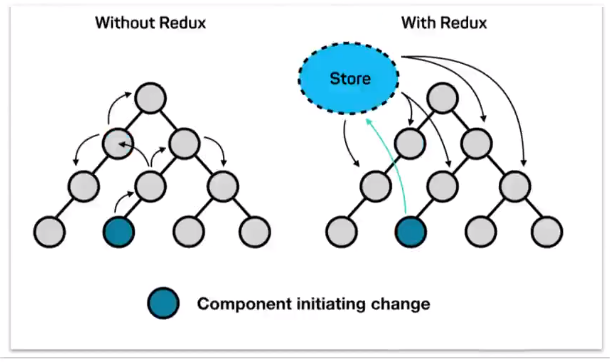
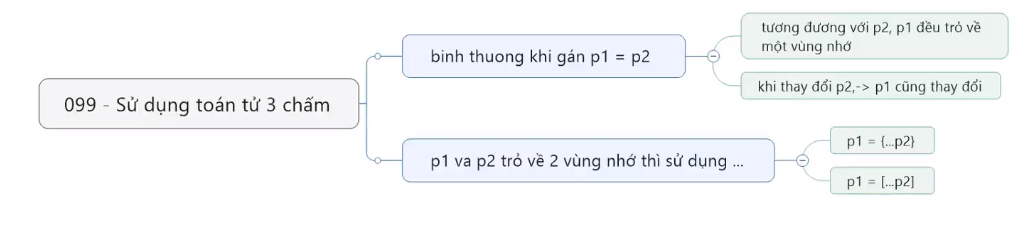
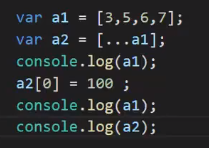
REDUX:

* WHY: 
  + Dễ dàng kiểm soát được các state, props của tất cả các component thông qua STORE của Redux.

TOÁN TỬ 3 CHẤM (…):





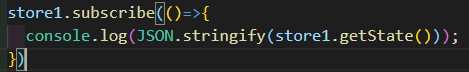
CẤU TRÚC CỦA 1 STORE TRONG REDUX:

* Gồm 4 phần: ACTION, REDUCER, STATE, STORE:
  + Action: là những chỉ thị gồm thuộc tính bắt buộc là type:
    - Type: tên chỉ thị.
    - Item: Phần thử thao tác.
    - Action ko trực tiếp thực hiện mà chỉ đưa ra các chỉ thị.
    - Reducer: Người thực hiện chi tiết.
  + Reducer: là 1 hành động:
    - Hành động này trả về State.
    - Có 2 tham số:
      * Old state
      * Chỉ thị (action): dùng để thay đổi state cũ và tính ra state mới.
  + State.
  + Store:
    - Là một đối tượng quản lý reducer và state,vì state là private, ko thể trực tiếp thay đổi được, mà phải thông qua đối tượng quản lý của nó là store.
    - Hàm khởi tạo Store: var store1 = redux.createStore(reducer1).
    - Hàm truy cập state: store1.getState()

CÀI ĐẶT và CÁC CHỨC NĂNG THÊM, XÓA, SỬA State cơ bản:

* Npm install redux –save.
* Khai báo var redux = require(‘redux’).
* Thêm , xóa, sửa State cơ bản:
* var redux = require('redux');
* var oldState = {
* num: ["man hình", "chuột", "bàn phím"],
* editStatus: true
* }
* var reducer1 = (state = oldState, action) => {
* switch (action.type) {
* case "CHANGE\_STATE":
* return { ...state, editStatus: !state.editStatus }
* break;
* case "ADD\_NEW\_STATE":
* return {...state,num:[...state.num,action.newItem]};
* break;
* case "DELETE\_STATE":
* return {...state,num:state.num.filter((value,index) => index != action.number)};
* break;
* default:
* return state;
* break;
* }
* return state;
* }
* var store1 = redux.createStore(reducer1);
* console.log("state ban đâu:");
* console.log(store1.getState());
* store1.dispatch({type:"CHANGE\_STATE"})
* console.log("State sau khi CHANGE:");
* console.log(store1.getState());
* store1.dispatch({type:"ADD\_NEW\_STATES",
* newItem: "Tai nges"
* })
* console.log("State sau khi ADD NEW STATE:");
* console.log(store1.getState());
* store1.dispatch({
* type:"DELETE\_STATE",
* number: 0
* })
* console.log("STATE sau khi DELETE:-");
* console.log(store1.getState());
  + s

