sản phẩm vào localStorage rồi lấy thông tin của nó đổ ra phần hiển thị là ok

* + - * 1. Thực hiện thống kê

Lodash sum value of key => tìm hàm tính tổng

Thực hiện thống kê thông qua các hàm trong lodash

* + - * 1. Chức năng xóa sản phẩm

Lick vào button delete -> in ra sản phẩm đó

Dishpath nó truyền vào sản phẩm này để reducer xóa nó

* + - * 1. Chức năng update số lượng sản phẩm

Cần biết sản phẩm nào và số lượng là bao nhiêu

In nó ra xem nhận dc chưa trong component của nó ấy

Trong reducer thì in tk action đó ra xem có nhận được những thông tin cần thiết chưa

* + - * 1. Redux và các vấn đề nâng cao

Vào trang chủ của redux và xem phần example

Folder container :mang ý nghĩa là gì

Nó sẽ kết nối tới redux và truyền dữ liệu cho component

Component <- Container -> redux

React Proptype : xem nó là gì : kiểm tra xem các propertive nhận vào đứng theo kiểu dữ liệu đó ko : thiết lập kiểu dữ liệu của các thuộc tính truyền vào

Truyền theo dạng children : {this.props.children}

Chỉ thực hiện connect 1 lần duy nhất dựa vào product container thôi khi click vào thì vẫn truyền thông qua 1 cái props thôi

* + - * 1. Áp dụng tương tự để áp dụng cho phần cart :D
        2. CartContainer
        3. Notifile container
        4. => dự án lớn mới sử dụng 😊)

Chuong 05

Gg : react router

* Phai import import { BrowserRouter as Router, Route, Link } from "react-router-dom"; vas u dung react router
* Exatc : giup mo dung component do
* Ko dùng thẻ a mà dùng navlink để tạo ra => cấu trúc k đúng => gg react router add class => khi dùng tk link nó tạo giúp ta cái thẻ a và muốn dùng thuộc tính gì thì bổ xung them vào
* Khi vào menu nào thì phải active cho nó (activeClassName)
* Khi người dùng gõ vào đường dẫn ko tồn tại thì lấy 1 nội dung của component notfound => để hiển thị ra => dùng tk swith ý nghĩa đường dẫn đã trùng với tk này rồi thì lấy chính nó : nghĩa là chỉ lấy 1 trong những trường hợp đó chứ k phải lấy tất cả
* Làm menu dạng abc/123 : thì lấy giá trị từ trên url xuống this.props thì xem in ra dc cái gì :D
* Khi chọn 1 khóa học thì nó lấy cái param ở trên url hiển thị ra
* Xấy dựng 1 component để thiển thị thôn tin chi tiết của 1 khóa học courseITem
* Chú ý tk exac : nếu /muốn nó vẫn đúng khi /avfc/ad thì exac cho no = false
* Muốn lấy đường dẫn trên url thì lấy thông qua cái params
* Túm lại là phần router-config : nó bảo là truy cập vào đường dẫn / thì hiển thị home component : đọc phần **https://reacttraining.com/react-router/web/example/sidebar**
* Còn phần menu thì phải hiển thị theo định dạng mình quy định sẵn
* Tìm hiểu về Propmpt : [**https://reacttraining.com/react-router/web/example/preventing-transitions**](https://reacttraining.com/react-router/web/example/preventing-transitions) **: bạn có chắc chắn chuyển trang ko => s**ử dụng tk promt và dựa vào cái location thì cũng lấy dc giá trị mà người dụng muốn đi đến
* Chức năng Redirect (auth) : khi truy cập vào admin1 thì k được mà nó sẽ đẩy về trang redirect
  + Chú ý tại component sử dụng match thì router config cũng phải truyền cho nó đối tượng match

Muốn chuyển nó về trang 1 thì dùng redirect là xong và đẩy kèm theo 1 cái props location và chú ý cần truyền them location cho nó ở file config => tại file home có thê in ra được ai là tk redirect về nó

