React và Javascript ES6

*Ôn tập các kiến thức Javascript -ES6 dùng với React.JS*

# #1 ES6 Variables

Before ES6 there was only one way of defining your variables: with the var keyword. If you did not define them, they would be assigned to the global object.

*Trước version ES6, chỉ có 1 cách duy nhất để định nghĩa biến, đấy là sử dụng từ var. Nếu không sử dụng var để khai báo biến, biến đấy sẽ trở thành biến global*

Now, with ES6, there are three ways of defining your variables: **var, let**, and **const**.

*Từ version ES6 trở đi, khai báo biến có thể bắt đầu bằng:* ***var, let*** và ***const***

### var

Ví dụ: var name = ‘Anne’;

* If you use var outside of a function, it belongs to the global scope.
* If you use var inside of a function, it belongs to that function.
* If you use var inside of a block, i.e. a for loop, the variable is still available outside of that block.
* var has a function scope, not a block scope.

let

Ví dụ: **let x = 10;**

* let is the block scoped version of var, and is limited to the block (or expression) where it is defined.
* If you use let inside of a block, i.e. a for loop, the variable is only available inside of that loop.
* let has a block scope.

const

ví dụ: const y = ‘abc;

* const is a variable that once it has been created, its value can never change.
* const has a block scope.
* constant cannot change through re-assignment
* constant cannot be re-declared

Why can I change a constant object (array) in JavaScript ??? <https://stackoverflow.com/a/23436563><https://www.javascripttutorial.net/es6/javascript-const/>

When you're adding to an array or object you're not re-assigning or re-declaring the constant, it's already declared and assigned, you're just adding to the "list" that the constant points to.

*Khi bạn thêm vào một mảng hoặc đối tượng, bạn không gán lại hoặc khai báo lại hằng số, vì hằng số đó đã được khai báo và gán rồi, bạn chỉ cần thêm vào "danh sách" mà hằng số đó trỏ tới.*

**this works fine:**

**const x = { };**

**x.foo = 'bar';**

**console.log(x); // {foo : 'bar'}**

**x.foo = 'bar2';**

**console.log(x); // {foo : 'bar2'}**

**const y = [ ];**

**y.push('foo');**

**console.log(y); // ['foo']**

**y.unshift("foo2");**

**console.log(y); // ['foo2', 'foo']**

**y.pop();**

**console.log(y); // ['foo2'] but neither of these:**

**const x = {};**

**x = {foo: 'bar'}; // error - re-assigning**

**const y = ['foo'];**

**const y = ['bar']; // error - re-declaring**

**const foo = 'bar'; foo = 'bar2'; // error - can not re-assign var foo = 'bar3'; // error - already declared function foo() {}; // error - already declared**

# #2.ES6 Classes

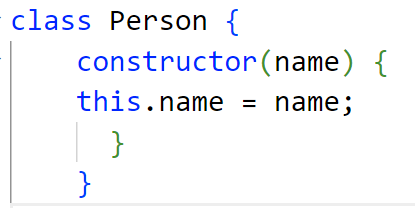
### **Classes**

ES6 introduced classes. (Class chỉ được sử dụng từ version 6 của javascript)

A class is a type of function, but instead of using the keyword function to initiate it, We use the keyword class, and the properties are assigned inside a constructor() method.

*Class là một loại hàm đặc biệt, thay vì sử dụng từ “function” để khởi tạo, chúng ta sử dụng từ “class”, và những thuộc tính của class được gán bên trong phương thức của hàm tạo - constructor*

Example: A simple class constructor:



Notice the case of the class name. We have begun the name, "Person", with an uppercase character. This is a standard naming convention for classes.

(Tên của class bắt buộc phải bắt đầu bằng ký tự in hoa)

Ex: Create an **object** called "**myInformation**" based on the Person class:

**class Person** {

**constructor**(name) {

this.name = name;

}

}

const **myInformation** = new **Person**("ABCxyz");

The constructor function is called automatically when the object is initialized. (Hàm tạo constructor sẽ được gọi tự động khi object được khởi tạo)

### **Method in Classes**

* You can add your own methods in a class:
* Create a method named "getAddress":

class Person {

    constructor(name, address) {

        this.name = name;

        this.address = address;

    }

    getAddress() {

        return 'I live in ' + this.address;

    }

}

const myInformation = new Person("ABC", "Ha Noi");

console.log(myInformation);

console.log(myInformation.getAddress());

### **Class Inheritance**

* To create a class inheritance, use the **extends** keyword.
* A class created with a class inheritance inherits all the methods from another class. *(Để sử dụng tính năng kế thừa class, sử dụng từ extends, khi đó nó sẽ quyền sử dụng tất cả method của class kế thừa)*

Ví dụ:

class Animal {

    constructor() {

    //todo

   }

   doAction() {

     return 'Go Go away';

   }

 }

class Dog extends Animal {

    constructor(model) {

**super();**

        this.model = model;

   }

 }

 const myDog = new Dog("BullDogs");

 myDog.doAction();

By calling the **super()** method in the constructor method, we call the parent's constructor method and get access to the parent's properties and methods.

### **Exercise**

Tài liệu tham khảo: <https://www.w3schools.com/react/react_es6_classes.asp>

Declare an object of the **Novel** class, then get it’s author

Novel

Variable Value

**Title** "Tôi thấy hoa vàng trên cỏ xanh

**Author** Nguyễn Ngọc Ánh

Class Novel { constructor(...){ ……

}

getAuthor(){....}

}

Let myNovel = new Novel(...)

console.log(myNovel….)

