**9. children props và render props**

**10. NodeJs là gì.**

- *NodeJs* => *JavaScript Runtim*e, Khi có NodeJs => tạo ra môi trường để thực thi code Js mà không cần dùng các môi trường khác (ví dụ như trình duyệt…).

- Để NodeJs có thể thực thi code js thì nó có => *Chrome’s V8 JavaScript engine*(dev by google)

- Vai trò của NodeJs trong FrontEnd:

+ Tạo ra một máy chủ web trên máy tính

+ Thư viện npm: sử dụng những thư viện cần thiết trên thư viện npm của NodeJs.

- Các thư viện được cài bởi npm được lưu trong thư mục node\_modules

**11. Tạo dự án React + Webpack**

- Cấu trúc dự án

//Cấu trúc dự án

React-Webpack

     - src // thư mục chứa source code chính

            -components // thư mục chứa các components

                -component.js // các components

                -component.js // các components

                -component.js // các components

            -index.js // file khởi tạo dự án. file đầu vào của dự án, render app vào #root

     - public

            - index.html // html page, nơi chứa #root element

Khởi tạo dự án:

1. npm init -> tạo một file package.json -> chứa thông tin mô tả của dự án, quản lý những thư viện được cài đặt trong dự án.

2. cài đặt thư viện webpack

- ***npm i webpack webpack-cli –save-dev*** (--save-dev để khai báo là thư viện này dùng trong môi trường phát triển -> Devdependencies trong package.json)

- Các thư viện được cài bởi npm được lưu trong thư mục node\_modules

3. cài đặt thư viện React và React-DOM

- *npm i react react-dom (có version thì thêm vào @<<version>> ví dụ* [*react@17.0.2*](mailto:react@17.0.2)*)*

4. cài đặt thư viện Babel

- npm i @babel/core babel-loader @babel/preset-env @babel/preset-react –save-dev

- babel/core: dùng để chuyển đổi ES6->ES5

- babel-loader: cho phép chuyển các files JavaScript sử dụng babel và webpack

- babel/preset-env: cài đặt sẵn giúp sử dụng JavaScript mới nhất trên nhiều môi trường khác nhau (nhiều trình duyệt khác nhau). Gói này đơn giản là support chuyển đổi tử các phiên bản JS cao (ES6,7,8…) về ES5

- babel/preset-react: giúp chuyển đổi Jsx thành JavaScript

5. tạo file index.html và index.js

- tạo file public/index.html trong thư mục gốc.

- tạo file src/index.js trong thư mục gốc

6. Cấu hình webpack

- tải thêm 2 thư viện để webpack hỗ trợ cho file css.

- npm i css-loader style-loader

- Tạo file webpack.config.js

const path = require('path') // path là thư viện của nodejs dùng để định tuyến được những vị trí một folder hay một file đang ở đâu trong dự án.

                             // dùng để cấu hình đường dẫn file bundle (là file cuối khi webpack đóng gói code.)

module.exports = {

    entry: "./src/index.js", // file đầu vào của dự án. là file mà từ nó có thể đi đến các file resouce mà ta đã code.

    output: {

        path: path.join(\_\_dirname, "/build"), //\_\_dirname dùng để lấy thư mục project // dòng này để cấu hình dường dẫn thư mục chứa file bundle

        filename: "bundle.js" // tên file bundle;

    },

    module:{ // nơi cấu hình những module js, css, img...

        rules:[ // các là mảng chứa những object để cấu hình cho những module riêng biệt.

            {

                test: /\.js$/, // cấu hình các file module theo extension

                exclude: /node\_modules/,

                use: ["babel-loader"]

            },

            {

                test: /\.css$/,

                use: ["css-loader","style-loader"]

            }

        ]

    },

    plugins:{

    }

}

7. Cấu hình babel.

**-** tạo file .babelrc

{

    "presets": [

        "@babel/preset-env",

        "@babel/preset-react"

    ]

}

8. thêm cấu hình để build và khởi chạy dự án

**Trong file package.json**

**ở key scripts cấu hình thêm start và build**

"scripts": {

    …,

    "start": "webpack --mode development --watch",

    "build": "webpack --mode production"

  },

**Khi run project ta không run bằng live server như trước mà sẽ dùng lệnh:**

**+ Npm run build run build là build dự án sang các file bundle để chạy trên môi trường production, + npm start chạy dự án ở môi trường dev**

**“start”: khi gõ npm start thì node sẽ tìm và thực thi câu lệnh được config:** webpack --mode development –watch (có thể cấu hình câu lệnh khác)**,** --watch để webpack sẽ luôn lắng nghe thay đổi, khi file thay đổi webpack sẽ thực hiện biên dịch nhé.