Chương 05 : phần 2 áp dụng

1. Chia component các kiểu đã xong
2. Khai báo router
   1. Cài đặt react router
   2. Tạo tập tin router-config để thực hiện cấu hình
   3. Vào lại app khai báo router dom và im port tập tin cấu hình roter config vào
   4. Trong component có chứa menu thì sẽ đưa vào 1 mảng cấu hình để đưa ra cấu trúc của menu phía dưới
   5. Sử dụng navlink để sinh ra các menu thích hợp
   6. Custom lại link thì trong doc của react router có phần custom link
   7. Chú ý : giữa **link** và **navlink** => tk link thì k tự động gắn class active vào còn tk navlink thì tự động gắn active vào
3. Xây dựng page Login và giới hạn vùng truy cập
   1. Cài đặt redux đã vs
   2. Chức năng khi người dùng truy cập đến phần blog thì kiểm tra xem người dùng đã đăng nhập thành công hay chưa nếu rồi thì ok còn ko thì redirect tới trang login.
   3. Tại blogPage thì cần kết nối tới redux
   4. Kiểm tra nếu user === login thi redirect về trang login
   5. Nhớ import redirect vào nha
   6. Chức năng login :
      1. Kiểm tra người dùng đăng nhập hay chưa nếu rồi thì hiển thị thông tin ra
      2. Nếu chưa thì hiển thị formlolgin
      3. => tại loginPage cũng thực hiện method connect
   7. Chức năng logout : tai component UserControl
      1. Khi click vào loglout chỉ cần kết nối tới store và thay đổi isLogin === false và cho username = rỗng là ok
      2. Tại formlogin lấy dc các thông tin người dùng nhập vào đã

Chương 06. Tìm hiểu về API

REST API

Get /tasks -> thưởng thì để hiển thị dữ liệu

Post /tasks -> thường để tạo mới dữ liệu

Pust /tasks/{id} -> update dữ liệu

Delete /tasks/{id}-> delete dữ liệu

* Dữ liệu trả về HTTP Status Codes
* Giả lập API thông qua mockapi
* Sau đó dùng postman để test các dữ liệu đã gửi được hay chưa
  + Chú ý với tk post để gửi dữ liệu thì dùng cái tk body và form-urlencoded

02. Tìm hiểu về json server

Gg json server :

* Cài đặt trong cmd luôn
* Cài đặt gói tổng hợp : npm install redux react-redux react-router-dom --save

1. Example
   1. Tạo các file cần thiết
   2. Chú ý rằng các component cần tạo các fodel nhỏ hơn để viết css cho các file cần thiết
   3. Tạo file router
      1. Chú ý với việc muốn lấy giá trị trên url thì cần truyền đối tượng **match**
   4. Chức năng custom link
      1. Nếu muốn sử dụng thì dùng navlink
      2. Còn nếu muốn custom thì cần import link và router
   5. Tích hợp redux
      1. Tại index.js thì cần import nhiều thứ createStore ….
      2. Tạo dữ liệu mẫu trên store
      3. Từ page : ProductListPage connect tới store để lấy dữ liệu ra
   6. Tìm hiểu thêm về 1 số thư viện hỗ trợ để kết nối tới api
      1. Fetch API hỗ trợ trong javaScript
      2. Isomorphich-fetch : cái này gg cũng ra
      3. Axios : (dùng tk này)
         1. Chạy json-server –watch db.json
         2. Đọc phần example của nó => import nó vào mà sử dụng
         3. Giải quyết vấn đề đồng bộ và bất đồng bộ
            1. Tạm thời tạo ra 1 cái state trong constructor
            2. Sau đó dử dụng 1 cái lifecyle hook => componentDidMount() nó sẽ được gọi sau lần render đầu tiên
         4. Chức năng thêm sản phẩm
            1. Thay button thêm bằng cái link và trỏ tời router thêm là ok
            2. Tại productActionPage là cái form thì dùng nó để lấy giá trị từ các ô input ra
            3. Lấy được rồi thì connect lên server rồi thêm dữ liệu đó vào
            4. Khi thêm dữ liệu đó vào thì chuyển trang tương ứng thì có cách là sử dụng **redirect** nhưng nó lại dùng trong phần render còn ngoài render thì phải dùng 1 đối tượng khác đó là tk history => hãy truyền nó trong file router

Và sử dụng history.goBack()

* + - * 1. Chức năng xóa

Bắt sự kiện và truyền vào cho nó cái id

Sử dụng confilm để hiển thị câu thông báo xác nhận lại

Chú ý cần có thêm //eslint-disable-line

Tai đây (ProductItem) thực hiện gọi lên server để xóa sp là xong => nhưng mà ko hay