# #3.Object javascript (array)

1. **object.property and object["property"]**

### Javascript Object ??? Object trong thế giới Javascript

( run a file with node.js: node file\_name)

Ví dụ: let **person** = "An";

=> This code assigns a simple value (An) to a variable named **person**

* **Objects** are variables too. But objects can contain many values.

*Object cũng là 1 biến,chỉ có điều, object có thể chứa nhiều loại dữ liệu cùng lúc*

Ví dụ: assigns many values (An, 26) to a variable named person:

let person = { name: "An", age: 26 };

The values are written as **name: value** pairs (name and value separated by a colon)

*Các giá trị của object được viết dưới dạng các cặp tên:giá trị , và được ngăn cách bởi dấu phẩy*

1. **Object Definition - Định nghĩa object**

**{ name1 : value1 , name2 : value2…}**

Ví dụ:

**const person** = {firstName:"An", lastName:"Nguyen", age:26, eyeColor:"black"};

Spaces and line breaks are not important. An object definition can span multiple lines (*không quan trọng dấu cách và dấu xuống dòng khi định nghĩa object*)

1. **Object Properties - Thuộc tính của object**

The **name:values** pairs in JavaScript objects are called properties:

|  |  |
| --- | --- |
| Property(thuộc tính) | Property Value (giá trị) |
| firstName | An |
| lastName | Nguyen |
| age | 26 |
| eyeColor | black |

1. **Accessing Object Properties (Truy cập thuộc tính của object)**

Access object properties in two ways: (có 2 cách để lấy thuộc tính của object) **objectName.propertyName hoặc objectName["propertyName"]**

Arrays are a special type of object. The typeof operator in JavaScript returns "object" for arrays.

But, JavaScript arrays are best described as arrays.

Ví dụ

const person = ["An", " Nguyen ", 26];

1. **5.Exercise**

Tài liệu tham khảo: <https://www.w3schools.com/js/js_objects.asp>

Định nghĩa object với tên là React, với các thuộc tính là language, author, tương ứng giá trị “javascript”, “facebook”

Let React = …

Alert "React Tutorial" by extracting information from the tutorial object.

*(Hiển thị thông báo “React Tutorial” bằng cách lấy thông tin ấy từ object có tên là tutorial*”

let tutorial = { name: "React Tutorial", author: " Mohal vs An", language: "javascript" };

|  |
| --- |
|  |

alert();

# #4 ES6 Arrow Functions

1. **Arrow Functions**

* Arrow functions allow us to write shorter function syntax
* *Arrow functions cho phép chúng ta viết function một cách ngắn gọn hơn*

**Before:**

function hello() {

    return "Hello World!";

}

Hoặc

const hello = function() {return "Hello World!";}

**With Arrow Function: ( )=>{}**

hello = () => {return "Hello World!";}

It gets shorter!

*If the function has only one statement, and the statement returns a value, you can remove the brackets and the return keyword*:

hello = () => "Hello World!";

*(This works only if the function has only one statement)*

**Arrow Function With Parameters**

const hello = (val) => "Hello " + val;

if you have only one parameter, you can skip the parentheses as well:

*Có 1 tham số, có thể bỏ qua cặp dấu { }*

const hello = val => "Hello " + val;

# #5 ES6 Array Methods - map và filter

**Array Methods**

* There are many **JavaScript array methods**.
* One of the most useful in React is the **.map() array method.**
* The .**map() method** allows you to run a function on each item in the array, returning a new array as the result.
* **In React, map()** can be used to generate lists.

### **map()**

const myArray = ['apple', 'banana', 'orange'];

const myList = myArray.map((item) => <p>{item}</p>)

const numbers = [4, 9, 16, 25];

const newArr = numbers.map(Math.sqrt)

map() creates a new array from calling a function for every array element.

map() calls a function once for each element in an array.

map() does not execute the function for empty elements.

map() does not change the original array.

**Syntax:**

**array.map(function(currentValue, index, arr), thisValue)**

<https://developer.mozilla.org/en>[-](https://developer.mozilla.org/en-US/docs/Web/JavaScript/Reference/Global_Objects/Array/map)

[US/docs/Web/JavaScript/Reference/Global\_Objects/Array/map](https://developer.mozilla.org/en-US/docs/Web/JavaScript/Reference/Global_Objects/Array/map)

### **filter**

Ví dụ: Return an array of all values in ages[] that are 18 or over:

const ages = [32, 33, 16, 40];

const result = ages.**filter**(checkAdult);

function checkAdult(age) {

return age >= 18;

}

* The **filter() method** creates a new array filled with elements that pass a test provided by a function.
* The filter() method does not execute the function for empty elements.
* The filter() method does not change the original array.

**Syntax**

**array.filter(function(currentValue, index, arr), thisValue)**

<https://developer.mozilla.org/en>[-](https://developer.mozilla.org/en-US/docs/Web/JavaScript/Reference/Global_Objects/Array/filter)

[US/docs/Web/JavaScript/Reference/Global\_Objects/Array/filter](https://developer.mozilla.org/en-US/docs/Web/JavaScript/Reference/Global_Objects/Array/filter)

**Exercise**

Complete the array method that will allow you to run a function on each item in the array and return a new array.

const myList = myArray. ((item) => <p>{item}</p>)

—------------

Const input = [1, 2 , 3, 4, 5]

Const output = [10, 20, 30, 40, 50]

const output = input. ();

# #6 Template literals (Template strings) - Dấu nháy chéo backticks

1. **Back-Tics Syntax**

Template Literals use back-ticks (``) rather than the quotes (" ") to define a string: (Sử dụng dấu nháy chéo thay vì nháy đơn/nháy đôi)

Vì dụ: let text = `Hello World!`;

With template literals, you can use both single and double quotes inside a string:

*(với template strings, chúng ta có thể sử dụng nháy đơn và nháy đôi cùng 1 lúc)*

let text = ` He's often called "An" `;

1. **Multi-line strings**

Using normal strings, you would have to use the following syntax in order to get multiline strings:

console.log('string text line 1\n' +

'string text line 2');

// "string text line 1

// string text line 2"

**Using template literals, you can do the same with this:**

console.log(`string text line 1 string text line 2`); // "string text line 1 // string text line 2" let a = 5;

let b = 10;

console.log('Fifteen is' + (a + b) + ' and\nnot' + (2 \* a + b) + '.');

// "Fifteen is 15 and

// not 20."

