**Reactjs**

**Giới thiệu:**

* Xây dựng giao diện cho backend:

+ Reactjs chuyên về sử lý dữ liệu , dùng API để tương tác với dữ liệu.

+ Load dữ liệu, cách thao tác với dữ liệu như thế nào cho nhanh thông qua API.

* WHY:

+ Declative 🡪 khả năng load dữ liệu nhanh với SPA.

+ Component Based 🡪 Xem mọi thứ trong website là component.

+ Lern one, write anywhere 🡪 Học một lần viết ở đâu cũng được. Có thể sử dụng API nhiều ngôn ngữ khác nhau.

Component:

* Là một khối đóng gói bên trong gồm các thẻ HTML, props, state,...
* Tạm hiểu giống như một thẻ HTML tự định nghĩa ra.
* Chuyển đổi HTML sang dạng Component.
* Khối độc lập so với khối khác.

**Prop in Component:**

WHAT: prop là từ khóa

* Prop = properties.
* Prop là một thuộc tính của component.

WHY: là cách để truyền tham số ngoài vào trong Component.

2 Cách là function và class:

* Function: prop.thuoctinh: có tham số là props.
* Class: this.prop.thuoctinh.

**JSX:**

WHAT: Viết tắt của javascript extension, giúp ta có thể viết html trong javascript.

HOW:

* Viết dạng biến trong component với cặp ngoăc: {}.
* Viết dạng gần giống HTML: key: HTML to jsx.

**Map:**

Duyệt từng phần tử trong mảng => tạo ra mảng mới.

**Sử dụng phím tắt cho Component**

Rcc

**Lập trình tương tác với reacjs**

* onClick 🡪 C viết hoa.
* Thay vì dấu nháy kép” 🡪 dấu {}.
* Hàm phải có từ khóa this.
* Hàm không có tham sô thì không cần dấu ngoặc ( ) sau tên hàm
* Viet tương tác trên reactjs có tham số: arrow function, bind.

**State trong reactjs**

* State là biến trung gian hoạt động trong suốt quá trình thao tác với Component.
* State đóng vai trò trung gian, giữ các biến, tham số cần thiết trong lúc lập trình React js.
* State = scope trong Angular js.
* State lưu trữ được không giới hạn tham số, vì nó là kiểu object.

HOW:

* Sst( this.setState({ thuộc tính: giá trị }) ) 🡪 để set lại trạng thái state.
* Input 🡪 defaultValue: giá trị của input.

**Ref**

* Khi thay đổi props 🡪 re-render ( gọi lại hàm ) mới render lại.
* Ref lưu vào biến chung gian 🡪 hiển thị lại.
* Ref cho phép không cần render lại thông số.

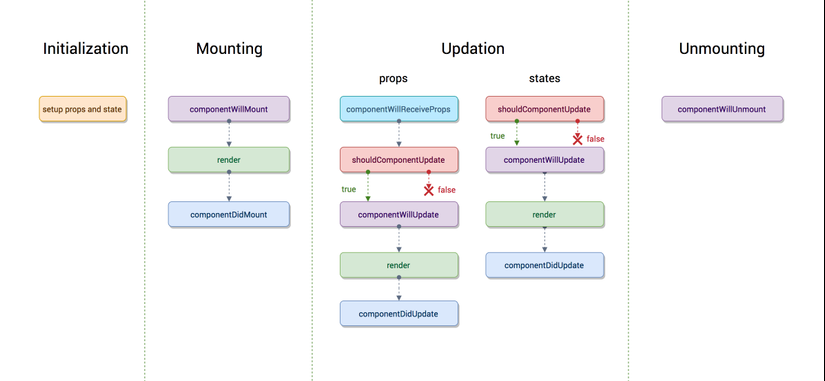
**Lifecycle trong reactjs**

WHAT:

* Lifecycle là vòng đời của component.
* Lifecycle là tập hợp các hàm mà react js đã dựng sẵn cho chúng ta sử dụng.
* Học về lifecycle là học thuộc các tên hàm mà react js đã tạo này cùng với ý nghĩa của các hàm.

HOW:

* Init: Qúa trình khởi tạo một component, đặt biến, lấy props và khai báo state.
* Mounting: Những hàm kết nối và hiện thị component ra trình duyệt. Trước khi hiển thị như thế nào, sau khi hiển thị như thế nào.
* Updation: Thay đổi giá trị tăng giảm, thao tác với giá trị. Có thể sử dụng cho props hoặc state.
* Unmounting: Hủy bỏ gắn kết.

