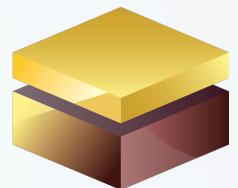
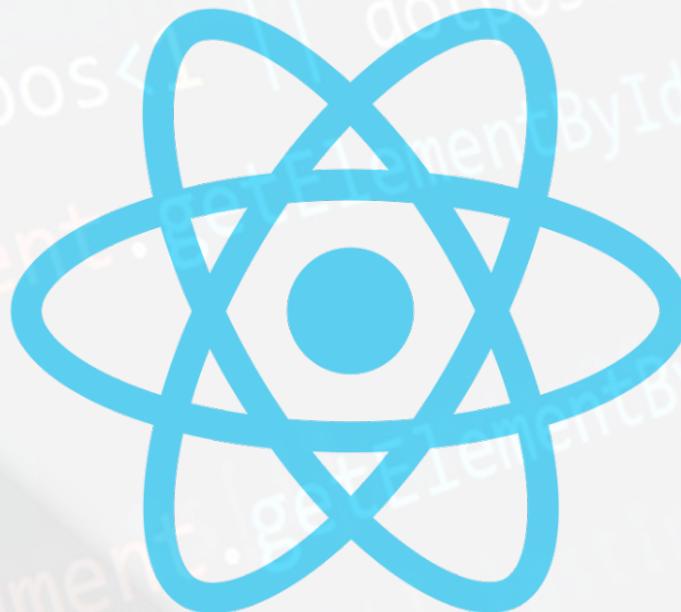


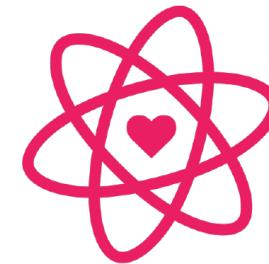
# React js



**CYBERLEARN**  
ĐÀO TẠO CHUYÊN GIA LẬP TRÌNH

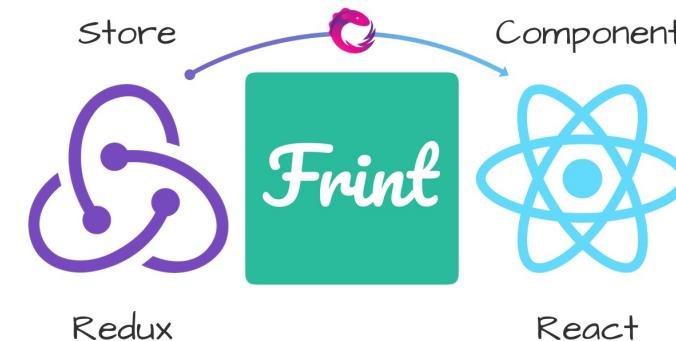
Hướng dẫn : Trương Tấn Khải

# Giới thiệu



# React JS

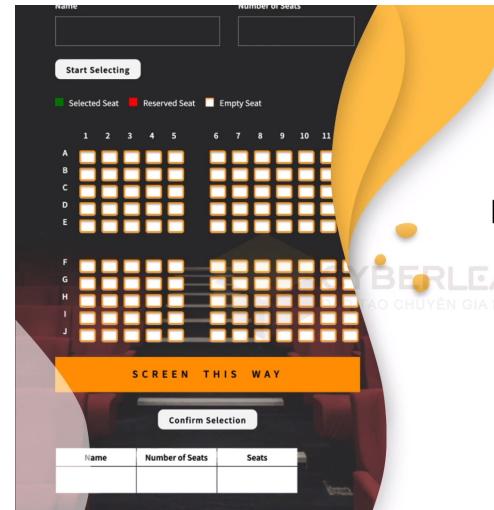
# Redux



# Giới thiệu

- Props
- Props truyền sự kiện qua Props
- Truyền dữ liệu qua component children
- Thực hành qua ví dụ chọn và xem chi tiết sản phẩm
- Phân biệt state và props
- Bài tập giỏ hàng với state props
- Redux với ví dụ xem chi tiết sản phẩm
- Redux với ví dụ giỏ hàng
- Pure component
- Redux qua ví dụ đặt vé xem phim
- Tổng kết state props, context api, redux

