1. **Định nghĩa Component**

Cách 1: định nghĩa 1 function bình thường có tên

Function Champ() {

Return ( .....)

}

Note: ko thể return 2 div cùng 1 cấp >>> chỉ 1 div chứa các tag con – only 1 wrapper.

cách 2: anonymous function

var Champ = function(){

return ( ............ )

}

Cách 3: arrow function(ko cần keyword: return)

() => (.............);

Cách 4: Class

Class Champ extends Component {

Render(){

Return ( ......... );

}

}

1. **Props(Properties)**

**Props:** thuộc tính của Component. Là cách truyền input vào Component.

1. Function Champ(props){

Return (

<div className=”test”>{props.*ten*}</div>

);

}

2. Class

class Champ extends Component {

render() {

return ({this.props.*ten*});

};

}

Note: cách tạo class này thì ko cần truyền ghi input props. Mà mặc định đã có sẵn. Nhớ có keyword: this

1. **JSX**

Cho phép viết biến với {} (code, props của js)

Dùng công cụ: html to JSX

1. **COMPONENT HÓA**

Cắt từng phần của file html sang các component.

1. **SỬ DỤNG PUBLIC\_URL**

SET-UP thư viện, file css, bootstrap, js bằng cách thêm trước các link tiền tố **%PUBLIC\_URL%.** Đây là 1 biến dc định nghĩa sẵn trong ReactJs.

**Href=”%PUBLIC\_URL%/**index.css”

1. **CÁCH PHÍM TẮT TẠO NHANH**

Install React code snippet extension

Rcc: react component

Sau khi set-up component: **export component** trong file component lun

**Import** *tênClass đó* vào file App.js (import *tênClass* from ‘./components/tenfile’)

1. **ĐƯA CÁC COMPONENTS VÀO FILE APP.JS**

**Copy** code html và chuyển sang JSX bằng ‘convert html to JSX’.

Then, đưa vào file component.

Để có thể truy cập các props: khởi tạo constructor(props){super(props)};

Shortkey: ‘con’ > tạo constructor(props){super(props)};

1. **VIẾT ,GỌI FUNCTION CÓ/KHÔNG THAM SỐ TRONG REACTJS**

Ko có tham số: onClick={this.*tênfunc*}; Note: ko có ()

Có thêm tham số:

1. Dùng arrow function trong onCLick luôn. onClick={() => this.*tênfunc*(param)}; (có thể dùng cho hàm ko tham số luôn nhớ có (), phải có this )
2. Dùng bind(): onClick={this.*func.*bind(this, ‘string’);}
3. **STATE TRONG REACTJS**

Là nơi lưu các biến, tham số trong react.

Là từ khóa > có sẵn trong react và ko cần khai báo lại.

Là 1 Obj > lưu ko giới hạn

Cú pháp ghi trong constructor(props){

Super(props);

This.state = { //shortkey: stt

Key: value

}

}

* Để thay đổi state: this.setState({key: value}); // shortkey: sst.

Trong reactJS: khi thay đổi các state bằng setState thì nếu thỏa điều kiện, các hàm có liên quan đến state sẽ tự động gọi. 🡺 khi đó ta ko cần gọi lại hàm nữa. (đây gọi là biến trung gian).

1. **DEFAULTVALUE**

Có thể dùng trong thẻ component: <Header defaultValue={this.props.title} ........ />

1. **Ref**

Đây là biến trung gian 🡺 để thay đổi thông tin mà ko cần load lại trang.

Ko cần render lại thông số khi **sửa sẽ lưu vào biến ref**. Dùng để lấy value trong input

Khác với State: state là biến luôn có. Dùng để phân bổ thông tin cho các hàm.

1. **LifeCycle**

Vòng đời của component.

lifeCycle là tập hợp các hàm dựng sẵn(build-in) trong reactJS.

Có thể can thiệp vào trước, sau khi render()

Gồm 4 quá trình:

1. init – khởi tạo
2. mounting – gắn những hàm kết nối component.
3. updation – props, states.
4. Dismount – hủy gắn kết

* Note: đây là những hàm build-in 🡺 keyword phải viết đúng và có viết hoa.

Shortkey: cwm > **componentWillMount()**

**Cdm > componentDidMount()**

**Cwr >componentWillReceiveProps()**

**Scup > shouldComponentUpdate()**

**Cwup >** componentWillUpdate()

**Cdup >** componentDidUpdate()

**Cwun > componentWillUnmount()**

1. **Các hàm build-in của lifecycle**

**componentWillMount**(): hàm này sẽ chạy **trước** hàm render();

**componentDidMount():** hàm này sẽ chạy **sau** hàm render();

* **Updatation**:

State: **shouldComponentUpdate**(nextProps, nextState){} \\ hàm này gọi khi ta gọi đến 1 method làm thay đổi state, hay props.

Khi hàm **shouldComponentUpdate** return **true** thì sẽ chạy tiếp 3 hàm : componentWillUpdate(), render(), componentDidUpdate(). **False** thì sẽ ko chạy 3 hàm này.

Props: **componentWillReceiveProps():** hàm này gọi khi prop của component bị thay đổi.(thường là 1 event trong parent Component – App.js)

Còn lại 3 hàm kia giông như State.

1. **REACT ROUTER**

Khi click vào menu thì ko đổi trang, chuyển hướng web mà ko load lại.

1. Tạo giao diện
2. Điều hướng SPA
3. Truyền tham số
4. Dựng Slug cho url = router
5. redirect
6. **CÁCH BƯỚC CHUYỂN HTML SANG COMPONENTS**
7. Copy mã HTML vào các components. (note: dùng công cụ Html to JSX)
8. Import vào file App.js(đây coi như file root của Reactjs)
9. Đặt tenclass vào thẻ div của App.js
10. **REACT ROUTER**

**Keyword: react-router-dom**

**Property:** activeClassName

1. **Dùng Array map để đưa dữ liệu ra từ .JSON**

**Tạo** 1 File .JSON gồm 1 mảng chứa các Objects(id,name,img,excerpt,...)

**Dùng Array map** để render ra giao diện

Render(){

Data.**map((value, index)** =>

Return(

<Com

key={index}

name={value.name}

img={value.img}

excerpt={value.excerpt}

></Com>

)

)

}

1. **Đường dẫn thân thiện với SEO**

**Keyword:**  slug javascript.