

Chapter 1: React và Javascript ES6	3
#1 ES6 Variables	3
#2.ES6 Classes	6
#3.Object javascript (array)	9
#4 ES6 Arrow Functions	13
#5 ES6 Array Methods - Map và Filter	14
#6 Template literals (Template strings) - Dấu nháy chéo ` backticks	16
#7.Spread syntax (...) - Cú pháp toán tử mở rộng	18
#8 Destructuring assignment - Giảm lược hóa cấu trúc Object/Array	20
#9 ES6 Ternary Operator - Toán tử điều kiện	22
#10 Optional chaining (?.)	23
Chapter 2: Học React Một Cách Vừa Đủ	26
#11 Setup ENV - Cài Đặt Môi Trường Dự Án	26
#12 React Overview - Tổng Quan Về React	30
#13 Hello World với React	31
#14 Project structure - Kiến trúc dự án React	32
#15 How React Works ?	33
#16 React Component	34
#17 State	35
#18 React dev tool/ Redux dev tool	35
#19 DOM Events	35
#20 setState - Cập nhập State cho ứng dụng React	36
#21 Form in React	37
#22 Nesting Component - Component Cha Lồng Con	37
#23 Props	38
#24 Outputting list - Render Array/Object với React	39
#25 Conditional Output - Sử dụng câu điều kiện	40
#26 Function as props - Truyền Hàm từ Cha Sang Con	41
#27 CSS với React	42
#28 Image - Sử dụng hình ảnh với React	42
#29 Fragment	43
#30 Variables with JSX - Sử dụng biến số với JSX	44
#31 Delete data với Filter	44
#32 Recap - Tổng kết các kiến thức đã học	45
Chapter 3: Modern React - React với Hook	47
#33. React Lifecycle - Vòng đời ứng dụng React	47
#34. Stateful/Stateless Component	49
#35. useState Hook - Kiểm Soát State với Function Component	49

#36 Bài Tập: Sử Dụng useState Hook	50
#37 Giải Bài Tập useState Hook	50
#38 useEffect Hook - Sử Dụng LifeCycle với React Function Component	51
#39 Why Hook ? Tại Sao Chúng Ta Sử Dụng React với Hook	52
Chapter 4: Setup Dự Án Backend	53
#40 Tổng quan về dự án thực hành	53
#41 Tạo tài khoản Docker Hub	54
#42 Cài đặt Docker Desktop	54
#43 Run Docker Compose (todo)	55
#44 DBeaver - Kết Nối Database Postgres	56
#45 PostMan - Test APIs Backend	58
Chapter 5: Điều Hướng Trang với React Router v6	58
#46 Setup Bootstrap 5 & React Router Dom v6	58
#47 Design Header với Bootstrap Navigation	59
#48 Điều Hướng Trang với Links	59
#49 Nested Routes	59
#50 Active Link - NavLink	59
#51 Index Routes	60
#52 Design Homepage	60
#53 Design New Header	61
#54 Design Admin SideBar	62
#55 Setup Axios, React Toastify, React Paginate	62
Chapter 6: CRUD Users - Thêm/Hiển Thị/Cập Nhật/Xóa Người Dùng	63
#56 Modal Thêm Mới Người Dùng	63
#57 State Hóa Modal Add New User	63
#58 API Thêm Mới Người Dùng (CREATE)	64
#59 Validate data và React Toastify	64
#60 API Services - Customize Axios	65
#61 Hiển Thị Danh Sách Users (READ)	67
#62 Cập Nhật Danh Sách Users Khi Thêm Mới Thành Công	67
#63 Design Modal Update User	68
#64 API Cập Nhật User (UPDATE)	69
#65 Bài tập: Chức Năng Xem Chi Tiết User	69
#66 Design Modal Delete User	69
#67 APIs Delete User (DELETE)	69
#68 Hiển Thị Danh Sách User Phân Trang (Paginate)	69
#69 Design Login	70
#70 API Login - Đăng nhập	70
#71 Bài tập: Chức năng Register - Đăng Ký Người Dùng	70
#72 Chức Năng Register	70

Chapter 7: Quản Lý State Application với Redux	71
#73 Why Redux ? Tại Sao Lại Cần Redux	71
#74 Store - Lưu Trữ Data Redux	72
#75 Actions/Dispatch	72
#76 Reducer	74
#77 useSelector, useDispatch Hook	74
#78 Sử Dụng Redux Lưu Thông Tin Người Dùng	75
#79 Loading Bar - Hiện Thị Thanh Loading Khi Gọi APIs	75
#80 Redux persist - Xử lý Data Khi F5 Refresh	75
Chapter 8: Doing Quiz - Chức năng Bài Thi	76
#81 Design Danh Sách Bài Thi Của User - Display List Quiz	76
#82 Chi Tiết Bài Quiz - Sử dụng URL Params	76
#83 Process Data - Xử Lý Data Phía Frontend	77
#84 Design Quiz Layout - Tạo Base Giao Diện Bài Thi	77
#85 Design Question - Tạo Giao Diện Hiện Thị Question	77
#86 Xử Lý Data Khi Chọn Câu Trả Lời	77
#87. Build Data Trước Khi Submit API	78
#88. Submit Quiz - Nộp Bài Test	78
#89. Design Giao Diện Thêm Mới Bài Test	78
#90. API Thêm Mới Bài Thi	78
#91. Hiện Thị Danh Sách Bài Thi Admin	78
#92. Fix Lỗi ScrollBar	79
#93. Bài Tập Sửa/Xóa Bài Thi	79
#94. Design Base Giao Diện Thêm Mới Questions/Answers	79
#95. Tạo Fake Data Cho Giao Diện	79

Chapter 1: React và Javascript ES6

#1 ES6 Variables

1. 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 1 : var, let và const

2. Var

```
var name = 'Hoi Dan IT';
```

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.