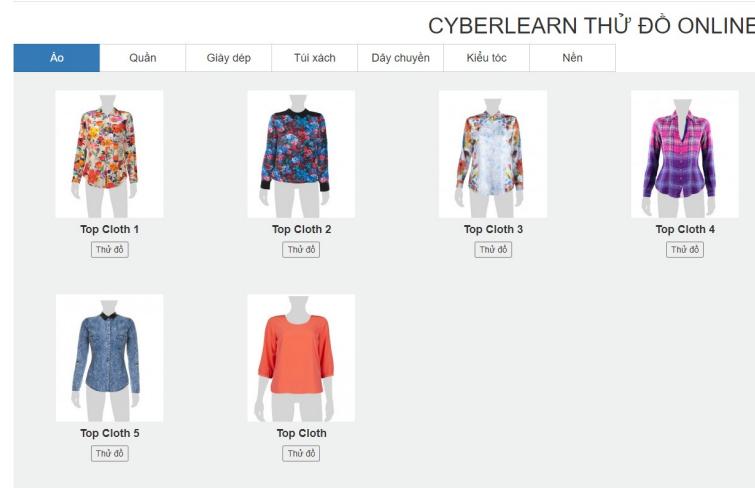
## ✓ DỰ ÁN ĐẶT VÉ XEM PHIM



Redux & Đặt vé xem phim

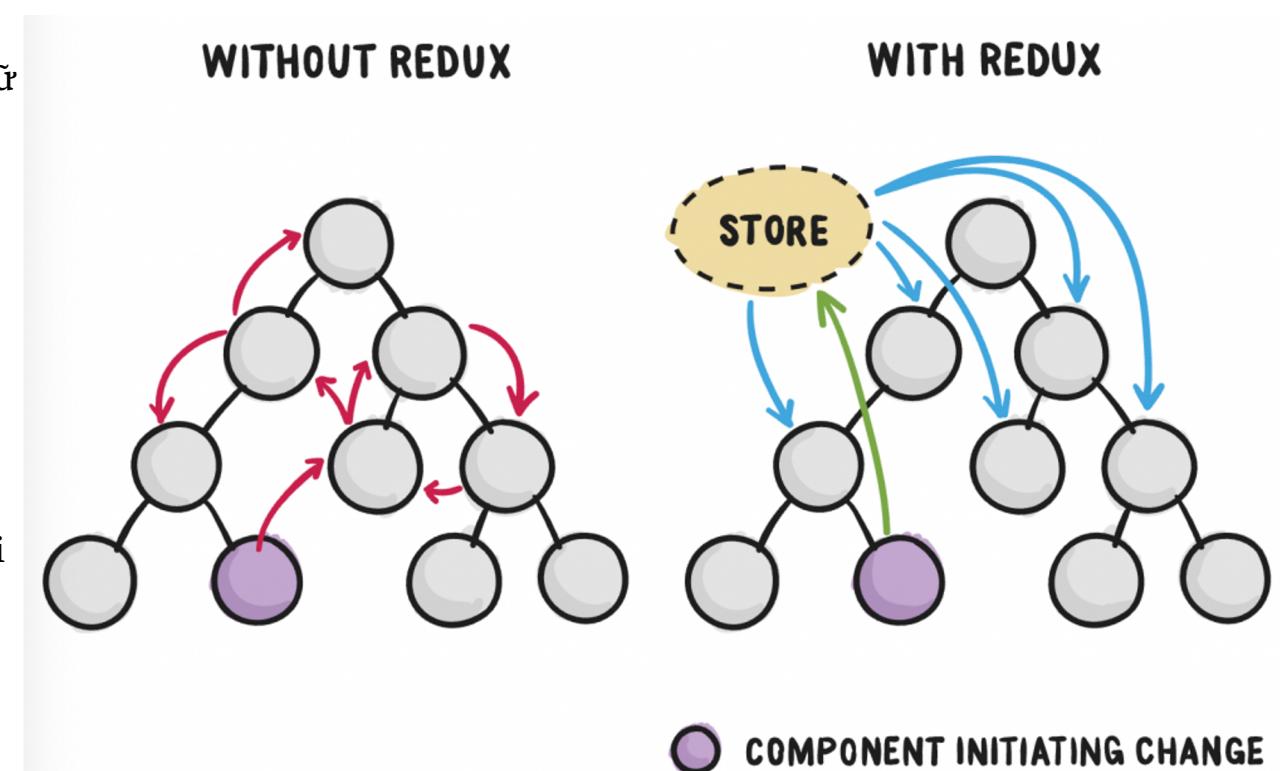
27

## ✓ DỰ ÁN THỦ ĐỒ ONLINE

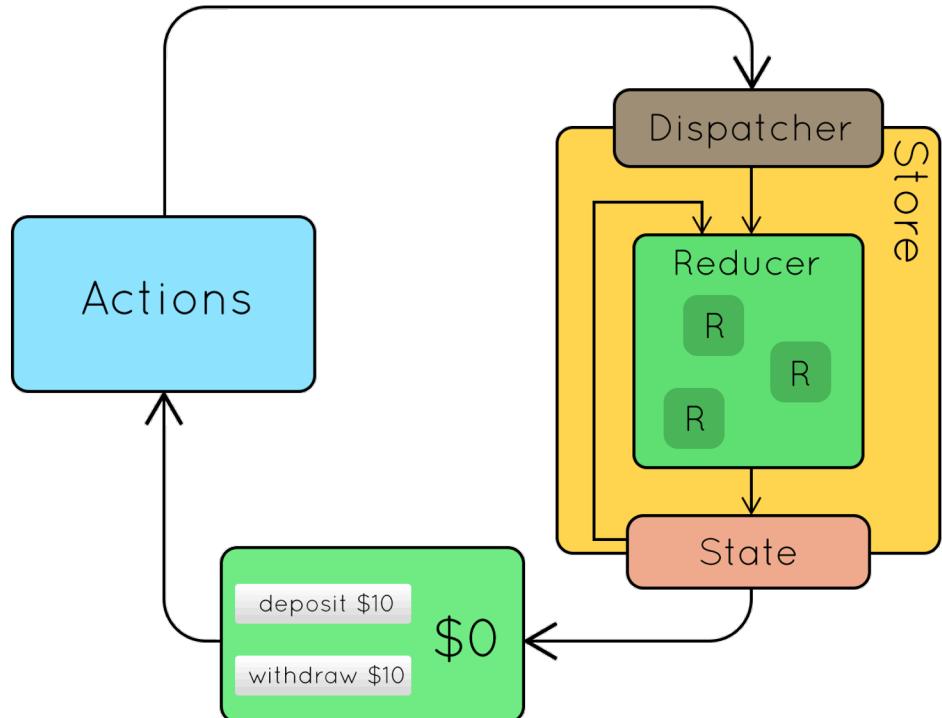


# Giới thiệu về Redux

- Bản chất làm việc với React là việc truyền dữ liệu giữa các component và thay đổi state để re-render lại giao diện component
- Redux là thư viện cung cấp cho ta một store trung tâm, lưu trữ tất cả các state, từ component muốn thay đổi state chỉ cần truy cập tới store để thay đổi
- **Ví dụ:**
  - Qua ví dụ bài tập giỏ hàng trên cho thấy để lưu trữ nguồn dữ liệu thay đổi (render lại giao diện mỗi khi setState được gọi). Ta phải xác định **component nào là nơi chứa dữ liệu thay đổi** binding, và phải xem xét thêm điều kiện nó có **chứa nút xử lý (hoặc bất kể event nào)** để gọi hàm setState không?
  - ➔ Ta phải chọn 1 component nào chứa cả 2 yếu tố trên, vì vậy dẫn đến việc truyền dữ liệu từ props cha sang con nhiều cấp.