That can be hard to read – especially when you have multiple expressions.

*(rất khó để đọc code, đặc biệt, khi trong code liên quan tới biến số và phép tính toán)*

let a = 5;

let b = 10;

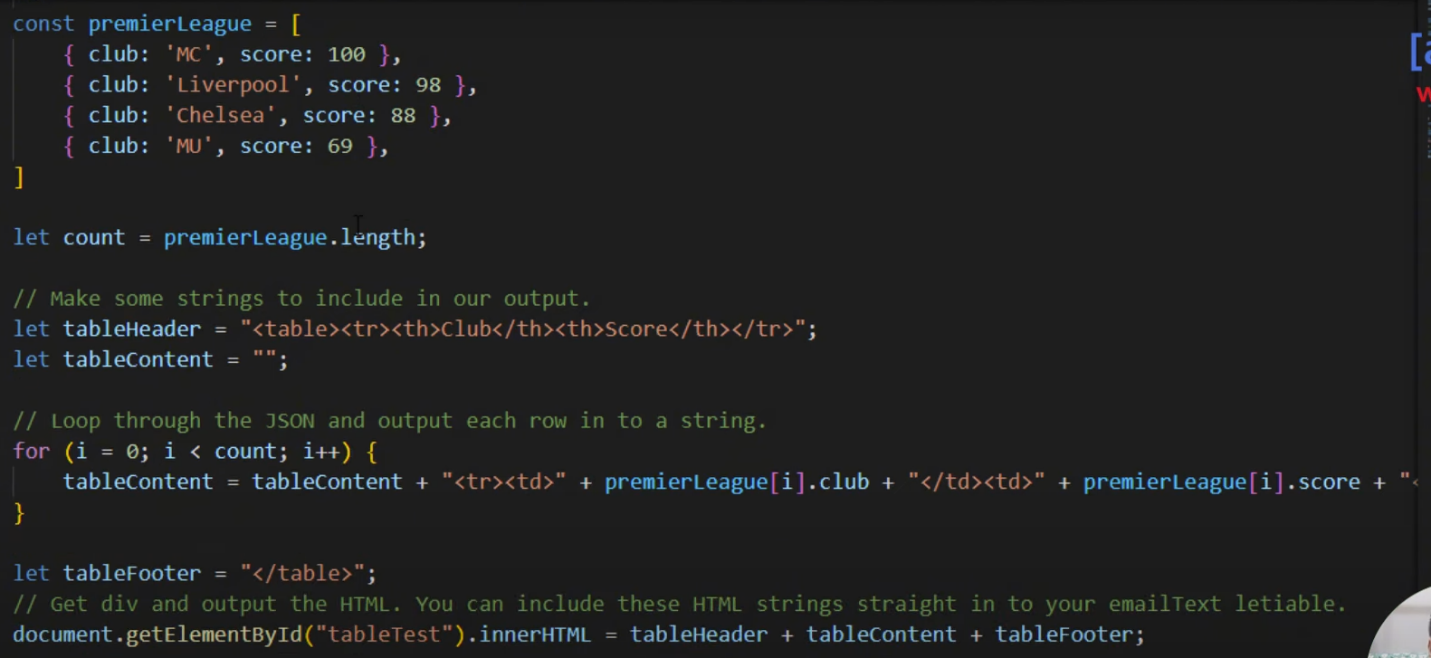
console.log(`Fifteen is **${a + b}** and not ${2 \* a + b}.`);

// "Fifteen is 15 and

// not 20."

=> chỗ nào cần thay biến số/phép tính toán, thì dùng ${ viết code trong này }

Ví dụ:



1. **Exercise**

Tài liệu tham khảo: [https://developer.mozilla.org/enUS/docs/Web/JavaScript/Reference/Template\_literals](https://developer.mozilla.org/en-US/docs/Web/JavaScript/Reference/Template_literals)

Input:

const base\_url = “localhost:8080”; cost api = “get-user”; fetch\_page = 2;

Output: // localhost:8080/get-user?page=2

Hoàn thiện example đầu bài với template strings

# #7.Spread syntax (...) - Cú pháp toán tử mở rộng

### **Spread Operator**

The JavaScript spread operator (...) allows us to quickly copy all or part of an existing array or object into another array or object.

*Toán tử 3 dấu chấm cho phép chúng ta copy tất cả (hoặc một phần) của một array/object sang một array/object khác*

const numbersOne = [1, 2, 3];

const numbersTwo = [4, 5, 6];

const numbersCombined = [...numbersOne, ...numbersTwo]; // [1,2,3,4,5,6]

const numbersCombined = [...numbersTwo, …numbersOne]; ???

### **Push a new item to array**

Let myArr = [“An”, “IUH”, “React”];

Thêm phần tử vào cuối mảng : **array.push()** or ???

Thêm phần tử vào đầu mảng : **array.unshift()** or ???