3.let

```
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.

4.const

```
const y = 'eric';
```

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.

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
```

5. Exercise

Tài liệu tham khảo:

Create a variable that cannot be changed.

```
x = 6.9;
```

#2.ES6 Classes

1.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:

```
class Person {  
  constructor(name) {  
    this.name = name;  
  }  
}
```

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)

2.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");
myInformation.getAddress();

```

3. 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)

```

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();

```

The **super()** method refers to the parent class.

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.

4.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)

(object.property and object["property"])

OOP - Object-oriented programming (Lập trình hướng đối tượng)

Python:

```
class Person:
    def __init__(self, name, age):
        self.name = name
        self.age = age
```

```
p1 = Person("Eric", 26)
```

```
print(p1.name) // Eric
```

```
print(p1.age) // 26
```

PHP:

```
<?php
class Person {
    public $name;
    public $age;

    function __construct($name, $age) {
        $this->name = $name;
        $this->age = $age;
    }
}
```

```
//getter, setter method
function get_name() {
    return $this->name;
}
}
```

```
$p1 = new Person("Eric", 26);
echo $p1->get_name(); //Eric
```

```
?>
```

Java:

```
public class Person {  
    String name;  
    int age;  
  
    public Person(String name, int age) {  
        name = name;  
        age = age;  
    }  
  
    Person p1 = new Person("Eric", 26);  
    System.out.println(p1.name);  
}
```

1. Javascript Object ??? Object trong thế giới Javascript

(run a file with node.js: node file_name)

```
let person = "Eric";
```

=> This code assigns a **simple value** (Eric) 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

This code assigns **many values** (Eric, 26) to a variable named person:

```
let person = { name: "Eric", 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

2. Object Definition - Định nghĩa object

```
{ name1 : value1 , name2 : value2...}
```

```
const person = {firstName:"Eric", lastName:"HoiDanIT", 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)

```
const person = {  
    firstName:"Eric",  
    lastName:"HoiDanIT",  
    age:26,  
    eyeColor:"black"};
```

3.Object Properties - Thuộc tính của object

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

Property(thuộc tính)	Property Value (giá trị)
firstName	Eric
lastName	HoiDanIT
age	26
eyeColor	black

4. 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"]`

try:

```
person.lastName;  
person["lastName"];
```

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

But, JavaScript arrays are best described as arrays.

```
const person = ["Eric", "HoiDanIT", 26];
```

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: "HoiDanIT vs Eric",  
  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!";}
```

or

```
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)

2.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;
```

3.Exercise

Tài liệu tham khảo: https://www.w3schools.com/react/react_es6_arrow.asp

Complete this arrow function:

```
const hello =        "Hello World!";
```

#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.

1. 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-US/docs/Web/JavaScript/Reference/Global_Objects/Array/map

2.Filter

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-US/docs/Web/JavaScript/Reference/Global_Objects/Array/filter

3.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

Example table with jquery

Synonyms:

Template Literals

Template Strings

String Templates

Back-Tics Syntax

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)

```
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 "Eric" `;
```

2. Multi-line strings

Using normal strings, you would have to use the following syntax in order to get multi-line 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 }`

3.Exercise

Tài liệu tham khảo:

https://developer.mozilla.org/en-US/docs/Web/JavaScript/Reference/Template_literals

Input:

```
const base_url = "localhost:8080";  
const 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

1. 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]; ???
```

2. Push a new item to array

```
Let myArr = ["Eric", "HoiDanIT", "React"];
```

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

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

3. 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

$X \leq Y$

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

4.Exercise

Tài liệu tham khảo: https://www.w3schools.com/react/react_es6_spread.asp
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 - Giảm lược hóa cấu trúc Object/Array

1.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

2.Destructuring Objects

The old way:

```
const person = { name: 'Eric', age: 26, eyeColor: 'black', like: 'girl' };
const name = person.name;
const age = person.age;
console.log(name); //Eric
console.log(age); //26
```

With destructuring:

```
const person = { name: 'Eric', age: 26, eyeColor: 'black', like: 'girl' };
const { age, name } = person;
console.log(name); //Eric
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)

3. Destructuring 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;
```

4. 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-the-sequence-unary-binary-ternary>

1. Ternary Operator

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

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

Here is an example using if / else:

Before:

```
if (authenticated) {  
  renderApp();  
} else {  
  renderLogin();  
}
```

After:

```
authenticated ? renderApp() : renderLogin();
```

2. Exercise

Exercise:

Complete this ternary operator statement.

```
blue   renderBlue()   renderRed();
```

#10 Optional chaining (?.)

1.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**

2. Other variants: ?.(), ?.[]

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

Được dùng để truy cập một function - thứ có thể không tồn tại

```
let userAdmin = {  
  admin() {  
    alert("I am Eric");  
  }  
};
```

```
let userGuest = { };
```

```
userAdmin.admin?.() ; // I am Eric
```

```
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: "Hoi Dan IT"  
};
```

```
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 = "Eric"; // Error, doesn't work
```

```
// because it evaluates to: undefined = "Eric"
```

3.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-US/docs/Web/JavaScript/Reference/Operators/Optional_chaining