****

Mounting:

* **Cwm 🡪** ComponentWillMount(): sẽ chạy trước xong mới render ra.
* Render(): hiển thị gia trị ra.
* **Cdm 🡪** ComponentDidMount(): sẽ chạy sau khi render ra.

Updation cho state: states: private riêng tư.

* **Scu 🡪** shouldComponentUpdate(): Kiểm tra nếu hàm trả về true thì thực hiện, false ko.
* **Cwup 🡪** componentWillMount(): Hàm trên trả về true thì hàm này chạy.
* render(): Hiển thị giá trị ra.
* **Cdup 🡪** componentDidUpdate(): Hàm chạy cuối cùng.

Updation cho props: props: public.

* **Cwr 🡪** componentWillReceiveProp(): props nhận được khi thay đổi.

**REACT router:**

WHAT:

* Cho phép SPA – single webpage application
* Thao tác chắc năng trên một trang, load một dung không cần load lại trang.

WHY:

* React js dùng cho các giao diện nhiều thao tác chức năng tương tác trê website.

HOW:

* B1: chuyển html 🡪 component react js.
* B2: Cài đặt react react-router-dom 🡪 trang reacttraning.com

+ <Router></Router>: bọc hết tài liệu.

+ <Navlink> để chỉnh active.

Truyền tham số qua url:

WHAT:

* Làm đường dẫn thân thiện dễ seo theo yêu cầu.

HOW:

* B1: Dữ liệu dạng json.
* B2: Mapping( ham map() ) dữ liệu trong trang html.

Tạo đường dẫn thân thiện với SEO trong Reactjs:

Đường dẫn thân thiện từ khóa : Slug.

Bước 1: tạo cấu trúc url theo ý muốn.

Bước 2 : convert từ tiếng việt thành Slug. Router gọi đối tượng component{ nam component }.

Bước 3: Trong trang chi tiết this.props để lấy id.

Bước 4: So khớp id bài viết với id dữ liệu.

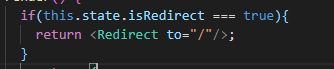
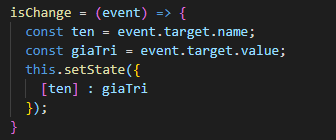
List tin liên quan:

Bước 1: tạo component mới.

Bước 2: mapping dữ liệu ra.

Xử lý form:

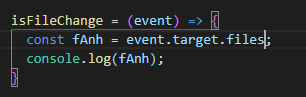
Ra redirects ( Auth ):

* Event.preventDefault( ): ngăn chặn các thao tác mặc định.
* 
* Cách dữ liệu dưới dạng mảng. 
* Lưu các trường trong state.

Thẻ select trong react js form:

State

Nhận thông tin file trong react form:



**2) Giao diện backend quản lý user bằng react js.**

B1: Tao giao diện html:

* B4-tbale: Table-striped: xen kẽ background xám, trắng.

B2: Conver thành component:

B3: Viết tương tác xử lý.

B4: Kết nối component A sang component B trong react js.

* Component con phải cùng một cha 🡪 componet con mới giao tiếp với nhau.

+ Tương tác thông qua props

B5: Tạo cơ sở dữ liệu và render dữ liệu.

* Mapping.

B6: Tìm kiếm.

* Component bố xử lý hết logic tìm kiếm ở bố.(app.js) : xử lý logic của web
* Con chỉ thu thập dữ liệu đẩy về bố.
* B1: Kết nối bố với con.
* B2: Tạo hàm ở bố có tham số là địa chỉ tìm kiếm của con( event.target.value ).
* B3: Lưu dữ liệu vào state cha.
* B4: Tìm kiếm. So sánh tền cần tìm với từng phần tử trong mảng.

Khai báo mảng rỗng trung gian lựu lại phần tử thỏa mãn.

B7: Chức năng thêm mới thành viên.

* B1: Điền nội dung ấn nút “ Thêm ” thì lấy được nội dung.
* B2: Đẩy dữ liệu vào state cha. Tạo object rông push vào state cha.
* B3: Xóa trắng form tìm kiếm.
* B4: Cài đặt package uuid tạo id.