### **Spread operator with objects (sử dụng với object)**

const myVehicle = {brand: 'Ford', model: 'Mustang', color: 'red' }

const updateMyVehicle = {type: 'car', year: 2021, color: 'yellow'}

const myUpdatedVehicle = {...myVehicle, ...updateMyVehicle}

//{ brand: ‘Ford’, model: ‘Mustang’, color: ‘red’, type: ‘car’, year: 2021, color: ‘yellow’ }

// { brand: ‘Ford’, model: ‘Mustang’, color: ‘red’, type: ‘car’, year: 2021 }

// { brand: ‘Ford’, model: ‘Mustang’,, type: ‘car’, year: 2021, color: ‘yellow’’ }

Notice the properties that did not match were combined, but the property that did match, was overwritten by the last object that was passed

*Lưu ý các thuộc tính không khớp đã được kết hợp, nhưng thuộc tính khớp đã được ghi đè bởi đối tượng cuối cùng được truyền.*

X <= Y

let objClone = { ...obj }; // pass all key:value pairs from an object

### **Exercise**

Tài liệu tham khảo: <https://www.w3schools.com/react/react_es6_spread.asp>[https://developer.mozilla.org/en-](https://developer.mozilla.org/en-US/docs/Web/JavaScript/Reference/Operators/Spread_syntax)

[US/docs/Web/JavaScript/Reference/Operators/Spread\_syntax](https://developer.mozilla.org/en-US/docs/Web/JavaScript/Reference/Operators/Spread_syntax)

const arrayOne = ['a', 'b', 'c'];

const arrayTwo = [1, 2, 3];

const arraysCombined = [];

function sum(x, y, z) {

return x + y + z;

}

const numbers = [1, 2, 3];

console.log(sum());

// expected output: 6

# #8 Destructuring assignment - Giản lược hóa cấu trúc Object/Array

### **Destructuring**

To illustrate destructuring, we'll make a sandwich. Do you take everything out of the refrigerator to make your sandwich? No, you only take out the items you would like to use on your sandwich.

*Để minh họa destructuring, chúng ta sẽ làm 1 cái sandwich. Liệu chúng ta có lấy mọi thứ trong tủ lạnh ra để làm sandwich không ? Dĩ nhiên là không rồi, chỉ cần lấy những nguyên liệu cần thiết mà thôi*.

Destructuring is exactly the same. We may have an array or object that we are working with, but we only need some of the items contained in these.

*Destructuring chính xác như vậy. Chúng ta hay làm việc với array/object, nhưng đôi khi, là chúng ta chỉ cần lấy một vài trường/thuộc tính của array/object.*

Destructuring makes it easy to extract only what is needed.

Destructuring giúp chúng ta lấy những cái chúng ta cần

**Destructuring = To destroy the structure of something**

### **Destructuring Objects the old way:**

const person = {name: 'An', age: 26, eyeColor: 'black', like: 'girl'};

const name = person.name;

const age = person.age;

console.log(name); //An

console.log(age); //26

**With destructuring:**

const person = {name: 'An', age: 26, eye Color: 'black', like: 'girl'};

const {age, name} = person;

console.log(name); //An

console.log(age); //26

Notice that the object properties do not have to be declared in a specific order. (khi dùng destructuring với object, *thứ tự của các thuộc tính không nhất thiết phải theo trình tự ban đầu trong object đó*)

### **Destructing Arrays**

const city = ['ha noi', 'da nang', 'sai gon', 'ca mau'];

//old way const hanoi = city[0]; const danang = city[1]; const hcm = city [2];

//With destructuring:

const [ hanoi, danang, hcm] = city;

When destructuring arrays, the order that variables are declared is important.

const [ hanoi, , , camau ] = city;

### **Exercise**

Use destructuring to extract only the third item from the array, into a variable named **tech** const **react** = [‘facebook’, 'all-in-one', 'javascript'];

const [ ] = react;

//complete this block code to print ‘bugs’

const dev = { salary: 2000, tool : ‘laptop’, like: ‘bugs’ };

const = dev;

console.log( ) //bugs

**#9 ES6 Ternary Operator - Toán tử điều kiện**

<https://english.stackexchange.com/questions/25116/what>[-follows-next-in-thesequence-unary-binary-ternary](https://english.stackexchange.com/questions/25116/what-follows-next-in-the-sequence-unary-binary-ternary)

### **Ternary Operator**

The ternary operator is a simplified conditional operator like **if / else**.

**Syntax: condition ? <expression if true> : <expression if false>**

Example

using if / else:

**Before:**

if (authenticated) {

    renderApp();

} else {

    renderLogin();

}

**After:**

authenticated ? renderApp() : renderLogin();

# #10 Optional chaining (?.)

### **Optional chaining '?.'**

The optional chaining ?. is a safe way to access nested object properties, even if an intermediate property doesn’t exist.

*(Toán tử ?. là một cách an toàn để truy cập thuộc tính của một object có nhiều lớp, kể cả khi thuộc tính đấy không tồn tại)*

let user = {}; // a user without "address" property

alert(user.address.street); // Error!

**How to fix it ?**

let user = {}; alert(user.address ? user.address.street : undefined);

let user = {}; // user has no address

alert( user.address && user.address.street ); // undefined (no error)

**Cú pháp: Value?.prop || undefined**

* works as **value.prop**, if value exists, otherwise (when value is undefined/null) it returns undefined. *Hoạt động bằng cách lấy giá trị* ***value.props****, nếu như value tồn tại. Nếu không, (khi value = undefined/null), nó sẽ trả về undefined*
* let user = {}; // user has no address alert( user?.address?.street ); // undefined (no error)
* let user = null;
* alert( user?.address ); // undefined alert( user?.address?.street ); // undefined
* user?.address?.street ?? defaultValue => remove undefined

### **Other variants: ?.( ), ?.[ ]**

**?.( ) is used to call a function that may not exist.**

* **Được dùng để truy cập một function** – mà nó có thể không tồn tại

let userAdmin = {

    admin() {

        alert("I am An");

  }

};

let userGuest = { };

userAdmin.admin?.( ) ; //I am An userGuest.admin?.( );

// nothing happens (no such method) => check function admin() có tồn tại hay không.

userGuest?.admin?.( )  // ???