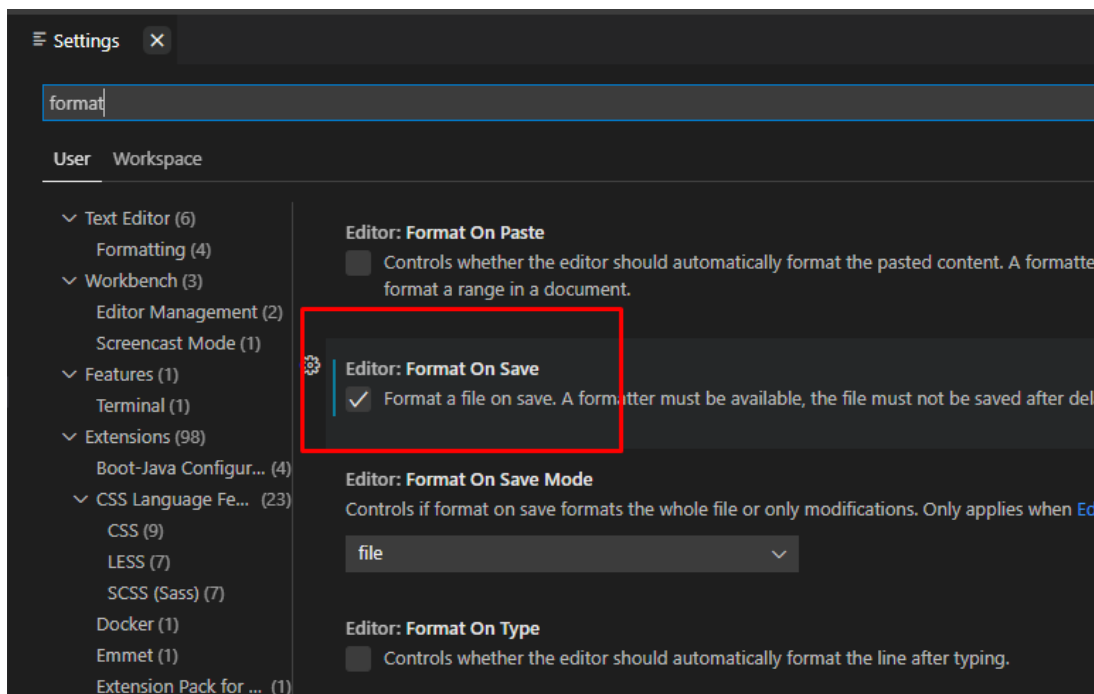
Chapter 2: Học React Một Cách Vừa Đủ

#11 Setup ENV - Cài Đặt Môi Trường Dự Án

1. Cài đặt Visual Studio Code (Công Cụ Để Code Ngôn Ngữ Javascript)

Download tại đây: <https://code.visualstudio.com/download>

- **Mở Terminal: View -> Terminal (hoặc phím tắt Ctrl + `)** dấu backtick dưới phím ESC. Dùng terminal để gõ các câu lệnh cần thiết, như cài đặt thư viện sử dụng, chạy dự án...
- **Tự động format code khi nhấn lưu (Ctrl + S)**
Mở Cài Đặt : File -> Preferences -> Settings . Gõ 'format' ở thanh tìm kiếm, tích vào 'Format On Save' như trong hình



- Extension mình sử dụng: **formate: CSS/LESS/SCSS formatter**
(Dùng để format code css)

2. Cài đặt và sử dụng GIT. Đi Làm Không Thể Không biết tới GIT

Tham khảo khóa học về GIT Siêu Tốc & Siêu Cơ Bản Tại đây:

<https://www.youtube.com/watch?v=-BtolPy15fg&list=PLncHg6Kn2JT6nWS9MRjSnt6Z-9Rj0pAlo&index=1>

- **Yêu cầu: Cài đặt thành công Git tại máy tính cá nhân, có tài khoản Github, biết đẩy code của mình lên Github (xem hết 5 video đầu tiên của link trên)**

3. Cài Đặt Môi Trường Node.JS

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

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

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

Download Node.JS (mới nhất) tại đây: <https://nodejs.org/en/download/>

Đối với khóa học này, mình sử dụng Node.JS v14.17.0.

Tải tại đây: <https://nodejs.org/download/release/v14.17.0/>

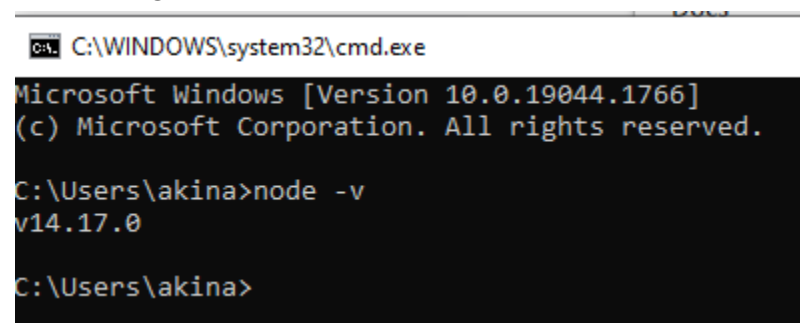
Với hệ điều hành windows, chọn file sau:

node-v14.17.0-x64.msi => ứng với windows 64bit

node-v14.17.0-x86.msi => ứng với windows 32bit

Kiểm tra xem đã cài đặt thành công Node.JS hay chưa ?

Mở CMD, gõ câu lệnh: **node -v** => nó sẽ hiển thị lên version Node.js đã cài đặt



```
C:\WINDOWS\system32\cmd.exe

Microsoft Windows [Version 10.0.19044.1766]
(c) Microsoft Corporation. All rights reserved.

C:\Users\akina>node -v
v14.17.0

C:\Users\akina>
```

Bạn nào đã biết Node.JS rồi, muốn sử dụng nhiều version của Node.JS trên cùng một máy tính, thì có thể tham khảo sử dụng NVM - Node version manager :

<https://www.youtube.com/watch?v=ccjKHLyo4IM>

Lưu ý: Chỉ cài đặt nhiều version của Node.Js khi máy tính của bạn chạy nhiều dự án Javascript với các version của Node.JS khác nhau. Nếu như đây là lần đầu tiên bạn làm quen và sử dụng Node.js, thì không cần cài đặt NVM, chỉ cần cài đặt thành công 1 version của Node.JS là được

Yêu cầu: Cài đặt thành công Node.Js trên máy tính, cụ thể là version 14.17.0

Khi gõ câu lệnh kiểm tra version của node.js (node -v) hiển thị kết quả 14.17.0

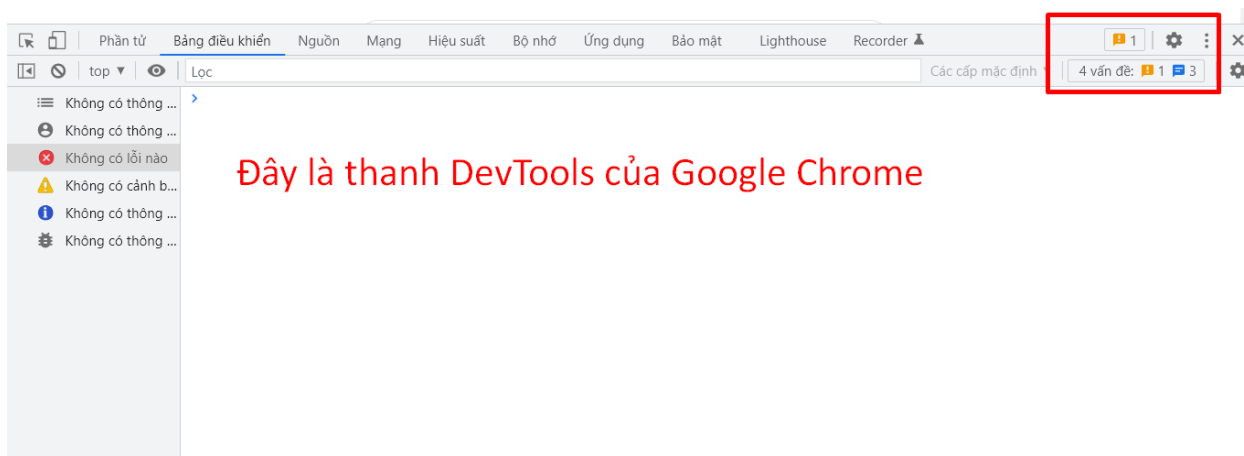
4. Cài Đặt Google Chrome

- Chuyển ngôn ngữ Chrome sang tiếng anh:

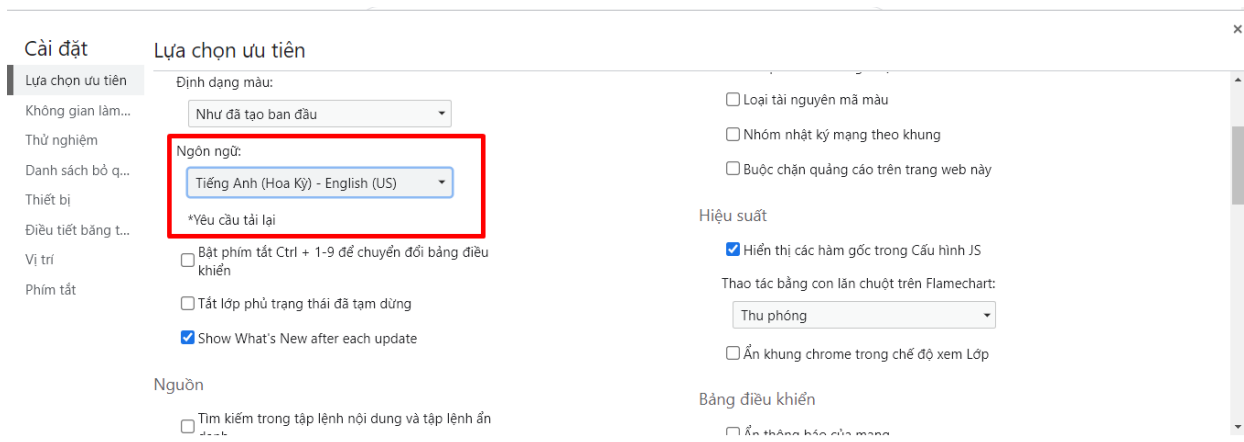
https://www.youtube.com/watch?v=GVwefuqLniM&list=PLncHg6Kn2JT6E38Z3kit9Hnif1xC_9Vql&index=11

- Chuyển đổi thanh devTool của Chrome sang tiếng anh:

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



=> chọn ngôn ngữ Tiếng Anh



=> Tắt Chrome. Bật lại sẽ thấy cập nhật ngôn ngữ

Lý do chuyển đổi sang tiếng anh là về lâu dài, những thuật ngữ, công cụ sử dụng đều là tên tiếng anh, dịch ra tiếng việt có nghĩa “rất đố”, và khi các bạn search google nó không ra đâu @@

Yêu cầu: Chuyển đổi Google Chrome sang ngôn ngữ Tiếng Anh

5.SOS Thì Làm Thế Nào - Khi Cần Hỗ Trợ Thì Nên Làm Gì ???

Youtube Hỏi Dân IT: <https://www.youtube.com/channel/UCVkBckjObNZiXavfAE1-fA/>

Fanpage Chính Thức của Hỏi Dân IT: <https://www.facebook.com/askITwithERIC>

Lưu ý: các bạn học viên mua khóa học trực tiếp trên Udemý (thanh toán qua Paypal/Visa/Master), thì chủ động inbox qua facebook ở trên cho mình, mình sẽ add các bạn vào group riêng dành cho học viên Udemý.

Khi gặp lỗi trong quá trình học tập, có câu hỏi cần giải đáp, thì các bạn đăng lên group Facebook dành cho học viên Udemý, và inbox trực tiếp cho mình qua Fanpage Hỏi Dân IT để được hỗ trợ.

Với các học viên mua “lậu” khóa học của mình (qua các kênh “hack” Udemý), không mua trực tiếp qua mình (Fanpage Hỏi Dân IT) và “Udemý” thì:

- Mình hi vọng các bạn ý thức được hành động của các bạn đang làm. Hãy biết trân trọng thành quả người khác làm ra, đặc biệt là các thành quả phải “vất óc” suy nghĩ. Tất cả là chất xám, chứ không phải nằm ngủ mà tạo ra được.
- Udemý và Facebook nó đều ghi logs lịch sử, thành ra ai mua, mình đều biết. Khi các bạn mua lậu (bỏ chi phí cho bọn ‘bán lậu’), các bạn đã mất kinh phí, tuy nhiên, lại mất đi quyền lợi của học viên (support 1:1 từ mình trong quá trình học)
- Với khóa học này, mình biết nhiều bạn, đặc biệt các bạn sinh viên sẽ không có điều kiện để tham gia. Mình hiểu cảm giác này, vì mình đã từng qua thời sinh viên giống các bạn. Bạn nào khó khăn quá, thì cứ inbox qua Fanpage Facebook, mình sẽ có giải pháp giúp đỡ các bạn, việc gì phải đi “ăn trộm” như vậy.
- Với phần thực hành của dự án này, mình sẽ tìm cách mã hóa, và chỉ những học viên mua “trong sạch” mới dùng được (kiểu như đánh license)

Với các bạn mới bắt đầu, mình biết các bạn sẽ không làm được gì, chỉ biết vút không khóa học không dùng được.

Với các bạn có trình độ, thì mình tin rằng các bạn biết cái gì nên làm. Ví dụ, mình bỏ 2h để mã hóa khóa học, thì các bạn sẽ dành x2, x3 thời gian đấy để crack khóa học. Liệu nó có đáng để dành thời gian đánh đổi, khi nó chỉ bằng “vài cốc cafe” các bạn uống hàng ngày ???

#12 React Overview - Tổng Quan Về React

1. 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>

2. 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

3. Tại sao lại dùng React

- So sánh React/Vue/Angular:
<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 ???