Sẽ thực hiện ở productlistPage vì productListPage là tk cha và truyền cho tk con thông qua 1 cái props

Xóa xong rồi thì chỉ cần xóa phần tử ở trong mảng là xong mảng **product** ấy

* + - * 1. Chức năng sửa

Thì dùng tham số **match** để lấy giá trị trên url

Click vào button đó lấy dc id đó ra và dùng thẻ link để chuyển route

Truyền thêm cho nó cái **history để**  gọi nó trở về router mong muốn

Tiếp theo khi vào router dạng : [**http://localhost:4000/product/4/edit**](http://localhost:4000/product/4/edit) **thì lấy id và** đổ ra nghĩa là phải gọi lên url để lấy dữ liệu về nghĩa là gọi lên server

**=> phải dùng DiDmount tron g**  ProductActionPages và gọi API set cái pages cho nó thì dữ liệu sẽ được đổ ra ở dưới thôi huhu

**Kiểm** tra khi ấn vào nút save thì trường hợp nào là thêm mới, trường hợp nào là cập nhập

**Ng**hĩa là khi có id rồi nghĩa là trường hợp edit

* 1. Vì dùng tk aoxi thì k dùng redux lên k khuyến cáo dùng nó

**Giờ**  phải xây dựng nó với redux : khi gọi lên sever thì danh sách các sản phẩm phải lưu vào store

**Tạo c**ác const

**Các**  hằng số

**Reducer** để xử lý

**mapDispatchToProp :**  lưu lên store

Tại trang gọi lên server để lấy các danh sách về chính là ProductListPages

Tại đây dischpath 1 action để gửi dữ liệu lên store

* Chú ý viết 1 action để gọi lên server lấy dữ liệu về
* Action thứ 2 lấy dữ liệu sau khi gọi api lưu lên store
* Sẽ gặp lỗi middleware
  + Async Action : là action chưa sẵn sàng khi mình gọi nó
  + Sử dụng redux-thunk để trì hoãn việc dispatch => đợi nó fetch xong dữ liệu mới dispatch
  + Sử dụng redux-thunk : npm install redux-thunk –save
    - Import thunk
    - import {applyMiddleware} from 'redux';
    - bơm nó vào cái store trong file index,js

Nâng cấp chức năng xóa

Hàm onDelete sẽ dischpatch 1 cái action

Gồm 2 hành động : xóa phần tử trên server và trong

Chức năng thêm sản phẩm

Khi thêm thì k gọi api 1 cách trực tiếp mà phải dichpath 1 cái action và gửi reques lên server và thêm thành công thì gửi dữ liệu vào store

Nâng cấp chức năng cập nhập sản phẩm

* + trong reducer thì tạo **itemEditing** Để quản lý các sản phẩm đang edit
  + sử dụng **conponentWillReceiveProps** : khi nhận những cái props thì mới gọi cái tk nay trong ProductActionPages và nó sẽ đổ vào form
* Khắc phục thẻ wapper bên ngoài tránh việc sinh ra div dư thừa
  + [<h1 key1> ,<h2 key = 2>]
  + Bọc các thành phần bên trong <React.Fragment></ React.Fragment> : nhớ import nó