* **Được dùng để truy cập thuộc tính, thông qua [ ]**

let key = "firstName";

let user1 = {

  firstName: "Nguyen"

};

let user2 = null;

alert( user1?.[key] ); // Hoi Dan IT alert( user2?.[key] ); // undefined

delete user?.name; // delete user.name if user exists

**We can use ?. for safe reading and deleting, but not writing**

Dùng **?.** là một cách an toàn để đọc/xóa dữ liệu, nhưng không được dùng với gán dữ liệu

**let user = null; user?.name = "An"; // Error, doesn't work**

// because it evaluates to: undefined = "An"

# Summary

* obj ?. a ?. b ?? defaultValue obj ?. a ?. b => undefined
* **The optional chaining ?.** syntax has three forms:
  + **obj?.prop** – **returns obj.prop** if obj exists, otherwise undefined.
  + **obj?.[prop]** – **returns obj[prop]** if obj exists, otherwise undefined.
  + **obj.method?.()** – calls **obj.method()** if obj.method exists, otherwise returns undefined.

Không nên lạm dụng ?. Nó là 1 cách an toàn để truy cập biến, tuy nhiên, đôi khi cũng là một cách để dấu bugs an toàn ^^

Tài liệu tham khảo:

[https://developer.mozilla.org/en-](https://developer.mozilla.org/en-US/docs/Web/JavaScript/Reference/Operators/Optional_chaining)

[US/docs/Web/JavaScript/Reference/Operators/Optional\_chaining](https://developer.mozilla.org/en-US/docs/Web/JavaScript/Reference/Operators/Optional_chaining)

Kiến thức cơ bản về React

# #11Setup ENV - Cài Đặt Môi Trường Dự Án

1. Cài đặt Visual Studio Code (Công Cụ Để Code Ngôn Ngữ Javascript)
2. Extension: formate: CSS/LESS/SCSS formatter ( Dùng để format code css )
3. **Cài đặt và sử dụng GIT.:**

[https://www.youtube.com/watch?v=-BtolPy15fg&list=PLncHg6Kn2JT6nWS9MRjSnt6Z-](https://www.youtube.com/watch?v=-BtolPy15fg&list=PLncHg6Kn2JT6nWS9MRjSnt6Z-9Rj0pAlo&index=1)

[9Rj0pAlo&index=1](https://www.youtube.com/watch?v=-BtolPy15fg&list=PLncHg6Kn2JT6nWS9MRjSnt6Z-9Rj0pAlo&index=1)

1. **Cài Đặt Môi Trường Node.JS**

Cần **Node.JS** để chạy dự án React. **Node.JS** là môi trường để chạy code javascript. Node.JS không phải là thư viện (library) hay là framework.

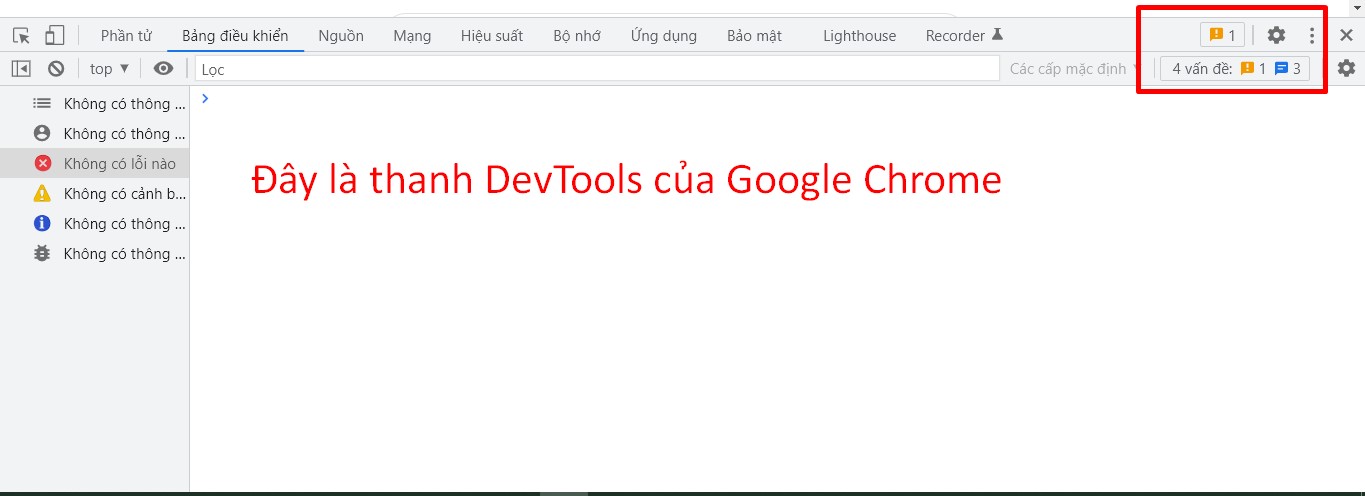
Ví dụ: cần cài đặt môi trường Windows để chạy phần mềm Microsoft Word.

tương tự, cần cài đặt môi trường **Node.js** để chạy dự án **React - viết bằng javascript**

1. **Cài Đặt Google Chrome**

* Chuyển ngôn ngữ Chrome sang tiếng anh:
* Chuyển đổi thanh devTool của Chrome sang tiếng anh:

Nhấn phím F12 => chọn biểu tượng bánh răng bên góc phải



# #12 React Overview - Tổng Quan Về React

### **React là gì ?**

* React là một Thư Viện (library) javascript để xây dựng giao diện người dùng (UI - User Interface)
* Ra đời vào tháng 5/2013. Version hiện tại : 18.2.0 (14/06/2022)
* Facebook (Meta) phát triển
* Learn Once, Write Anywhere. ReactJS (Web) vs React Native (Mobile)

Tài liệu về ReactJS : <https://reactjs.org/><https://www.npmjs.com/package/react>

### **React có thể làm được gì**

* Xây dựng website đơn giản như Facebook hay Instagram
* Ví dụ real time trading website:

[https://www.binance.com/en/futures/btcbusd\_perpetual?utm\_source=internal& utm\_medium=homepage&utm\_campaign=trading\_dashboard](https://www.binance.com/en/futures/btcbusd_perpetual?utm_source=internal&utm_medium=homepage&utm_campaign=trading_dashboard)

### **Tại sao lại dùng React**

* So sánh React/Vue/Angular: [https://npmtrends.com/@angular/core-vs-react-vsvue](https://npmtrends.com/@angular/core-vs-react-vs-vue)
* React (Facebook) vs Angular (Google) vs Vue - (Evan You / China)
* Nhiều công ty sử dụng React để phát triển sản phẩm ???

# #13 Hello World với React

### **Cài đặt dự án React**

Tài liệu: <https://reactjs.org/docs/create-a-new-react-app.html>

* Sử dụng React như là thẻ <script> ở đầu trang HTML
* Sử dụng Toolchains, ví dụ Create-react-app, vite

### **Đẩy code lên Github**

Yêu cầu: đã cài đặt thành công Git trên máy tính và đã có tài khoản Github

& xem series cơ bản học về cách sử dụng git

[https://www.youtube.com/watch?v=-BtolPy15fg&list=PLncHg6Kn2JT6nWS9MRjSnt6Z9Rj0pAlo&index=1](https://www.youtube.com/watch?v=-BtolPy15fg&list=PLncHg6Kn2JT6nWS9MRjSnt6Z-9Rj0pAlo&index=1)

B1: Tạo thư mục lưu trữ code trên Github (New Repository)

B2: Tại nơi lưu trữ code trên máy tính cá nhân (thư mục root), thực hiện các câu lệnh lần lượt theo thứ tự sau:

**git init** => câu lệnh này để khởi tạo thư mục git tại local, giúp quản lý code

**git add** . => câu lệnh này, viết từ add, sau đấy cách ra, có một dấu chấm cuối cùng **git commit -m** “nội-dung-commit”

**git remote**… paste câu lệnh copy trên github vào đây

**git push origin master** => câu lệnh này sẽ đẩy code lên github.

Done

# #14 Project structure - Kiến trúc dự án React

Tuỳ thuộc vào Tool, cấu trúc của dự án có khác nhau

# Tạo react project bằng CRA và vite

## **Tạo bằng CRA (create-react-app)**

* Mở terminal chạy lệnh:

**npx create-react-app myreact-app**

* Nếu chưa cài **create-react-app** cần install package [**create-react-app@5.0.1**](mailto:create-react-app@5.0.1)
* Nếu bị lỗi: [React JS npm start shows failed to compile web-vitals](https://stackoverflow.com/questions/65396568/react-js-npm-start-shows-failed-to-compile-web-vitals), Cài them gói web-**vitals**

**npm i web-vitals --save-dev**

* cd **myreact-app //**cd vào thư mục của project
* chạy **ứng dụng**

**npm start**

* truy **cập vào** [**http://localhost:3000**](http://localhost:3000) **xem kết quả**
* **Cấu trúc của project tạo bằng CRA**

****

## **Tạo bằng Vite**

**https://vite.dev/guide/**

* Mở **terminal và chạy lệnh sau:**

**npm create vite@latest**

* Điền tên dự án (ví dụ: my-react-app), hoặc nhấn Enter để dùng tên mặc định.
* Chọn framework: **React**.
* Chọn ngôn ngữ: **JavaScript** (React chỉ dùng JavaScript).
* Chuyển vào thư mục dự án:

**cd my-react-app**

* Cài đặt dependencies

**npm install**

* Chạy ứng dụng

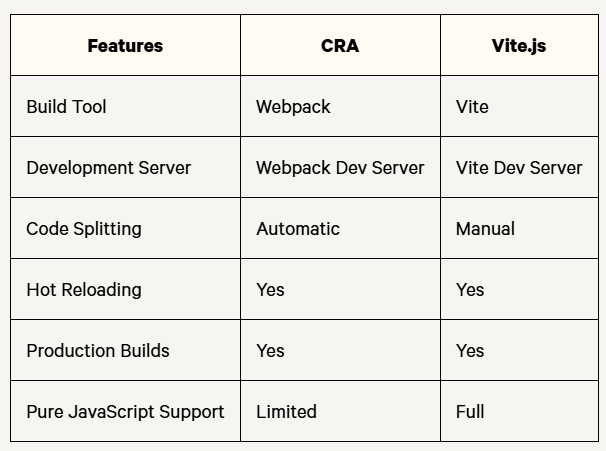
**npm run dev**

* Truy cập vào <http://localhost:5173/> xem kết quả
* **Cấu trúc project tạo bằng vite**

****

**Index.js** trong **src** có nhiệm vụ tổng hợp code react, nhúng vào div có **id=’root’** trong file **index.html** trong thư mục public để browser có thể truy cập được

## **The key differences between CRA and Vite.js:**

****

## **So sánh hiệu năng của webpack và vite**

[**https://storybook.js.org/blog/storybook-performance-from-webpack-to-vite/**](https://storybook.js.org/blog/storybook-performance-from-webpack-to-vite/)

Nếu mới làm quen với React hoặc muốn có một cách nhanh chóng và dễ dàng để thiết lập một dự án React mới, thì CRA là một lựa chọn tốt. CRA cung cấp mọi thứ bạn cần để bắt đầu và rất dễ sử dụng.