**“build”: khi gõ npm build thì node sẽ tìm và thực thi câu lệnh được config:** webpack --mode production

9. tải html-webpack-plugin để cấu hình webpack tự link script vào index.html.

**- npm install html-webpack-plugin –save-dev**

**- Sau đó cấu hình trong file webpack.config.js:**

const HtmlWebpackPlugin = require('html-webpack-plugin'); // import thư viện html webpack plugin

**- config plugins trong webpack.config.js**

plugins:[

        new HtmlWebpackPlugin({

            template: './public/index.html' // file html muốn webpack nhúng file script bundle

        }),

    ]

10. Cài đặt webpack-dev-server

- thay thế cho live server để chạy ứng dụng.

- npm i webpack-dev-server.

- cấu hình thêm ở key script trong file package.json

 "scripts": {

    "test": "echo \"Error: no test specified\" && exit 1",

    "start": "webpack-dev-server --mode development --open --hot ",

    "build": "webpack --mode production"

  },

Thay thế webpack bằng webpack-dev-server: sẽ vừa build vừa đảm nhận việc bật một server để thực thi code.

--open là cấu hình mở ứng dụng trên trình duyệt luôn

--hot (gọi là hot reload): khi sửa code thì nó sẽ lắng nghe và update đúng vào chỗ cần sửa thay vì refresh nguyên trang

**12. ReactCreateApp**

- Tạo dự án React với các thư viện cần thiết như React, React-DOM, Babel, webpack…

- lệnh:

Npx i create-react-app

* Npx là thư viện được cài chung với nodejs, giúp chúng ta chạy những thư viện của npm mà không cần tải về. khi chạy nó sẽ kiểm tra thư viện đã được cài đặt trên máy chưa nếu chưa thì nó sẽ tải về và chạy.
* Create-react-app là thư viện facebook

**13. NPM, NPX**

NPM

-Project: có phạm vi trong dự án.

- npm i react react-dom

- npm install react react-dom --save => dependencies

- npm i react react-dom -D

- npm install react react-dom --save-dev => devDependencies

- Global : có phạm vi ở toàn cục => cài dự án vào máy tính (lưu trữ ở User trong máy tính)

- npm install --global create-react-app

- npm i -g create-react-app

- nếu là các máy như mac, linux ubuntu thì thêm lệnh sudo để lấy quyền root

- sudo npm i -g create-react-app

- npm uninstall -g create-react-app

NPX

- để hỗ trợ thực thi những thư viện có folder bin để thực thi thư viện

**14. CRA folder structure.**

- thư mục public là thư mục root của dự án. Chứa những file tĩnh được public ra ngoài.

- trong thư mục src

- reportWebVitals.js là file thống kê báo cáo, cung cấp file này cho công cụ tìm kiếm trong tương lai (Google analytics) để giúp thống kê hiệu năng, thời gian tải bao nhiêu, thời gian render giao diện là bao nhiêu. để chúng ta tối ưu trải nghiệm người dùng.

**15. Hook là gì ?**

- giúp function component có thể làm được một ứng dụng hoàn chỉnh như class component chứ không chỉ render ra giao diện người dùng.

- hook là những function được viết sẵn bởi ReactJS. Và chỉ dùng cho **function component**

- mỗi Hook (mỗi hàm) có những tính năng và có trường hợp cụ thể để sử dụng nó.

- khi làm việc với function component ta cần những tính năng của hook thì ta sẽ lấy ra những hook đó và thêm nó vào function component để sử dụng

- để sử dụng hook ta cần file import nó từ module react

- quy ước đặt tên của hook là có chữ use ở đầu.

**15.1 Hook: UseState()**

- trạng thái của dữ liệu

- ví dụ nếu trang web đang hiển thị là nguyễn văn A sau đó dữ liệu thay đổi thành nguyễn văn B thì nếu code thuần thì khi code thuần chúng ta phải dùng DOM -> getElement -> thay đổi dữ liệu -> DOM render ra giao diện.

- khi sử dụng UseState thì chúng ta có thể dùng để áp dụng vào trường hợp khi dữ liệu thay đổi thì giao diện tự động được cập nhật. (render lại giao diện theo dữ liệu).

- để sử dụng ta phải import hook vào:

- đối số của useState là một initState, là giá trị khởi tạo có nghĩa là chỉ dùng 1 lần, khởi tạo lần đầu.