4. Lưu ý khi học React

- Cần biết HTML, CSS và Javascript cơ bản
- Sử dụng Javascript hay Typescript
- Sử dụng kết thúc file .js/.ts hay là .jsx/.tsx

#13 Hello World với React

1. 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

Download source code dự án:

https://drive.google.com/file/d/17u_Q40NVP95YIPhtW6JqnXjkHQZ18Bz/view?usp=sharing

2. Chạy dự án React tại local (máy tính cá nhân)

Yêu cầu: đã cài đặt thành công Node.js và Git trước khi thực hiện phần này

- Dùng câu lệnh : **npm i** hoặc **npm install** để cài đặt các thư viện cần thiết. Câu lệnh npm i và npm install là một. Từ i là viết tắt của từ install
- Dùng câu lệnh: **npm start** để chạy dự án. Mặc định, dự án React sẽ chạy trên localhost cổng 3000 (port)
- Biết sử dụng terminal VS Code để biết dự án đã chạy thành công chưa ??? Bonus thêm khái niệm localhost/port

3. Tại sao lại dùng Toolchains

- Hot Reloading
- Babel Compiler
- Webpack
- Ready for production

4. 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=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

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 làm lơ, làm bộ không biết gì :v

2. 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. Sẽ giải thích chi tiết sau khi học đến kiến thức về Redux
- **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

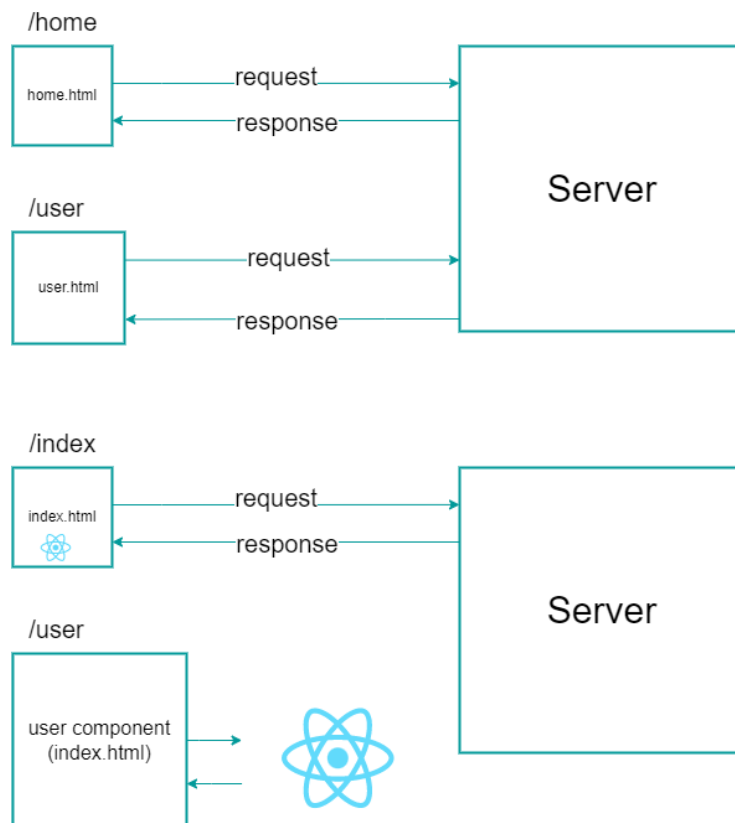
- **index.html**: file quan trọng nhất của React. Tất cả mã nguồn ứng dụng, sẽ được webpack bundle vào file này (bao gồm file index.js ở thư mục src). Ứng dụng React bản chất chỉ chạy mình file này.
- **favicon.ico**: file logo của website
- **manifest.json**: cung cấp thông tin của ứng dụng, như tên, 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 ?

1.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)



Với React, thứ tự chạy các file sẽ là:

- **File index.js** : 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

1. Component là gì

- Component là cách React tạo nên bố cục của một website. Giống như trò chơi xếp hình, 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, chúng ta có thể code logic Javascript cùng với HTML

2. Cách khai báo Component (Class Component)

(Về function component, chúng ta sẽ học ở chương sau)

- **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 này sử dụng cú pháp JSX

3. Một vài lưu ý

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 đây, dùng câu lệnh import để sử dụng biến đã export
```

```
import foo from "foo";
```

```
foo(); // hello!
```

Khi import, đôi khi chúng ta sử dụng 1 dấu chấm,

```
import { increaseCounter, decreaseCounter } from './redux/action/counterAction';
```

Đôi khi chúng ta sử dụng 2 dấu chấm

```
import { INCREMENT, DECREMENT } from '../action/counterAction';
```

Thì ở đây:

. là current directory (thư mục hiện tại)

.. parent directory (thư mục cha)

Với VS Code, khi click vào file 1 lần, chúng ta sẽ xem file ở chế độ preview. Thành ra, để xem nhiều file độc lập, các bạn nên có thói quen nháy đúp (click 2 lần) để xem files nhé

#17 State

1.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...

2.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-developer-tools/fmkadmapgofadopljbjfkapdkoienihi?hl=en>

Redux devtool:

<https://chrome.google.com/webstore/detail/redux-devtools/lmhkpmbekcpmknklieibfkpmmfiblj?hl=en>

#19 DOM Events

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

Một vài 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 nhật 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

Lưu ý về hàm setState của React Class, so với React Function Component(sau này mình sẽ nhắc lại điều này khi học về React Hook):

- **Hàm setState sẽ tự động merge state**, tức là nó sẽ chỉ cập nhật những state mới, những state 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 các bạn nhé :v)

#21 Form in React

Làm quen với form

- 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. Ở đây, đối với form và event submit, chúng ta sử dụng để không cho website bị load lại (refresh)

#22 Nesting Component - Component Cha Lồng Con

Lưu ý về form:

Thuộc tính "value" của input : `<input value={stateReact} />`

Sử dụng 'value' để cho React quản lý giá trị của input, như vậy 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 = 'Eric'` <= cập nhật biến name giá trị Eric

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

Cách 1: `this.setState({ name: 'Eric' })`

Cách 2: clone ra biến mới

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