Nếu bạn đang tìm kiếm một công cụ xây dựng nhanh hơn và nhẹ hơn, hoặc bạn cần hỗ trợ đầy đủ cho JavaScript thuần túy, thì **Vite.js** là một lựa chọn tốt. Vite.js cũng là một lựa chọn tốt nếu bạn muốn kiểm soát nhiều hơn cách phân chia mã của mình.

Cuối cùng, công cụ tốt nhất sẽ phụ thuộc vào nhu cầu và sở thích cụ thể. Nếu bạn không chắc chắn nên chọn công cụ nào, thì nên dùng thử cả CRA và Vite.js để xem tool nào thích hợp

# Cấu trúc của react project

Thư mục **root** là nơi chứa tất cả mã nguồn của dự án.

1. **Các files trong thư mục root bao gồm:**

* **README.MD** : khi bắt đầu sử dụng 1 phần mềm, thì đây chính là file nên đọc đầu tiên. Trong bất kỳ dự án nào, file này chính là nơi ghi thông tin về dự án đấy, bao gồm cách cấu hình dự án, cách chạy dự án, thông tin tác giả…
* **package.json**: ghi thông tin về dự án được khởi tạo trong môi trường **Node.js,** đặc biệt là các thư viện (libraries/dependencies) đã được cài đặt trong dự án.
* **package-lock.json**: được tạo ra tự động khi chạy câu lệnh **npm install**. File này sẽ lưu trữ chi tiết thông tin của các thư viện đã được cài đặt.
* **gitignore**: file này quy định những file/folder nào không cần đẩy lên remote server (Github/Gitlab…). Từ ignore có nghĩa là bỏ qua

1. **Các thư mục con trong thư mục root:**

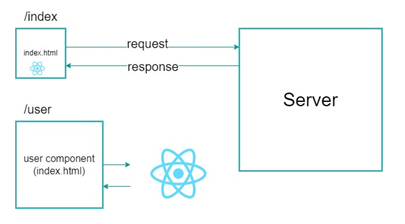
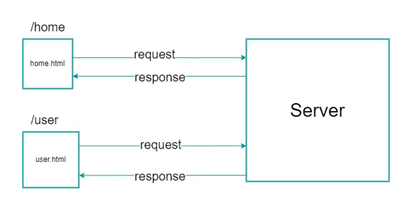
* **Thư mục src (viết tắt của source code)** : nơi lưu trữ code của dự án React, bao gồm:
  + **Thư mục redux**: cấu hình thư viện Redux để sử dụng với React.
  + **App.js** : là file component của dự án
  + **App.css:** là file giúp css cho component ở trên
  + **index.js**: nhúng tất cả logic xử lý javascript vào file này
  + **logo.svg**: file ảnh. Dùng svg để khi co giãn ảnh trông vẫn đẹp - reportWebVitals: sử dụng để đo lường hiệu năng website
* **Thư mục public**: nơi ứng dụng được chạy, khi run câu lệnh : **npm start**
  + **index.html**: file quan trọng nhất của React. Tất cả mã nguồn ứng dụng, sẽ được **webpack** bundle vào file này (bao gồm file **index.js** ở thư mục src). Ứng dụng React bản chất chỉ chạy mình file này.
  + **favicon.ico**: file logo của website
  + **manifest.json**: cung cấp thông tin của ứng dụng, như tên, hình ảnh… được sử dụng chủ yếu để hiển thị trên mobile
  + **robotstxt:** phục vụ mục đích SEO, crawl data của Google bot

# #15 How React Works ?

### **SPA - Single Page Application - Ứng dụng React là SPA.**

* **Browser** chỉ chạy 1 file duy nhất: **index.html**
* **React** kiểm soát thứ người dùng muốn nhìn, từ điều hướng trang cho tới hiển thị dữ liệu

### **Mô hình SSR - Server-side rendering (Multi page apps)**



### **Thứ tự chạy các file trong React:**

* **File index.jsx** : tổng hợp tất cả code **Component React** vào file này (bao gồm html, css và js). Sau đấy nhúng vào **div** có **id là root** bên trong file **index.html**
* **File index.html:** sẽ được nhúng phần xử lý logic (js,css) thông qua **webpack** và nội dung thông qua **div root** (thư viện react-dom) **Gọi là SPA**, vì đơn giản quá trình sử dụng web của người dùng là sử dụng file **index.html**. File **index.html** này là ‘**super file’**, khi đã bao gồm tất cả nội dung của website cũng như phần xử lý logic của nó.
* Mỗi lần người dùng điều hướng trang hay thao tác trên website, **phần xử lý logic** của **React** sẽ xử lý, tạo ra cảm giác mọi thứ đã có sẵn trên **website** này rồi, và làm hoàn toàn việc này ở browser (client)

# #16 React Component

### **Component là gì**

* **Component** là cách **React** tạo nên bố cục của một **website**, thay vì chúng ta code các khối HTML riêng lẻ, thì ở đây, chúng ta sử dụng Component
* Component khác HTML ở chỗ là nó sử dụng cú pháp **JSX. Với JSX**, ta có thể code logic Javascript cùng với HTML