  - ➔ Redux ra đời để khắc phục tình trạng đó, nó chỉ xét store (nơi chứa tất cả các state của component này được lưu tập trung) với component mà không phụ thuộc props từ component cha.

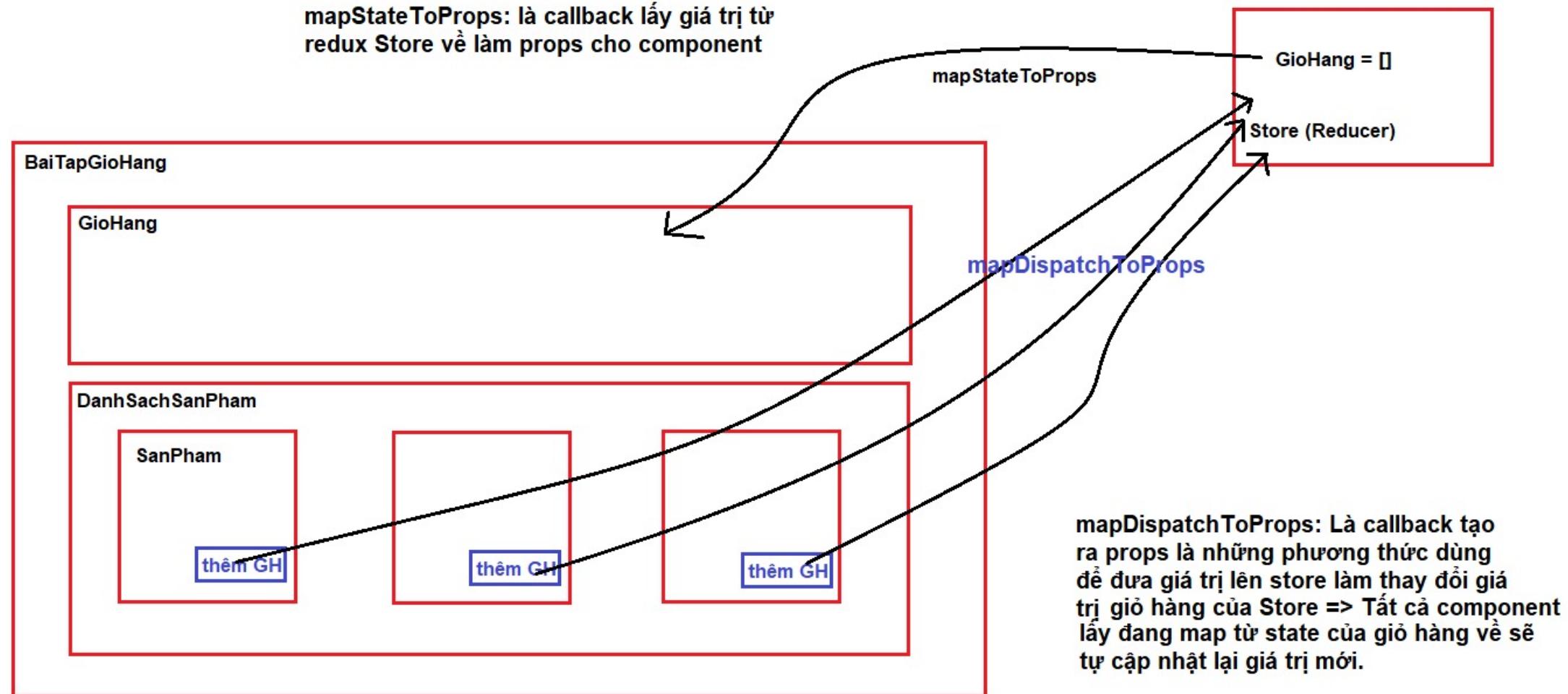


# Sơ đồ hoạt động của redux



- ❑ Ví dụ ta có 2 component Form và component DanhSachSinhVien
- ❑ Ta muốn từ component Form, sẽ thêm một sinh viên vào danh sách.
- ❑ Đầu tiên, ta sẽ để cho store lưu trữ state với thuộc tính là dssv  
state={ dssv: []}
- ❑ Từ component Form, ta sẽ gửi một action, là một object miêu tả điều chúng ta muốn làm .Ví dụ: action { type:'ADD\_USER', user: user}
- ❑ Reducer sẽ tiếp nhận và xử lý action. Tại reducer, ta sẽ có 1 Root reducer để tổng hợp, và 1 child reducer để xử lý từng thuộc tính, trong trường hợp này là **dssv**
- ❑ Tại child reducer sẽ kiểm tra action, cập nhật lại và trả ra **dssv** mới
- ❑ Tại Component DanhSachSinhVien, ta truy cập lên store để lấy **dssv** về và hiển thị ra màn hình