Chương 07 : project tổng kết kết hợp vs file base

1. Kiến thức yêu cầu
   1. ReactJS
   2. ReactRouter
   3. Redux
   4. Bootstrap
   5. Firebase
   6. Notify(react Notify)
2. Chức năng cần có
   1. Login
      1. Email + Pass (fire base)
      2. Đăng nhập thành công thì Fire base sẽ gửi về thông tin và lưu vào Save Redux để chia sẻ sang toàn bộ project
      3. Error : hiển thị Form Login
   2. logout(signin)
      1. Click thì nó thực hiện logout
         1. Xóa thông tin user trong redux là ok
   3. Register(signup)
      1. Cần cung cấp User , Email , Pass
      2. Authendication trong fire base để check
      3. Data base
   4. Task
      1. TaskDoing đang thực hiện
         1. Hiển thị danh sách list
         2. Add
         3. Completed (khi click vào đây thì nó đẩy qua cái khia)
      2. TaskCompleteed : task đã thực hiện xong
         1. List
         2. Delete
         3. Clear All
   5. Permission
      1. 2 nhóm người
         1. Nhóm admin
         2. Nhóm người dùng bình thường
   6. Notify (xây dựng câu thông báo)
3. Xây dựng
   1. Tìm hiểu về fire base
   2. Gg reactjs use fire base
      1. Đọc nó : chú ý cần tìm cái mảng cấu hình : trong setting
      2. Tìm npm fire base : npm install --save firebase
         1. Coppy những thông số cấu hình
         2. Tạo 1 tập tin firebase.js để chứa những thông số cấu hình
         3. Import vào file index.js
            1. Log filebaseApp ra xem được ko nếu được là ok
   3. Xây dựng Task Page – Task Doing
      1. Vào document tìm các api
      2. Mục reatime database
         1. Tạo task
            1. Id
         2. Đọc phần read and write Data
            1. var database = firebase.database(); đặt nó vào sau file trong index.js
            2. đọc tiếp phần bên dưới
      3. thực hiện ấn vào nút add thì công việc đó thêm vào bảng task và cái địa chỉ email cũng được thêm vào đây
         1. khi ấn nut add thì hãy in ra thông tin người dùng nhập ở phần log dựa vào
            1. component FormAddTassk :

Rất đơn giản là sử dụng form thì sẽ lấy được giá trị của nó ra

Giờ chỉ việc dùng giá trị đó đẩy lên tk fire base đọc phần read and write Data

Firebase.database().ref(‘users/’) : tham chiếu tới bảng đó

Quay lại form add chỉ cần import nó vào

Nhận thấy là method get thì cần id lên gg firebase javascrip addItem => method push

* + - * 1. Hiển thị các công việc ra tại **ComPonent Task Doing List**

Lấy dữ liệu đổ vào cái tk items rồi map nó ra

**Vậy**  trước khi đổ dữ liệu ra thì phải lấy được dữ liệu ra nghĩa là phải sử dụng trong componentWillMount thực hiện

**Vi**ết cái gì trong này : gg fire base list item là ra cái .on để làm việc

**Thực hiện** map nó ra và truyền sang cái **TaskDoingItem**  rồi in nó ra hoy

* + - * 1. Khi click vào conleteed thì chuyển dữ liệu đó từ bảng này sang bảng completed

Đầu tiên click vào thì lấy dc nó ra đi đã

**Sau**  đó trong filebase.js tạo tham chiếu tới bảng mới “taskCompleted”

**Khi** ckick vào thì phải remove nó khỏi bảng task sau đó thêm nó vào bảng Task Complete

**Gg firebase remove :-> method remove : nó có method : *.child(item.key).remove();***

**Gg firebase put -> method put:**

***tasksCompletedRef.push(item);***

* + - * 1. Xây dựng chức năng notifice

**Gg reactjs bootstrap notification (react-bs-notiffier)**

Cài đặt thư viện

**Tạo**  1 component để hiển thị thông báo Notify :

Quẳng nó vào app. Js để nó hiển thị thông báo => sử dụng ALERT nhưng muốn nó hiển thị vị trí câu thông báo nhưng ko thấy đâu đọc API của nó k thầy

**Đọc ALERT CONTAINER** : thì thấy có => cho tk **AlertContaiNeir** bao lấy toàn bộ

**Truyền position="top-right"** để xem vị trí xuất hiện ntn

**Headline** là tiêu đề của câu thông báo

**Type** : kiểu của alert

**onDissmiss : click vào thì nó ẩn đi** thằng này đặc biệt vì nó dùng thông qua 1 hàm

**TimeOut**  thời gian nó ẩn đi

* + - * 1. **Hiển thị câu thông báo sau 1 hành động**

Trong Constant thì định nghĩa các câu notify sẵn có

**Notify**  sẽ là cái tk dùng chung và chia sẻ lại cho các component khác => sử dụng redux

**Khi thêm** mới 1 công việc thì thay đổi lại phần content

**Vậy thay đổi** khi hành động thực hiện 1 công việc thay đổi nghĩa là sẽ thay đổi ở các component

**Ví dụ FormAddTask** connect lên store và thực hiện hành động

**Truyền vào**  style là gì, title là gì và … là gì

* + - * 1. **Khi**  lick vào button completed thì cũng ok rồi hiển thị ra thông báo đó
        2. **Cách viết mới**