B8: Sửa thông tin thành viên.

* B1: Kết nối ông cha cháu với nhau.
* B2: Đẩy thông tin qua from search.
* 

defaultValue giá trị mặc định.

* B3: Khi mà nhấn lưu thông tin. Sử dụng một state để lưu chữ thông tin cần sửa lấy được thông tin cần lưu gửi về về cha.
* B4: Trong app xử lý cập nhật dữ liệu. Dùng forEach so sánh id trùng thì sửa.

B9: Xóa

* Chuyền id từ con sang bố.

B10: Lưu trên localStorate.

* JSON.parse( mảng đối tượng ): chuyển về đúng kiêu.
* JSON.stringfy( mảng đối tượng ): chuyển đối tượng thành chuỗi.

**Đóng gói ứng dụng chạy trê server**

* Nếu chạy host: linux, window thì chạy:
* Code muốn đóng gói để quản lí tốt hơn: bảo mật code
* Npm run buld: nén thành file buld
* Tạo vitual host xampp:

CÁCH 2: Chạy bằng dòng lệnh:

Cài xong nhập lênh: serve –s.

**Cách đồng bộ, dowload quản lý code với Github trong VSCode**

Github: quản lý version code mã nguồn mở

Gần giống như google drive cho phép lưu trữ dữ liệu nhưng thông minh hơn chuyên dùng để lưu code.

Mỗi lần up load code sẽ tạo ra dòng thời gian, ta có thể quay về bất cứ lúc nào.

HOW:

* Git init: khởi tạo.
* Git remote add origin <duong dan>: kết nối tới đường dẫn.
* Upload: git add \*, git commit –m “message”.
* Git push origin master:

Cách tải code và chạy code.

* Git clone <duong dan>: tải source về một tài khoản bất kì.
* Npm install: tải hết module cần thiết về.

**Redux**

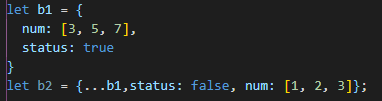
**Giới thiệu:**

What:

* Cần có một người đứng ra thâu tóm và quản lý các state này. Đó là khái niệm store trong redux.
* Store componet con thể tương tác 1 với 1.

Why:

* Toán tử 3 chấm: tách 2 vùng nhớ của 2 đối tượng khi p1 = {...p2} , p1 = [...p2].
* Ghi đè và thay đổi với toán tử 3 chấm



**Cấu trúc của một store Redux:**

Action:

* Những chỉ thị bắt buộc phải có thuộc tính type

+ type: tên chỉ thị. Vd: action.chiso, action.key...

+ item: phần tử thao tác.

* Action không trực tiếp thực hiện mà chỉ đưa ra các chỉ thị.
* Reduce là người thực hiện chi tiết.

Reducer:

* Là một hành động tính toán ra state tiếp theo 🡪 có 2 tham số:

+ old state: state trước đó.

+ chỉ thị ( action ): dùng để thay đổi state cũ và tính ra state mới

* Khi một reducer nhập vào tham số là action thì:

+ 1.action có thể đặt tên thông qua action.type.

+ 2.tác dụng của action là chỉ thị để tính ra state tiếp theo.

State:

* State đi kèm với reducer lưu trạng thái của ứng dụng.

Store:

* Là một đối tượng quản lí reduce và state vì state là private.
* Hàm khởi tạo store 
* Chỉ thị thêm:



* Chỉ thị xóa bằng filter: 
* Store1.getState(): truy cập trực tiếp vào state.

 gọi tên hành động action để reducer thực hiện. 

* Hàm Subscribe ( theo dõi ) trong store:

+ được kích hoạt mỗi khi state được thay đổi.

+ state này là state của store.



**Cài đặt plugin redux cho visual code:**

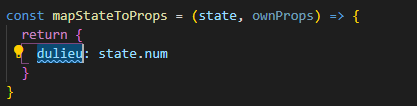
React Redux ES6 Snippets

**Kết hợp react và redux:**

Cài 2 module:

* Npm install redux.
* Npm install react-redux.