# Ứng dụng redux làm bài tập giờ hàng



# Cài đặt redux

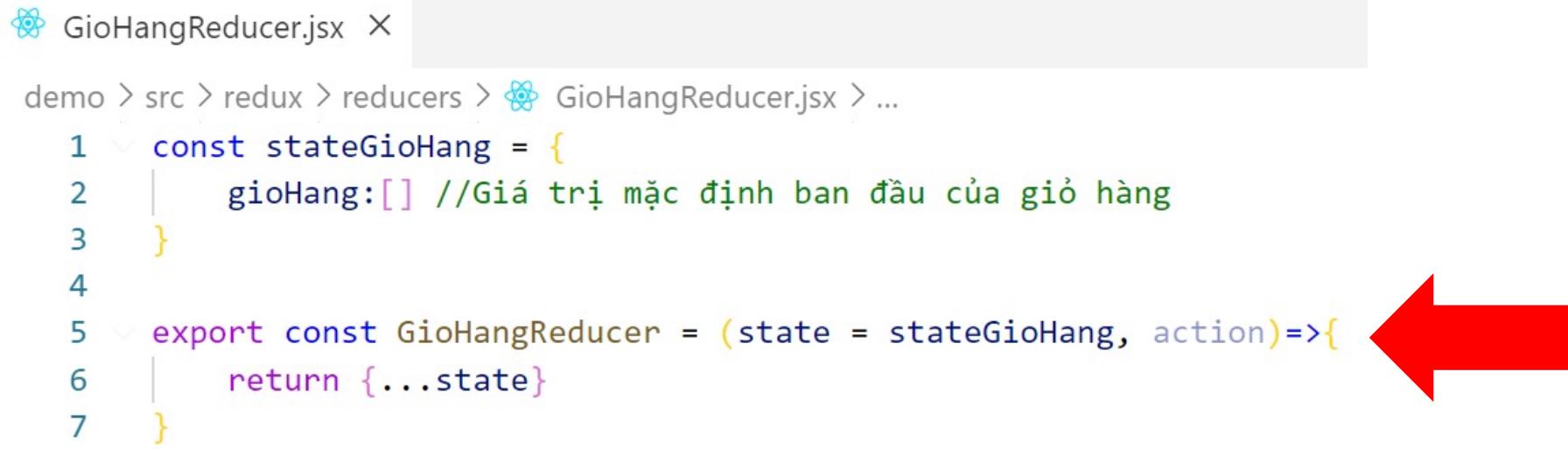
```
npm i redux --save  
npm I react-redux --save
```

## □ Cấu hình file index.js

```
1 import React from 'react';
2 import ReactDOM from 'react-dom';
3 import './index.css';
4 import App from './App';
5 import * as serviceWorker from './serviceWorker';
6
7 //Root reducer là file mình tạo ra nó là store tổng của toàn ứng dụng
8 import {rootReducer} from './redux/reducers/rootReducer';
9 //Provider là component kết nối redux store với component react
10 import {Provider} from 'react-redux';
11 import {createStore} from 'redux';
12 const store = createStore(rootReducer);
13
14 ReactDOM.render(
15   <Provider store = {store}>
16     <App />
17   </Provider>
18   ,
19   document.getElementById('root'));
```

# Cài đặt redux

## □ Cấu hình file **gioHangReducer.jsx**



```
demo > src > redux > reducers > GioHangReducer.jsx > ...
1 const stateGioHang = {
2   |   gioHang:[] //Giá trị mặc định ban đầu của giỏ hàng
3 }
4
5 export const GioHangReducer = (state = stateGioHang, action)=>{
6   |   return {...state}
7 }
```

## □ Cấu hình file **rootReducer.jsx**

```
1 import {combineReducers} from 'redux';
2 import {GioHangReducer} from './GioHangReducer';
3 export const rootReducer = combineReducers({
4   |   //Nơi sẽ chứa các store theo từng nghiệp vụ
5   |   //GioHangReducer:GioHangReducer có thể viết theo cách này
6   |   GioHangReducer //Hoặc viết theo kiểu rút gọn
7 });


```

## ❑ Thực hiện phương thức mapStateToProps lấy dữ liệu từ store về

### ❑ GioHangReducer

```
1
2 const stateGioHang = {
3     gioHang:[
4         {maSP:1,hinhAnh:'./img/vsphone.jpg',tenSP:'VinSmart Live',soLuong:1,
5          giaBan:5700000}
6     ] //Giá trị mặc định ban đầu của giỏ hàng
7 }
8 export const GioHangReducer = (state = stateGioHang, action)=>{
9     return {...state}
10 }
```