- return ra một mảng có 2 phần tử. 1 là state, 2 là hàm setState

- trong lần đầu tiên khởi tạo thì useState sẽ lấy giá trị khởi tạo gắn cho state.

- mà muốn set lại state thì ta dùng hàm setState

import { useState } from 'react';

function App() {

  const [counter, setCounter] = useState(0);

  const handleIncrease = ()=>{

     setCounter(counter + 1);

  }

  return (

    <div className="App">

        <h1>counter: {counter}</h1>

        <button onClick={handleIncrease}>counter++</button>

    </div>

  );

}

export default App;

Lưu ý \*:

* Mỗi lần setState lại thì react sẽ thay đổi state bằng giá trị mới. chứ không phải update vào state cũ.  
  ví dụ state là một object user khi setState là truyền hẳn một object user mới có thông tin mới, có thể dùng toán tử spread để giải những thông tin cũ và truyền những thông tin cũ vào.
* Để useState có thể render lại giao diện thì mỗi khi chúng ta setState thì react sẽ tự động thực thi lại function component đó. Nhưng nếu chúng ta gọi lại setState nhiều lần trong cùng một đoạn code React sẽ chờ các lệnh setState chạy hết rồi mới render lại nên chỉ render lại 1 lần. nên state có thể bị giữ nguyên giá trị lần đầu và giao diện chỉ được render lại 1 lần thôi  
    
  ví dụ:
* const handleIncrease = ()=>{
* // ví dụ state counter = 0
* setCounter(counter + 1); //state = counter + 1 // counter vẫn bằng 0
* setCounter(counter + 1); //state = counter + 1 // counter vẫn bằng 0
* setCounter(counter + 1); //state = counter + 1 // counter vẫn bằng 0
* //setState 3 lần đều nhận giá trị là 1 sau đó re-render giao diện counter = 1
* }
* Vậy muốn dùng setState 3 lần như trên thì ta có thể dùng callback
* const handleIncrease = ()=>{
* setCounter(afterCounter => afterCounter + 1);
* setCounter(afterCounter => afterCounter + 1);
* setCounter(afterCounter => afterCounter + 1);
* }

Tham số của useState có thể nhận một hàm callback:

Trường hợp dùng để tăng performance và tránh việc useState thực thi lại hàm thì sẽ thực thi lại những đoạn mã logic dùng để khỏi tạo initState;

Trong đoạn code dưới, tạo ra một hàm logic dùng để tính initState là tổng price của mảng cars.

Sau đó truyền vào useState, khi chạy thì hàm này chỉ chạy một lần những lần sau useState sẽ không gọi nó nữa.

function createInitState(){

    const cars = [

      {

          brand: "honda",

          price: 1000

      },

      {

        brand: "honda",

        price: 1000

    }

    ]

    const total = cars.reduce((total,c)=> { console.log(total);console.log(c); return  total + c.price},0);

    return total;

  }

  const [counter, setCounter] = useState(createInitState);

**Lưu ý: two-way binding**

* **Là ràng buộc dữ liệu 2 chiều:**

**+ có nghĩa là khi thay đổi dữ liệu trên ui thì dữ liệu ở trong component cũng thay đổi và ngược lại thay đổi dữ liệu ở trong component thì ui cũng được render lại để hiện thị dữ liệu mới nhất.**

* **ràng buộc 1 chiều có nghĩa là chỉ thay đổi một chiều. thay đổi ở ui có thay đổi ở trong component nhưng trường hợp ngược lại dữ liệu trong component thay đổi lại không render lại để hiển thị ra ui dữ liệu mới.**
* **hoặc là dữ liệu trong component thay đổi render lại để show ra ui nhưng khi thay đổi trên giao diện lại không binding lại dữ liệu trong component**
* **trong project bài số 15.1 có component car là ví dụ thể hiện two way binding**

**15.2 useEffect**

**1. useEffect(callback)**

**- gọi callBack mỗi khi component re-render.**

**- gọi callback sau khi component thêm element vào DOM. Có nghĩa là sau khi render ra giao diện mới gọi callback.**

**2. useEffect(callback,[])**

**- gọi callback 1 lần sau khi component mouted**

**3. useEffect(callback,[dependencies]);**

**- gọi lại callback mỗi khi dependencies thay đổi**

**Note cho cả 3 trường hợp:**

**- Trong cả 3 trường hợp thì callback luôn được gọi sau khi component mounted**