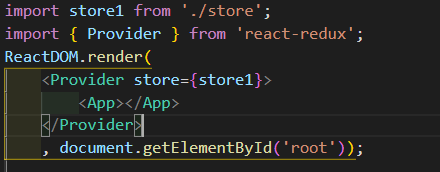
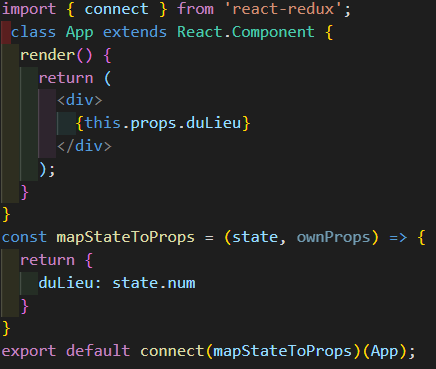
HÀM SUBSCRIBE TRONG REDUX:

* Tác dụng:
  + Được kích hoạt mỗi khi State được thay đổi.
  + Chuyên dùng để theo dõi sự thay đổi của State trong quá trình Dispatch().
* Ví dụ:
  + 

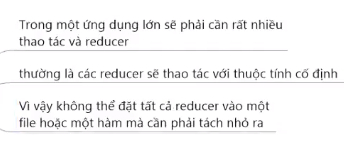
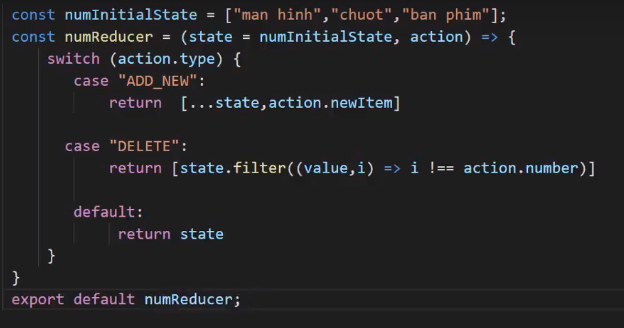
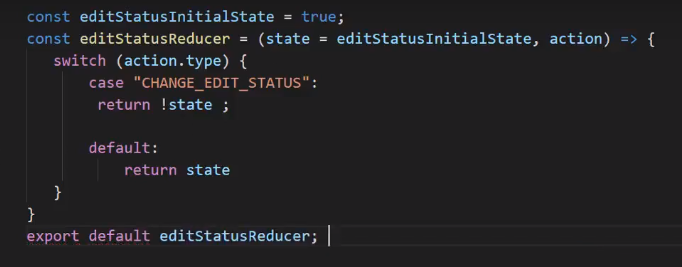
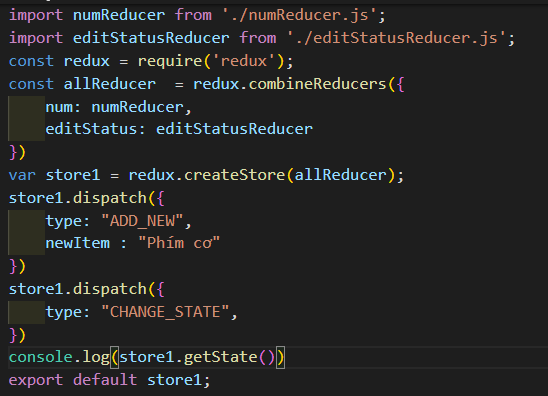
CÀI ĐẶT CÁC PLUG IN REDUX:

* React redux es7 snippet.
* React redux es6 snippet.

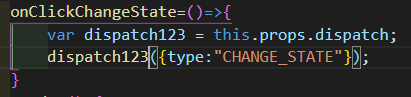
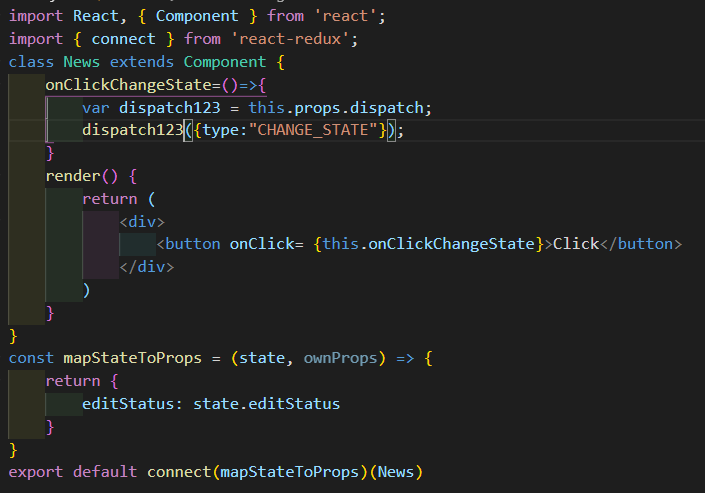
CÁCH KẾT HỢP,KẾT NỐI REACT vs REDUX: Truy xuất dữ liệu trong Store từ các component Con:

* Cài đặt 2 module:
  + Npm install redux
  + Npm install react-redux
* Kết hợp:
  + Tạo ra file ‘store.js’ trong folder “src” và export biến store ấy.
  + Trong file “index.js”:
    - Import Store1 from “./store’;
    - Import {Provider} from ‘react-redux’;
    - Đặt <App> vào trong <Provider> , và truyền vào props ‘store’ = {store1} (đây là yêu cầu của redux).
    - 
  + Trong file “App.js”:
    - Import {connect} from ‘react-redux’;
    - Export default connect(mapStateToProps)(App);
    - Gọi các hàm cơ bản của redux để thao tác, kết nối với các State của Store như các hàm :
      * mapStateToProps().
      * mapDispatchToProps().
      * mapDispatchToPropsBind().
      * 
      * Coi như trong <App> này có cái props this.props.duLieu được lấy từ Store rồi.

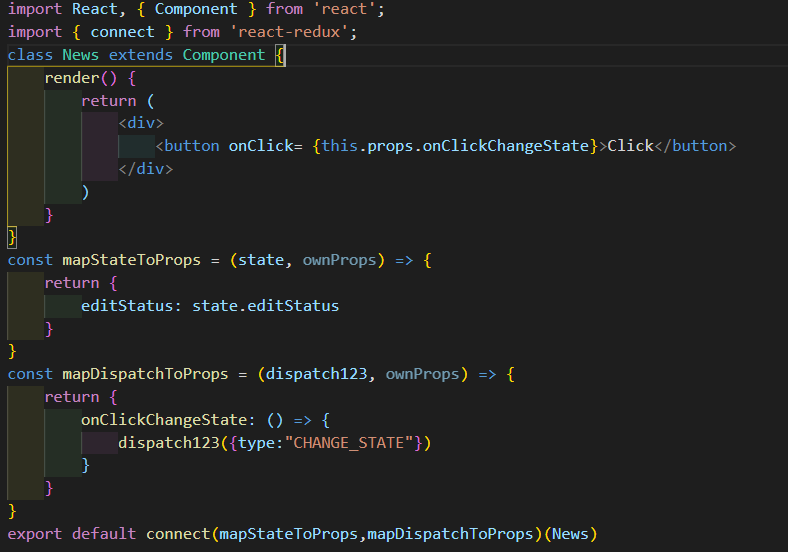
CÁCH CHIA TÁCH REDUCER THÀNH CÁC FILE, ỨNG VỚI MỖI THUỘC TÍNH Ở TRONG STATE:

* WHY:
  + 
* HOW:
  + Tạo folder ‘Reducer’:
    - Tạo file numReducer:
      * 
    - Tạo file editStatusRedecer.js:
      * 
    - Ở file ‘Store.js’:
      * Import 2 reducer ‘editStatus’ và ‘num’.
      * Dùng hàm redux.combineReducer({}) để kết hợp 2 reducer kia lại thành 1 reducer rồi tạo Store quản lý reducer tổng này lại.
      * 

CÁCH GỌI HÀM TRONG STORE TỪ COMPONENT CON:

* Import hàm { connect } và export export default connect(mapStateToProps)(News)
* Khai báo biến dispatch thực thi bằng this.props.dispatch:
  + 
  + Biến ‘dispatch123’ có thể được đặt tên tùy ý
* 

HÀM DISPATCHTOPROPS:

* Công dụng cũng giống như cái trên vừa học, dùng để truyền 1 cái action từ Store đến các Component con để có thể tương tác được với State:
  + 

TỔNG KẾT CƠ BẢN:

* Thay đổi khi viết code bằng redux:
  + Qui trình ban đầu vẫn như cũ (từ HTML sang Component).
  + Định nghĩa dữ liệu, các handle function, các biến trạng thái…. Trong 1 file gọi là store.js
  + Sử dụng hàm mapStateToProps() và hàm mapDispatchToProps() để các component con kết nối và sử dụng các hàm, state và thao tác với Store.