### ❑ gioHangComponent. Lấy giá trị từ store về.

```
63 //state chính là rootReducer
64 const mapStateToProps = (state) => {
65     //state.GioHangReducer => rootReducer.GioHangReducer
66     return { gioHang: state.GioHangReducer.gioHang }
67 }
68 ;
69
70 //Sử dụng hàm connect để kết nối Component và Redux Store
71 export default connect(mapStateToProps, null)(GioHangModal)
```

### ❑ rootReducer

```
1 import {combineReducers} from 'redux';
2
3 import {GioHangReducer} from './GioHangReducer';
4
5 export const rootReducer = combineReducers({
6     //Nơi sẽ chứa các store theo từng nghiệp vụ
7     //GioHangReducer:GioHangReducer có thể viết theo cách này
8     GioHangReducer //Hoặc viết theo kiểu rút gọn
9 });
10
```

Hàm kết nối giữa redux store và reactComponent

Chuyển các state của redux => các props của component

- ☐ **gioHangComponent.** Sau khi lấy kết quả lấy từ store về thông qua props ta binding dữ liệu lên giao diện dựa vào hàm map.

```
<div className="modal-body">
  <table className="table">
    <tr>
      <th>Mã sản phẩm</th>
      <th>Hình ảnh</th>
      <th>Tên sản phẩm</th>
      <th>Số lượng</th>
      <th>Đơn giá</th>
      <th>Thành tiền</th>
      <th></th>
    </tr>
    {
      this.props.gioHang.map((spGioHang, index) => {
        return <tr key={index}>
          <td>{spGioHang.maSP}</td>
          <td><img src={spGioHang.hinhAnh} width={50} height={50} /></td>
          <td>{spGioHang.tenSP}</td>
          <td><button className="btn btn-primary">+</button>{spGioHang.soLuong}<button className="btn btn-primary">-</button></td>
          <td>{spGioHang.giaBan}</td>
          <td>{spGioHang.soLuong * spGioHang.giaBan}</td>
          <th><button className="btn btn-danger">Xóa</button></th>
        </tr>
      })
    }
  </table>
</div>
<div className="modal-footer">
  <button type="button" className="btn btn-secondary" data-dismiss="modal">Đóng</button>
</div>
```

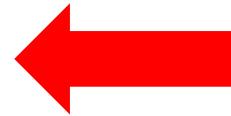
```
6  //Lưu ý bỏ export default của component
7  class GioHangModal extends Component {
8    constructor(props) {
9      super(props);
10 }
```

## ❑ Thực hiện phương thức mapDispatchToProps tạo phương thức đưa dữ liệu lên Redux store

Lưu ý : tại component bỏ export default ta chuyển export default xuống dưới hàm connect

```
4  class SanPham extends Component {  
5  
6      render() {  
7          let { sanPham } = this.props;  
8          let { xemChiTiet } = this.props;  
9          //Hoặc có thể khai báo  
10         // let {sanPham,xemChitiet} = this.props;  
11         return (  
12             <div className="card text-left">  
13                 <img className="card-img-top" src={sanPham.hinhAnh} width={170} height={300} alt={'true'} />  
14                 <div className="card-body">  
15                     <h4 className="card-title">{sanPham.tenSP}</h4>  
16                     <button className="btn btn-success" onClick={() => xemChiTiet(sanPham)}>Xem chi tiết</button>  
17                     &nbsp; <button className="btn btn-danger" onClick={() => this.props.themGioHang(sanPham)}>Thêm giỏ hàng</button>  
18                 </div>  
19             </div>  
20         )  
21     )  
22 }  
23 }  
24 }
```

Gắn phương thức  
themGioHang vừa định nghĩa  
vào sự kiện của nút button  
thêm giỏ hàng



```
26 const mapDispatchToProps = (dispatch) => {  
27     return {  
28         themGioHang: (sanPham) => {  
29             let sanPhamGioHang = {...sanPham, soLuong:1}; //Sản phẩm giỏ hàng cần thêm  
30             dispatch({  
31                 type:'THEM_GIO_HANG',  
32                 sanPhamGioHang  
33             });  
34         }  
35     }  
36 }  
37  
38 export default connect(null,mapDispatchToProps)(SanPham)
```



Tương tự props là dữ liệu lấy  
về từ store về thì ở đây hàm  
dispatch cho ta định nghĩa 1  
props là dữ 1 phương thức  
đưa dữ liệu lên store

## ❑ Thực hiện phương thức mapDispatchToProps tạo phương thức đưa dữ liệu lên Redux store

Tại **gioHangReducer** ta nhận được dữ liệu từ nút thêm giỏ hàng sau phương thức dispatch đưa lên store qua biến **action**. Ta dùng thuộc tính type để phân biệt xử lý và dùng thuộc tính giá trị(ở đây là thuộc tính **sanPhamGioHang**) để thay đổi state của Reducer gioHang (state.gioHang)

```
2  const stateGioHang = {  
3    gioHang: [  
4      { maSP: 1, hinhAnh: './img/vsphone.jpg', tenSP: 'VinSmart Live', soLuong: 1, giaBan: 5700000 }  
5    ] //Giá trị mặc định ban đầu của giỏ hàng  
6  }  
7  
8 export const GioHangReducer = (state = stateGioHang, action) => {  
9  
10  switch (action.type) {  
11    case 'THEM GIO HANG': {  
12      let gioHang = [...state.gioHang];  
13      const index = gioHang.findIndex(spGH => spGH.maSP === action.sanPhamGioHang.maSP);  
14      if (index === -1) {  
15        //Kiểm tra nếu sản phẩm chưa có trong giỏ hàng thì thêm sản phẩm đó vào giỏ hàng  
16        gioHang.push(action.sanPhamGioHang)  
17      } else {  
18        //Ngoại lệ nếu đã có trong giỏ hàng rồi thì tăng số lượng  
19        gioHang[index].soLuong += 1;  
20      }  
21      //Cập nhật lại state  
22      state.gioHang = gioHang;  
23      return { ...state }  
24    }  
25  }  
26  
27  return { ...state }  
28}
```