Hàm mapStateToProps:

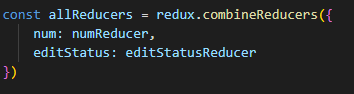
* Store sẽ kết nối với các component thông qua props
* Biến đổi tất cả bên trong store thành props
* file app.js.
*  (App) 2 chứa component muốn chứa store. Connect là hàm hết nối.
* Thêm  file index.js provider quản lý app , provider cung cấp store cho toàn bộ ứng dụng. Store props bắt buộc của provider.

Chia tách và kết hợp Reducer:

What:

* Trong một ứng dụng lớn sẽ có rất nhiều thao tác với reducer, action vì vậy cần tách nhỏ nó ra

How:

* combineReducers: 

Tách các reducer nhỏ ra 🡪 khai báo trong hàm combineReducers.

**Component con gọi store.**

Khai báo hàm mapStateToProps.

**Hàm mapDispatchToProps**

Có 2 cách để thực thi một action trong store:

C1: viết một hàm trong component con gọi hàm dispatch({});

C2: sử dụng mapDispatchToProps: truyền action dưới dạng Props

**Tổng kết cách viết code bằng Redux.**

Có module redux:

B1: chuyển đổi HTML sang Component

B2: trạng thái, hàm lưu hết trong store

B2: sử dụng hàm mapStateToProps và mapDispatchToProps để component kết nối và sử dụng.

**3)Project Note**

B1: Tạo giao diện html ứng dụng.

**FireBase:**

* web lớn dùng api:
* web nhỏ dùng Fibase:

+ Cơ sở dữ liệu trực tuyến của Google 🡪 không cần cài đặt.

+ Cũng không cần API.

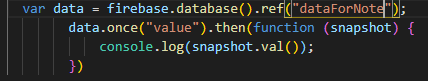
+ React js giao tiếp trực tiếp với filebase.

WHY:

* Lưu dữ liệu online.
* Giải pháp chỉ cần host không cần database.
* Tối đa 5G free.
* Thích hợp phát triển app, các ựng dụng web nhỏ.
* Realtime( thời gian thực ).

HOW:

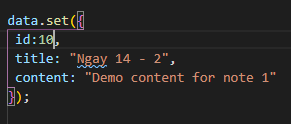
* Tạo project.
* Npm install firebase.
* Lệnh cơ bản giao tiếp với firebase: thêm mới dữ liệu vào firebase thông qua reacjs( trực tiếp ).

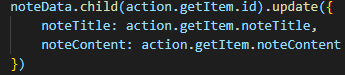
+ lấy dữ liệu 

Data là biến chưa tất cả dữ liệu, ref gọi dữ liệu ở đây là bảng.

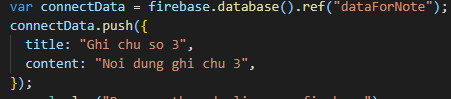
Once mội khi lấy giá trị then in ra.

Cách 2 

+ sửa: .



+ Thêm bằng cách 2: cách 1: set, cách 2 viết hàm push().



+ xóa dữ liệu. 

Kết nối firebase với reactjs

* Npm
* Vào setting.

Cách làm với redux store

* mapStateToProps: muốn sử dụng state nào strong store.
* mapDispatchToProps: Muốn dùng hàm nào sử lý thực thi.
* Connect vào component:
* B2: truyền tham số component vào store: sử dụng action.tham số.
* Lấy dữ liệu ra từ firebase:

+ key: đóng vai trò là id.

+ val: nội dung.

1. Lấy dữ liệu ra, đặt biến trung gian là state.
2. Đem lữ liệu ra, thì lấy từ state.

Hiển thị thông báo các thao tác bằng plugin notify

npm install react-bs-notifier --save

**4) Reactjs kết hợp với NodeJs**

NodeJs: API tương tác với dữ liệu trả về cho reactJs sử dụng.

* Cài đặt express framwork.
* Express ten –e.
* Vào trong file 🡪 npm start 🡪 port 3000
* Đổi cổng vào file bind.
* Kết nối NodeJs với xampp.
* res.send(results);
* Từ khóa cấu hình nodejs cho phép truy cập: access control allow method nodejs/

ReactJs: frontend.

* Tạo giao diện react.
* Lấy dữ liệu từ API.
* Module axios ( npm install axios ) hỗ trợ việc gửi nhận thông tin bằng phương thức get, post...
* Hứng dữ liệu từ nodejs
* Gửi thông tin từ reactjs lên nodejs.