1. **Khái niệm về Redux**

**Npm install redux –save (lưu vào file package.json)**

**Npm install react-redux --save**

**Npm install firebase --save**

Gồm:

1. **Action**:

bắt buộc có thuộc tính **type – tên chỉ thị.**

**Item:** phần tử đê thao tác

* **Action** k trực tiếp thao tác, mà để đưa ra chỉ thị(thao tác là reducer)

1. **Reducer**: là 1 hành động. Trả về 1 state, quản lý State

Có 2 tham số: old state, chỉ thị(action)

1. **State**: là các trang thái
2. **Store**: cái quản lý State và Reducer, state là privacy. Nên phải dùng store để thay đổi

\* Note: cơ bản thì Action sẽ tạo ra 1 chỉ thị > Reducer sẽ thực hiện chỉ thị > thông qua Store để gọi/chạy cái Reducer đó.(dùng **dispatch**()). Muốn lấy state thì cũng thông qua store và dùng **getState**().

1. **Cách chạy của redux**

**Var** redux = require(‘redux’);

**Tạo reducer:**

**Var reducer1 =** (state, action) => {}

**Khởi tạo store:**

Var store = redux.createStore(reducer);

Truy cập vào store:

store.getState

chạy reducer:

store.dispatch({type:”....”})

1. **hiểu hơn về Action**

**có** thể tự đặt properties cho Action. Nhưng property: **type** là bắt buộc

store.dispatch({

type:”....”,

num: 2

})

1. **Hiểu hơn về Store**

Hàm subscribe(): mỗi lần thay đổi state. (dispatch) thì sẽ chạy hàm này. Có thể ứng dụng làm những func chạy sau khi state bị thay đổi.

* Store cho phép định nghĩa thêm action.[*props*]

1. **Chuyển State sang Props**

* State là private nên ko thể truyển qua các Com. Phải dùng Props.

const mapStateToProps = (state, ownProps) => {

return

}

Lúc này đã biến State thành Props trong App.js nhưng phải connect được.

Dùng hàm connect –trong react-redux. (nhớ import {connect} from ‘react-redux’)

Export default connect(mapStateToProps)(App) \\ có 2 parathesises () ()

Trong index.js : <Provider store=”store”> \\ nhớ import store

<App />

</Provider>

1. **Chia/ tách Reducer**

**Khi action muốn**  thay đổi cái properties nào thì chia Reducer ra .

vd: thay đổi arr > 1 Reducer, thay đổi boolean > 1 Reducer

dùng **redux.combineReducers({})**

**có t**hể viết trong 1 file hoặc nhiều file. Rồi lun nhớ Import vào

1. **gọi hàm/Reducer trong store từ ComCon**

**cách 1:**

khai báo 1 func trong comCon gọi đến Action(dispatch) trong Store và thực thi.

useFunctionInStore = () => {

const {dispatch} = this.props;

dispatch: ({type: “....”})

}

Trong comCOn này gọi ra onClick={() => this.useFunctionInStore() }

**Cách 2:**Dùng mapDispatchToProps()

**Const** mapDispatchToProps = (dispatch, ownProps) =>{

Return {

funcName: (*param*) => {

dispatch({

type:”ADD\_DATA”,  
 *param*}) \\\\ **đây chính là dispatch Action**

}

}

} \\\\\\\\\ có param: Cách này có thể truyền từ ComCon đến Store.

1. **Tổng kết**
2. Chuyển HTML sang thành các Components
3. Define các dữ liệu, function handle(buttons, input),các biến trạng thái, trong store.js
4. Dùng mapStateToProps() và mapDispatchToProps() truyển cho ComCon qua Props
5. **FIREBASE**

**Thông thường:**

* Get/lấy dữ liệu trong React > gọi API (nodeJS,PHP,...). trả về 1 đường dẫn, chứa data .JSON
* Edit/sửa data >gửi đến API >để API update lên data.
* Delete/Xóa data > tìm id cần xóa > gửi đến API > API update lại data.
* Add/Thêm mới:

Dùng Firebase:

* Dùng trực tiếp. Là database free của google.
* Luôn nhớ các lệnh cơ bản: THêm, Sửa, Xóa, Xem
* Tạo database trên Firebase. Export JSON

Datasnapshot: firebase.database().ref(‘[tenthumuc]’);

* Get/lấy ra dữ liệu: [var].once(‘value’).then(function(snapshot){console.log(snapshot.val())})
* Edit/Sửa data: [var].set({})
* Add/Thêm data: hàm .set(), hoặc push()
  + Set: phải chi ra id/tên của nó
  + Push: tự động tạo id
* Delete/xóa: data.child(‘[id]’).remove()

\*State trạng thái: isShow: true/false. Define trong state(là cái **nameInitialState** lun) của Store. Kết nối Store với comCon. Xét condition của isShow. Dùng ... operator để copy cả State rồi ghi đè lên.(obj{...state},arr[...state] )