action gửi lên 2 thuộc tính là type, và dữ liệu => dựa vào type để xác định xử lý tương đương cho action.  
type: là thuộc tính bắt buộc

dữ liệu gửi lên từ dispatch

## ❑ Thực hiện phương thức mapDispatchToProps tạo phương thức đưa dữ liệu lên Redux store

Tại **gioHangReducer** ta nhận được dữ liệu từ nút thêm giỏ hàng sau phương thức dispatch đưa lên store qua biến **action**. Ta dùng thuộc tính type để phân biệt xử lý và dùng thuộc tính giá trị(ở đây là thuộc tính **sanPhamGioHang**) để thay đổi state của Reducer gioHang (state.gioHang). Những nơi mapStateToProps từ gioHangReducer khi thay đổi sẽ tự động cập nhật lại.

```
2 const stateGioHang = {  
3     gioHang: [  
4         { maSP: 1, hinhAnh: './img/vsphone.jpg', tenSP: 'VinSmart Live', soLuong: 1, giaBan: 5700000 }  
5     ] //Giá trị mặc định ban đầu của giỏ hàng  
6 }  
7  
8 export const GioHangReducer = (state = stateGioHang, action) => {  
9     // action gửi lên 2 thuộc tính là  
10    // type, và dữ liệu => dựa vào type  
11    // để xác định xử lý tương đương  
12    // cho action.  
13    // type: là thuộc tính bắt buộc  
14  
15    switch (action.type) {  
16        case 'THEM GIO HANG': {  
17            let gioHang = [...state.gioHang];  
18            const index = gioHang.findIndex(spGH => spGH.maSP === action.sanPhamGioHang.maSP);  
19            if (index === -1) {  
20                //Kiểm tra nếu sản phẩm chưa có trong giỏ hàng thì thêm sản phẩm đó vào giỏ hàng  
21                gioHang.push(action.sanPhamGioHang)  
22            } else {  
23                //Ngược lại nếu đã có trong giỏ hàng rồi thì tăng số lượng  
24                gioHang[index].soLuong += 1;  
25            }  
26            //Cập nhật lại state  
27            state.gioHang = gioHang;  
28            return { ...state }  
29        }  
30    }  
31  
32    return { ...state }  
33 }
```