**TaskDoingList : có**

**TaskDoingITem**

**FormAddTassk**

**Cả 2 tk con đều** thực hiện connect tới redux và thực hiện việc thay đổi câu thông báo thì có 2 cách connect từng tk con nhưng k hay

**Mà chỉ cần connect từ tk cha và truyền vào cho tk con 1 cái props** changeNotify={this.props.changeNotify

**ở phía con sử dụng cái Props đó thôi**

* + - 1. **chức năng task Finish**
         * **trong tập tin firebase phải tham chiếu tới bảng đó đã**
         * **tại tập tin component taskfinish.js là nơi đổ dữ liệu ra**
         * **tại componentWillMount() thì thực hiện lấy dữ liệu từ firebase ra**
         * **in nó ra xem được ko => được rồi thì thực hiện đổ nó ra ngoài bằng method showElementBody()**
         * **thực hiện gắn dữ liệu bằng task finnish item trước**
         * **tiếp tục vs tk taskfinishItemAdmin=> khi click vào nút delete thì in ra được tk id của phần tử đó dễ mà**
         * **in ra được rồi thì giờ thực hiện chức năng xóa 1 công việc**

**sử dụng pmethod remove để xóa thôi**

**thực hiện câu thông báo bằng cách dùng tk cha kết nối tới redux rồi truyền nó cho tk con**

* + - * 1. **Chức năng ClearAll(xóa hết)**

**Hiển thị câu thông báo**

**Tại cái sự kiện ấn vào cái nút clearAll thì chỉ việc .set([]); là nó xóa hết thôi**

* + - * 1. **Xây dựng trang đăng kí**

**Sử dụng api của firebase**

**Tìm hiểu tk authendication**

**Có method sign up new Users có 1 cái phương thức truyền vào địa chỉ email vs password**

**Khi click vào button submit ở formSignUp thì lấy thông tin người dùng họ nhập vào và dùng phương thức mà firebase hỗ trợ CreateUser gì gì đó (cần kéo vào tk firebase) để dùng nó . cái tk catch để in lỗi đó ra thông qua cái tk notefile nhưng mà cần phải lấy cái props của tk notefile đó ra thì phải lấy thông qua cái dispath**

**Chú ý là phần sign-in method trong authentication của fire base thì muốn người dùng đăng nhập = email , fb .. gì gì đó thì cần click vào nó và active nó lên**

**.then() : sau khi đăng kí thành công rồi thì nó sẽ trả về 1 kết quả gì đó => sẽ có 1 giá trị uid (chính là id của người dùng đã đăng kí) =>**

**Lỡ như người dùng đăng kí ngoài email vs pass word mà họ đăng kí thêm 1 thông tin gì đó thì cần lấy thông tin đó lưu vào 1 cái bảng => nghĩa là cần thực hiện như sau : tại firebase thì cần tham chiếu tới cái bảng đó (bảng là users)**

**Tại component FormSignUp thì cần import cái tham chiếu của fire base đó vào rồi sử dụng**

**Chú ý khi sử dụng createUserWithEmailAndPassword thì có thể tìm hiểu về nó là API được hỗ trợ nó return về *promise*** thì có thể .then thoải mái

**Nghĩa là .promise thì chỉ việc sau khi dữ liệu được push lên thành công thì thực hiện 1 cái gì đó tại đây dử dụng method *child* và thêm vào 1 những trường mà ta cần lưu .set({website:website , isAdmin : false , uid : data.id})**

* + - * 1. **Tiến hành đăng kí thành công rồi -> giờ thực hiện lấy thông tin đó ra thì đọc doc của nó phần authentiacation -> web-> getstarted**

**onAuthStateChanged : hàm này giúp quản lý trạng thái người dùng đã đăng nhập hay chưa đăng nhập**

**coppy toàn bộ nội dung đó vào dự án của chúng ta vấn đề ở đây là coppy nó vào đâu thì cần quản lý ở tất cả các trang -> tốt nhất là quăng nó vào trang index.js là được nhớ là phải import firebase nó vào và muốn lấy thôn tin nào thì lấy**

**sau khi đăng nhập thành công rồi thì gửi tới redux cái thông tin của user và nói với redux phải lưu trạng thái của user là đăng nhập thành công còn khi k thì rơi vào trường hợp đăng nhập thất bại => tạo thêm 1 reducer User để quản lý việc này**