### **Cách khai báo Component (Class Component)**

* **Class javascript** cần kế thừa (extend) lớp **Component của React** để trở thành **Component**
* Với component, chúng ta có hàm **render( )** giúp tạo nên giao diện website. Hàm render sử dụng cú pháp JSX
* **Cú pháp export default**, có nghĩa là trong file **js** ấy, chúng ta **chỉ export “1 hàm” duy nhất.**

// **foo.js export default function() {**

**console.log("hello!")**

**}**

//sau đó, dùng câu lệnh import để **sử dụng biến đã export**

**import** **foo** from "foo"; foo(); // hello!

# #17 State

### **Component State**

* Là Javascript Object
* Miêu tả trạng thái (state) hiện tại của Component: data/UI-state
* **State của Component** có thể được cập nhật, ví dụ như: đóng/mở Modal…

### **Sử dụng State**

* Khai báo: **state = { }** <= state là javascript object
* Sử dụng: { this.state.property }

# #18 React dev tool/ Redux dev tool

### **React devtool:**

### <https://chrome.google.com/webstore/detail/react>[-developertools/fmkadmapgofadopljbjfkapdkoienihi?hl=en](https://chrome.google.com/webstore/detail/react-developer-tools/fmkadmapgofadopljbjfkapdkoienihi?hl=en)

# #19 DOM Events

Tài liệu tham khảo về Events của React: <https://reactjs.org/docs/events.html>

* Lưu ý khi sử dụng xử lý sự kiện với React:
* Cú pháp sử dụng: bắt đầu bằng từ ‘**on’**, sau đấy là tên sự kiện viết hoa chữ cái đầu, ví dụ: onClick, onChange…
* Để sử dụng sự kiện, cần truyền vào 1 function,

ví dụ:

<button onClick= { **your-function-here** } > Submit </button>

# #20 setState - Cập nhập State cho ứng dụng React

* **Lưu ý khi sử dụng event với React Class:**
* Sử dụng **arrow function** để không phải bind keyword this
* Arrow function khắc phục được lỗi trên, vì mặc định, nó sẽ lấy biến this trong phạm vi gần nhất, tức là nó “auto” lấy biến this của component định nghĩa.
* **Cập nhật state của component React Class:**
* Sử dụng hàm **setState()**
* Hàm **setState** sẽ cần truyền vào 1 object, vì **state của React là object**
* Hàm **setState** sẽ tự động merge state, tức là nó sẽ chỉ cập nhật những state mới, những tate cũ (không thay đổi) sẽ được giữ nguyên, không bị ảnh hưởng.
* Sau khi hàm **setState**() được chạy, component được cập nhật lại state, dẫn tới hàm **render()** sẽ auto được chạy lại thay đổi state component sẽ dẫn tới re-render (chạy lại hàm render), điều này sẽ tạo cảm giác cho người dùng giao diện được cập nhật.
* (hàm render( ) của component có thể được chạy nhiều lần

# #21 Form in React

* Dùng event **onChange** với **input** để bắt sự kiện người dùng gõ bàn phím
* Dùng event **onSubmit** với **form** để bắt sự kiện nhấn nút button, hoặc nhấn Enter
* *Không dùng event onClick với button vì sẽ không bắt được hành động người dùng nhập input và nhấn Enter*
* Dùng event.**preventDefault( )** để cancel hành động mặc định của event, đối với form và event submit, chúng ta sử dụng để không cho website bị load lại (refresh)

# #22 Nesting Component - Component lồng

### **Lưu ý về form:**

* Thuộc tính **“value”** của input : **<input value={ stateReact} />**
* Sử dụng ‘**value’ để cho React quản lý giá trị của input**, sẽ tối ưu hiệu năng website hơn

### **Tại sao dùng nested component (component lồng nhau):**

* Giúp tái sử dụng code, code ít đi
* Giúp component có thể tái sử dụng được
* Tạo ra khái niệm cha và con. Cha nằm ngoài, bọc con nằm trong.

### **Lưu ý về code React Class (bad code)**

Không được (never) thay đổi state của React bằng cách sửa đổi trực tiếp biến state.

Ví dụ: this.state.name = ‘An’ <= cập nhật biến name giá trị An

Thay vào đấy, phải gọi qua hàm **setState**, hoặc, **clone state** ra biến mới:

* Cách 1: this.setState({ name: ‘Bình’ } )
* Cách 2: clone ra biến mới

Ví dụ:

let cloneState = { ... this.state };

cloneState.name = ‘Nguyen ‘ <= modify clone state, chứ không phải state React this.setState({...cloneState})

# #23 Props

### **Props là gì**

* **Props** là viết tắt của từ ‘**property’**
* **Props** là một javascript object (giống State, cũng là một object)
* Dịch nghĩa là ‘**property’**’ vì props sinh ra để giúp **component** **con** có thể kế thừa lại (sử dụng được) **property’** component cha để lại (ở đây là truyền data từ cha xuống con)

### **Cách sử dụng Props**

* Khai báo thông qua nơi gọi component con, với cú pháp :

**tên Biến = giá Trị**

Ví dụ : <ChildComponent name= “An” />

* Ở đây, tại ChildComponent sẽ được nhận props có tên là name, giá trị là ‘An’
* Sử dụng props, thông qua : **this.props.Tên-Props- Muốn-Truy-Cập**

### **Lưu về cách sử dụng Props**

* Để truyền props từ cha sang con, khi truyền props, sử dụng cặp dấu { }

Ví dụ: **<ChildComponent data= {any-data-here}/>**

* Có thể dùng cú pháp Giản lược hóa cấu trúc Object/Array , để có thể truy cập nhanh props

Ví dụ: **const { name } = this.props**;

* Cách làm trên, sẽ tương đường với việc viết code như sau:

**const name = this.props.name**;