## ❑ Thực hiện phương thức mapDispatchToProps tạo phương thức đưa dữ liệu lên Redux store

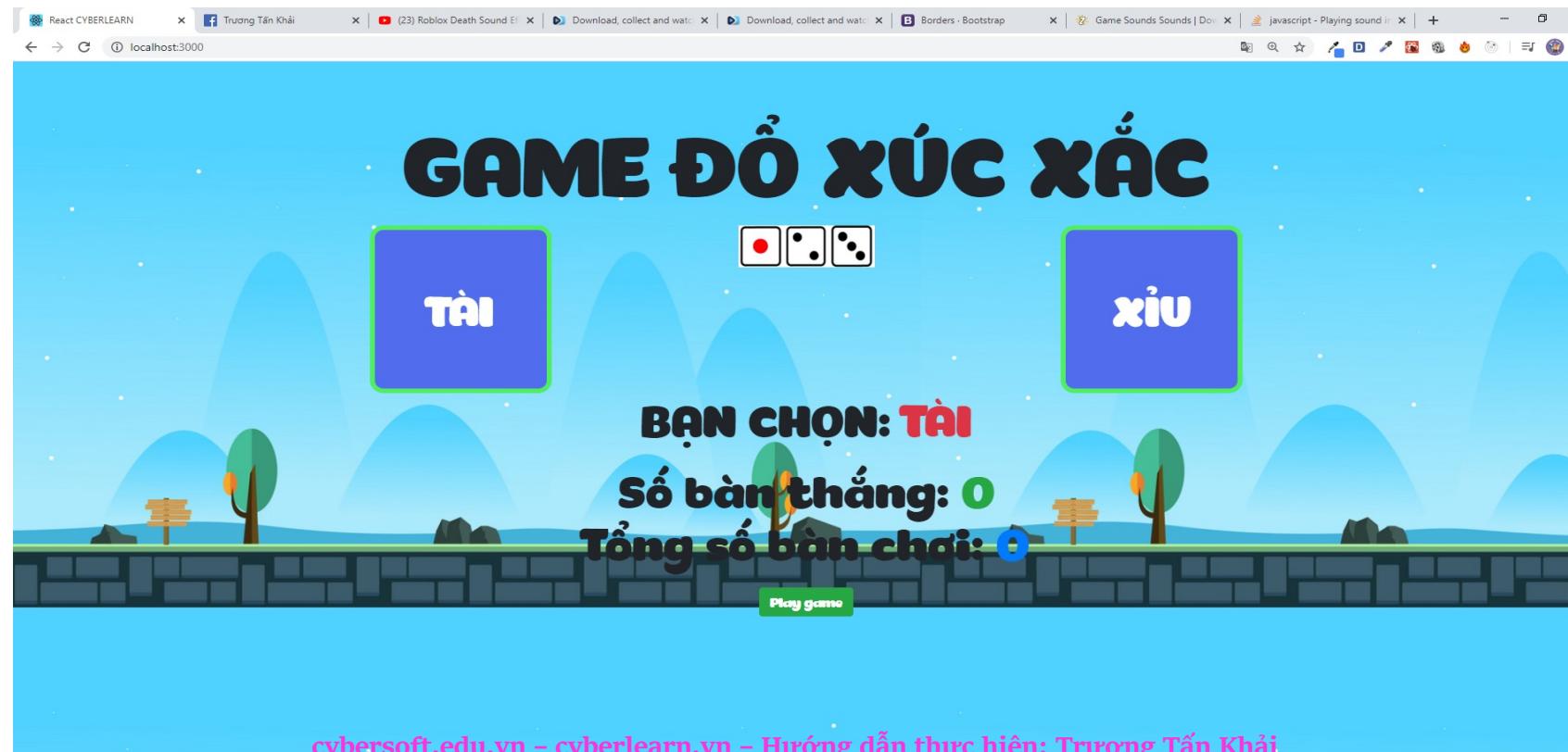
Tại **gioHangReducer** ta nhận được dữ liệu từ nút thêm giỏ hàng sau phương thức dispatch đưa lên store qua biến **action**. Ta dùng thuộc tính type để phân biệt xử lý và dùng thuộc tính giá trị(ở đây là thuộc tính **sanPhamGioHang**) để thay đổi state của Reducer gioHang (state.gioHang). Những nơi mapStateToProps từ gioHangReducer khi thay đổi sẽ tự động cập nhật lại.

```
2 const stateGioHang = {  
3     gioHang: [  
4         { maSP: 1, hinhAnh: './img/vsphone.jpg', tenSP: 'VinSmart Live', soLuong: 1, giaBan: 5700000 }  
5     ] //Giá trị mặc định ban đầu của giỏ hàng  
6 }  
7  
8 export const GioHangReducer = (state = stateGioHang, action) => {  
9     // action gửi lên 2 thuộc tính là  
10    // type, và dữ liệu => dựa vào type  
11    // để xác định xử lý tương đương  
12    // cho action.  
13    // type: là thuộc tính bắt buộc  
14  
15    switch (action.type) {  
16        case 'THEM GIO HANG': {  
17            let gioHang = [...state.gioHang];  
18            const index = gioHang.findIndex(spGH => spGH.maSP === action.sanPhamGioHang.maSP);  
19            if (index === -1) {  
20                //Kiểm tra nếu sản phẩm chưa có trong giỏ hàng thì thêm sản phẩm đó vào giỏ hàng  
21                gioHang.push(action.sanPhamGioHang)  
22            } else {  
23                //Ngược lại nếu đã có trong giỏ hàng rồi thì tăng số lượng  
24                gioHang[index].soLuong += 1;  
25            }  
26            //Cập nhật lại state  
27            state.gioHang = gioHang;  
28            return { ...state }  
29        }  
30    }  
31  
32    return { ...state }  
33 }
```

# Bài tập game xúc xắc

## Yêu cầu:

- + Phân tích giao diện (chia component hợp lý, có thể chia thành nhiều cách khác nhau).
- + Xác định state (nguồn dữ liệu thay đổi trên giao diện).
- + Tổ chức lưu trữ redux.
- + Xây dựng chức năng đặt cược và play game của ứng dụng.
- + Link tài nguyên: [https://drive.google.com/drive/folders/1cFlPw51YzVfUa-c4FDQoEz6QL4V3O\\_Xx?usp=sharing](https://drive.google.com/drive/folders/1cFlPw51YzVfUa-c4FDQoEz6QL4V3O_Xx?usp=sharing)

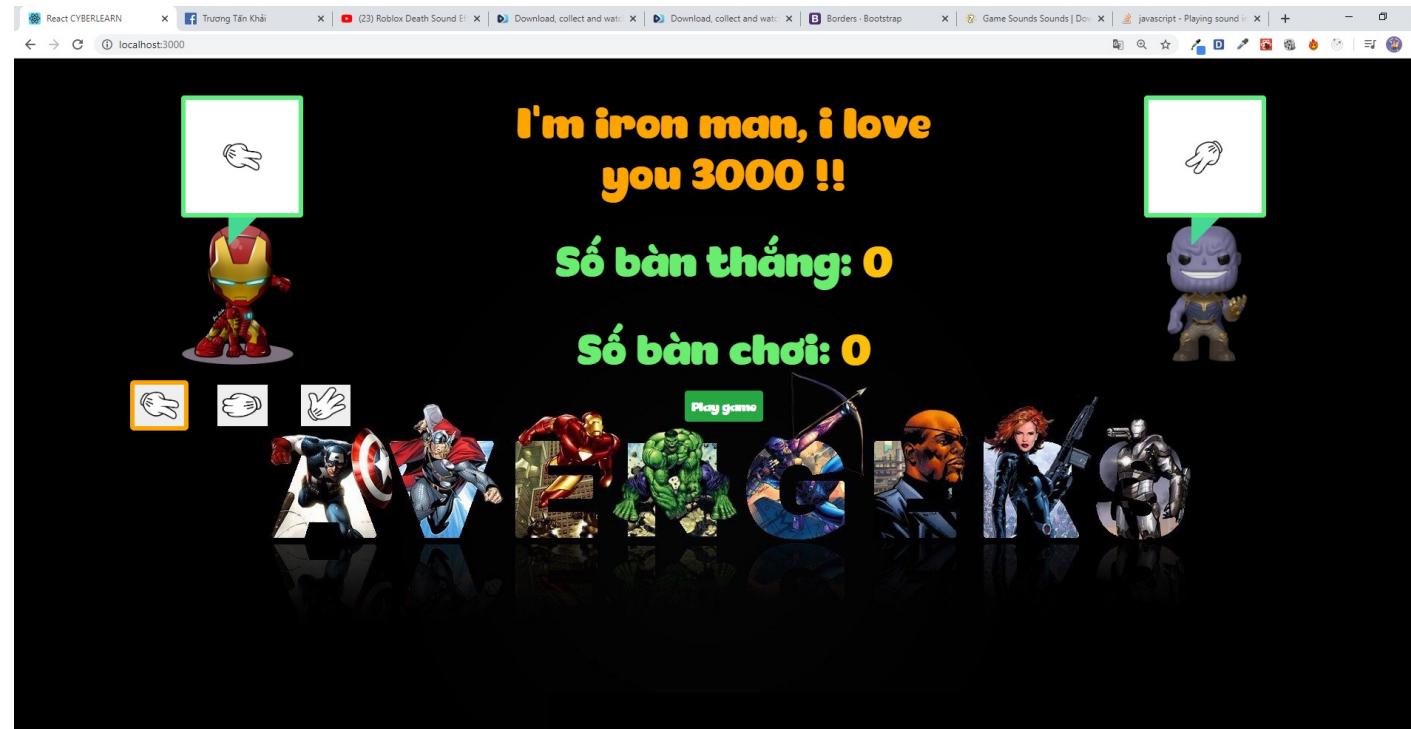


# Bài tập game oẵn tù tì

## Yêu cầu:

- + Phân tích giao diện (chia component hợp lý, có thể chia thành nhiều cách khác nhau).
- + Xác định state (nguồn dữ liệu thay đổi trên giao diện).
- + Tổ chức lưu trữ redux.
- + Xây dựng chức năng đặt cược và play game của ứng dụng.

Link tài nguyên: [https://drive.google.com/drive/folders/1RWZN4o2NFiB1xMIdPbS\\_j-f\\_hsQX7\\_Oq?usp=sharing](https://drive.google.com/drive/folders/1RWZN4o2NFiB1xMIdPbS_j-f_hsQX7_Oq?usp=sharing)



# Cấu trúc xây dựng đầy đủ của redux (action creator)

- ❑ Redux bao gồm 3 thành phần chính
  - ❑ Action:Là nơi mang các thông tin dùng để gửi từ ứng dụng đến Store. Các thông tin này là 1 object mô tả những gì đã xảy ra . Nói dễ hiểu, từ 1 component, ta muốn thay đổi state trên store, ta phải gửi action , là một object để miêu tả muốn làm gì.
  - ❑ Reducer: nơi tiếp nhận action và thay đổi state.Gồm 2 loại:
    - ❑ Root Reducer: là Boss, quản lý tất cả reducer con
    - ❑ Child Reducer: như đã biết về state, state là một object có nhiều thuộc tính, mỗi child reducer chịu trách nhiệm thay đổi 1 thuộc tính trong state.
  - ❑ Store:Nơi lưu trữ và quản lý state (Chính là Big Boss)

# Hướng dẫn thực hiện bài tập đặt vé xem phim – sử dụng action creator

- + Xây dựng giao diện tương tự (Không cần phải css giống quá)

Link giao diện: [https://demo.w3layouts.com/demos\\_new/template\\_demo/23-03-2018/movie\\_seat\\_selection-demo\\_Free/759053290/web/index.html](https://demo.w3layouts.com/demos_new/template_demo/23-03-2018/movie_seat_selection-demo_Free/759053290/web/index.html)

- + Xác định các trạng thái thay đổi của giao diện tổ chức lưu trữ redux (Sử dụng action creator)

- + Xây dựng chức năng cho ứng dụng

+ Link tài nguyên: <https://drive.google.com/drive/folders/1s7cbWmpNBciuQdwkwYAkMsO9ODPP2Ckk?usp=sharing>

- + Giao diện tham khảo

Fill The Required Details Below And Select Your Seats

Name: khai Number of Seats: 2

Start Selecting

Please Select your Seats NOW!

Selected Seat    Reserved Seat    Empty Seat

A	1	2	3	4	5	6	7	8	9	10	11	12
B												
C												
D												
E												
F												
G												
H												
I												
J												

SCREEN THIS WAY

Confirm Selection

Name	Number of Seats	Seats
khai	2	A1,A2

ĐẶT VÉ XEM PHIM CYBERLEARN.VN

Màn Hình

DANH SÁCH GHẾ BẠN CHỌN

Số ghế	Giá	Ngày
A1	150.000	X
A2	150.000	X
A3	150.000	X
Tổng tiền	450.000	