```
cloneState.name = 'Hỏi Dân IT ' <= modify clone state, chứ không phải state React  
this.setState({...cloneState})
```

#23 Props

1. Props là gì

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

2. Cách sử dụng Props

- **Khai báo thông qua nơi gọi component con**, với cú pháp : **tên Biến = giá Trị**
Ví dụ : `<ChildComponent name= "Eric" />`
Ở đây, tại ChildComponent sẽ được nhận props có tên là name, giá trị là 'Eric'
- **Sử dụng props, thông qua : `this.props.Tên-Props`**- Muốn-Truy-Cập

3. 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 tại video #8 Destructuring assignment** - Giảm lược hóa cấu trúc Object/Array , để có thể truy cập nhanh props
Ví dụ: `const { name } = this.props;`

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

```
const name = this.props.name;
```

#24 Outputting list - Render Array/Object với React

DRY - Don't repeat yourself : nếu 1 khối code, được code đi code lại nhiều lần, thì kiểu gì cũng có cách để code tốt hơn - tối ưu hóa code

Coding convention: quy định (quy tắc) đặt ra để viết code cho chuẩn (giúp tối ưu và hạn chế bugs)

Tại sao dùng Map để render List với React ?

- Map trả ra 1 array mới, và không làm ảnh hưởng tới array ban đầu
- **Đối với React, cần thuộc tính 'key' trong vòng map**, để giúp React tối ưu hóa hiệu năng. Chúng ta không thấy thuộc tính 'key' - chỉ React thấy, tuy nhiên, React dùng thuộc tính này để biết chúng ta đang thao tác với phần tử HTML nào.
- Không dùng vòng lặp for hay while để render giao diện React vì:
Code dài hơn và nó không trả ra new array,
chi tiết xem tại video **#5 ES6 Array Methods - Map và Filter**

#25 Conditional Output - Sử dụng câu điều kiện

1. React Strict Mode

- Làm cho component tự động render 2 lần, thành ra, trong khóa học, bị tình trạng **console.log chạy 2 lần**
- Strict Mode được sinh ra để giúp phát hiện các lỗi tiềm ẩn có thể xảy ra trong app
- Strict Mode chỉ chạy với môi trường development, môi trường production không chạy

Chi tiết về Strict Mode xem tại đây: <https://reactjs.org/docs/strict-mode.html>

2. Câu điều kiện

- Dùng câu điều kiện để **"code ít hơn"**

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

Bổ trợ kiến thức, xem tại video #9 ES6 Ternary Operator - Toán tử điều kiện

- Toán tử && là điều kiện "và".
Nó sẽ bằng câu lệnh: **if(điều kiện) { kết quả điều kiện 'và' }**
- Toán tử ! sẽ trả ra giá trị phủ định của biến
VD: let result = false; let kq = true;
Let a = ! result => a = true;
Let b = ! kq => b = false

#26 Function as props - Truyền Hàm từ Cha Sang Con

Function as props

- Truyền tương tự như truyền 1 variable (biến số) từ component cha sang con
- Để sử dụng function, sử dụng keyword **this.props.Tên_Function()**

Kiến thức bổ trợ:

- **Toán tử ...** gọi là toán tử mở rộng : spread syntax , giúp copy phần tử bên trong mảng và object. **Chi tiết xem tại video #7.Spread syntax (...)** - **Cú pháp toán tử mở rộng**
- **Hàm Array.unshift**, giúp thêm 1 phần tử mới (hoặc nhiều phần tử 1 lúc) vào vị trí đầu tiên (index = 0) của array

Hàm này trả ra chiều dài (length) của mảng sau khi được cập nhật

Thành ra, trong video, khi code như thế này:

newUsers = newUsers.unshift(userObj) => biến newUsers = 4,
tức là chiều dài của mảng newUsers sau khi thực hiện unshift là 4 (do có 4 phần tử)
Lúc này sẽ dẫn đến lỗi **newUsers.map is not a function**, **bởi vì map là hàm được thực hiện với Array**, newUsers không phải là Array (giá trị là 4) nên gây ra lỗi.

Về hàm unshift và push đối với Array, xem tại đây:

https://developer.mozilla.org/en-US/docs/Web/JavaScript/Reference/Global_Objects/Array/unshift

https://developer.mozilla.org/en-US/docs/Web/JavaScript/Reference/Global_Objects/Array/push

#27 CSS với React

Có 2 trường phái code CSS trong thực tế hiện nay:

1 là sử dụng CSS bên trong Component, giống kiểu code cho React Native.

Ví dụ tiêu biểu là **styled-component**.

<https://styled-components.com/>

2 là sử dụng CSS bên ngoài Component, dưới dạng sass, less... Trong khóa học này, mình hướng dẫn các bạn sử dụng SASS để code CSS, với lý do:

- Nó giúp tiết kiệm thời gian code CSS
- Phong cách code gần gũi với CSS
- Mình thích trường phái 2 hơn là trường phái 1 ở trên

Ở đây, việc viết file CSS thuần (.css) hay style inline (sử dụng thuộc tính style với HTML) mình không đề cập, vì trong thực tế đi làm, không ai code CSS như vậy cả @@

Về công cụ SASS:

- Học nhanh về SASS tại đây: <https://www.w3schools.com/sass/>
- Sử dụng trong React, bằng cách đặt tên file với **tiền tố .scss**
- Cài đặt thư viện sass cho dự án React:
npm install --save-exact sass@1.53.0

#28 Image - Sử dụng hình ảnh với React

Lưu ý về sử dụng image trong ứng dụng React:

- **Sử dụng hình ảnh với create-react-app:**
<https://create-react-app.dev/docs/adding-images-fonts-and-files/>
- Không nên lưu hình ảnh trong thư mục public vì hiệu năng ứng dụng sẽ không cao, khi không tối ưu thông qua webpack
- Lưu ảnh trực tiếp trong source code, và dùng câu lệnh import để sử dụng
- Nếu ứng dụng có nhiều hình ảnh, thì nghĩ giải pháp để lưu ảnh, React (frontend) chỉ là nơi hiển thị:
 - + Lưu ảnh trong database
 - + Lưu ảnh trong server backendỞ đây, react sẽ nhận lại URL của ảnh và hiển thị => không liên quan gì tới việc lưu trữ ảnh bên trong ứng dụng React (React không l=> hiệu năng cao)

#29 Fragment

Tại sao cần Fragment ?

Tài liệu tham khảo về Fragment: <https://reactjs.org/docs/fragments.html>

Fragment giúp chúng ta **code “ít đi” và “không thừa thãi”**.

Cú pháp: `<React.Fragment> </React.Fragment>` hoặc `<> </>`

Thông thường, đối với hàm `render()` của JSX, **chúng ta cần trả ra (return) 1 khối block hay còn gọi là 1 element**, thành ra, đôi khi chúng ta luôn cần 1 phần tử `<div>` bọc ngoài cùng để đảm bảo nguyên tắc này.

Ví dụ:

```
render( ) {  
  return (  
    <div>  
      <Component1/>  
      <Component2/>  
    </div>  
  )}
```

Tuy nhiên, việc tạo ra 1 div bọc ngoài không cần thiết như vậy, đôi khi sẽ làm **vỡ ‘layout’, đặc biệt là khi CSS giao diện** => Fragment sinh ra để giải quyết vấn đề trên.

Fragment vẫn đảm bảo nguyên tắc return 1 block của JSX, đồng thời, **không render thêm bất cứ thứ gì vào DOM HTML**, thành ra không bị tình trạng vỡ layout khi CSS.

Ví dụ trên, khi dùng với Fragment:

```
render( ) {  
  return (  
    <>  
      <Component1/>  
      <Component2/>  
    </>  
  )}
```

#30 Variables with JSX - Sử dụng biến số với JSX

1. Component khác gì HTML

Component = Template JSX (HTML) + Logic Javascript

2. Variables

- Đối với String, Number thì dùng bình thường trong JSX
- Để check giá trị của Object, Array => convert sang String, sử dụng **JSON.stringify(data)**

https://www.w3schools.com/js/js_json_stringify.asp

#31 Delete data với Filter

Hàm filter giúp xóa phần tử, vì nó sẽ 'lọc' các phần tử của mảng theo tiêu chí đề ra.

Chi tiết về hàm filter với array:

https://developer.mozilla.org/en-US/docs/Web/JavaScript/Reference/Global_Objects/Array/filter

Tài liệu tham khảo về hàm filter xem tại **video #5 ES6 Array Methods - Map và Filter**

#32 Recap - Tổng kết các kiến thức đã học

1. Cách code React theo OOP - lập trình hướng đối tượng

- Cần có **hàm constructor** (hàm tạo), và hàm này sẽ được chạy đầu tiên trong component.

Cú pháp:

```
constructor(props) {  
    super(props);  
    this.state = { //init state here }  
}
```

- Hàm constructor giúp khởi tạo các giá trị ban đầu của state component, đồng thời, với việc sử dụng `super(props)`, component con sẽ kế thừa được props từ component cha truyền xuống
- Xuyên suốt khóa học, **chúng ta không code hàm constructor, tuy nhiên, code vẫn chạy bình thường**, lý do vì:
khi tạo app react bằng thư viện `create-react-app`, nó đã cấu hình thư viện 'babel' - một trình dịch code javascript.
Thư viện babel đã tự động thêm hàm constructor vào cho chúng ta (nếu như không viết) khi nó dịch code của ứng dụng react

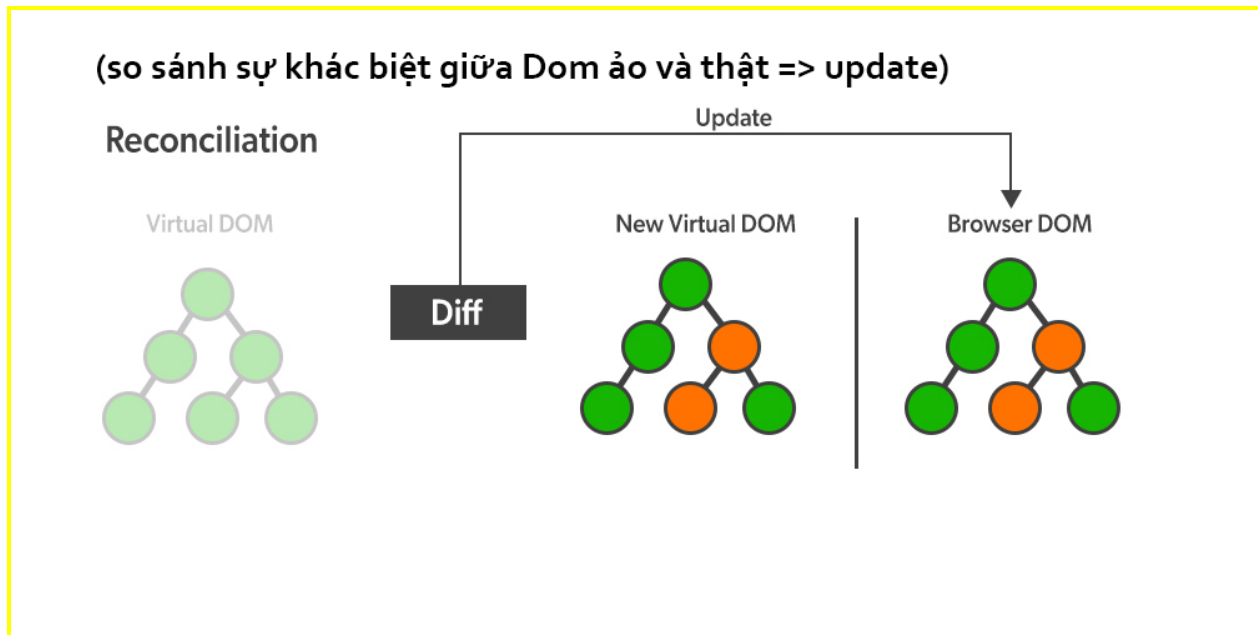
2. HTML DOM

- Tài liệu về javascript DOM: https://www.w3schools.com/js/js_htmlDOM.asp
- Hiểu đơn giản về DOM (Document Object Model) - chính là cây HTML giúp hiển thị lên giao diện website

3. Làm sao để component nó render lại (vẽ lại giao diện mới)

- **Thay đổi state** của component, thông qua hàm `setState`
- **Thay đổi props** của component cha truyền xuống component con.

4. React Virtual DOM



- **Virtual DOM** (cây DOM ảo), là cây DOM do React tạo ra khi chạy ứng dụng, và chỉ có React có thể nhìn thấy cây DOM này.
- **Browser DOM** (cây DOM thật), là cây DOM trình duyệt web dùng, chính là cây DOM mình đề cập ở mục 2 (chúng ta có thể nhìn thấy/thay đổi cây DOM này)
- **Ứng dụng React chạy nhanh, có hiệu năng cao vì: nó sử dụng cây virtual DOM.**

Mỗi khi có sự thay đổi State/Props, React sẽ so sánh quá khứ và hiện tại của cây DOM ảo này, tìm điểm khác biệt giữa quá khứ và hiện tại, và **chỉ update cây DOM thật 'nơi bị thay đổi'**, chứ **không update toàn bộ cây DOM** thật (theo kiểu ghi đè)

Ví dụ trong quá trình code React, chúng ta có thuộc tính 'key' trong vòng lặp map.

React cần thuộc tính key này cho cây DOM ảo, để giúp nó biết chúng ta đang thao tác với phần tử HTML nào. Key lúc này, sẽ giống với thuộc tính ID, giúp định danh phần tử HTML

Mỗi khi chúng ta thêm/xóa/cập nhật phần tử HTML ở trên, dựa vào key, React sẽ tốn ít thời gian nhất để biết phần tử nào đang thêm/xóa/cập nhật, từ đó update cây DOM thật "nơi bị thay đổi", giúp hiệu năng ứng dụng React luôn cao ^^

Chapter 3: Modern React - React với Hook

#33. React Lifecycle - Vòng đời ứng dụng React

1. Mô hình vòng đời của React (diagram):

<https://projects.wojtekmaj.pl/react-lifecycle-methods-diagram/>

2. Tại sao cần quan tâm tới lifecycle (vòng đời) của React (cụ thể là component) ?

Nếu 1 ứng dụng chỉ dùng để hiển thị thông tin (dữ liệu luôn không đổi), thì chúng ta chỉ cần HTML/CSS, và không cần tới React.

Tuy nhiên, có bao giờ các bạn tự hỏi, là làm sao 1 website, không cần reload lại trang, dữ liệu nó tự động được cập nhật.

Ví dụ như website trading online:

https://www.binance.com/en/futures/btcbusd_perpetual?utm_source=internal&utm_medium=homepage&utm_campaign=trading_dashboard

Hoặc:

Bạn đang dùng Facebook, tự động nhận được notification thông báo.

Ở đây, mình không muốn nói tới công nghệ đứng sau nó (như socketIO - nhiều bạn có thể biết), mà mình muốn nói tới sự thay đổi giao diện của ứng dụng React theo thời gian.

Và để làm được 2 ví dụ kể trên, bắt buộc chúng ta phải dùng React lifecycle.

3. React lifecycle - Vòng đời của Component

Tương tự việc các bạn chơi game, chúng ta sẽ theo trình tự:

Nhấn nút play -> Game chạy -> Game có thể pause -> Game có thể resume -> Thoát trò chơi (end game).

Ở đây, game **được sinh ra** khi các bạn nhấn nút 'play'. Game **được phát triển** khi các bạn tiến hành chơi game, và game **kết thúc (die)** khi thoát khỏi trò chơi. Đây chính là vòng đời, lặp đi lặp lại mỗi lần chơi game.

OOP - lập trình hướng đối tượng đang miêu tả sự sinh ra, phát triển, và die giống hệt con người (lifecycle)

Với component React, chúng ta có 3 giai đoạn như trên, được gọi là **Mounting** (born - được sinh ra), **Updating** (develop - quá trình phát triển) và **Unmounting** (die - bị xóa đi)

Giai đoạn Mounting - Component được sinh ra

Thứ tự chạy các function trong Component lần lượt là:

- Constructor: giúp khởi tạo state cho component
- Render () : giúp chèn HTML vào cây DOM
- ComponentDidMount: sau khi đã có HTML trong DOM, chúng ta có thể thao tác tùy ý tại đây, ví dụ như gọi APIs... **và chỉ chạy duy nhất 1 lần.**

Giai đoạn Updating - Component được cập nhật (dữ liệu bị thay đổi)

Khi nào **giai đoạn Updating xảy ra**:

- Khi có sự **thay đổi props** từ Cha truyền xuống con
- Khi có sự **cập nhật state**, thông qua hàm setState

Thứ tự chạy:

- Render()
- componentDidUpdate(prevProps, prevState, snapshot)
<https://reactjs.org/docs/react-component.html#componentdidupdate>

Bằng cách dùng hàm này, chúng ta có thể **so sánh sự khác biệt giữa quá khứ và hiện tại của props/state**, từ đó xử lý logic chúng ta mong muốn

Lưu ý ở đây là, hàm **Render ()** và **componentDidupdate()** có thể được chạy nhiều lần, chỉ cần có sự thay đổi của props/state thì nó sẽ chạy lại (giống hệt một vòng lặp, cứ thay đổi là chạy lại)

Giai đoạn Unmounting - Component bị xóa khỏi màn hình

Khi nào **giai đoạn Unmounting xảy ra**: khi component không còn tồn tại trên DOM nữa, ví dụ như việc điều hướng trang chẳng hạn.

Tại sao cần Unmounting ? Đôi khi các bạn muốn tối ưu hóa hiệu năng website, hoặc cần phải xử lý logic khi 1 component không còn tồn tại trên màn hình nữa, thì chúng ta sẽ cần sử dụng hàm này.

#34. Stateful/Stateless Component

1. Phân biệt Stateful/Stateless

Đối với React, một component, có **sử dụng State để kiểm soát data**, thì gọi là **Stateful**.

Còn với component, **không sử dụng State**, chỉ sử dụng props để hiển thị dữ liệu, thì được gọi là **Stateless**

2. Cách chuyển đổi từ Stateful sang Stateless Component

- Sử dụng **Arrow function**
- **Không sử dụng** hàm **constructor** và **keyword this**
- **Không sử dụng** hàm **render**, thay vào đây là keyword **return**
- **Props** được truyền **tự động** từ cha xuống con

Ví dụ về stateless component:

```
const myComponent = (props) => {  
  
    return ( //code JSX here )  
  
}
```

#35. useState Hook - Kiểm Soát State với Function Component

useState giúp sử dụng được State cho Function Component

Cú pháp: `const [state, setState] = useState(initValue);`

Ở đây, chúng ta code như thế này, vì đang sử dụng cú pháp Array Destructuring, chi tiết xem tại video #8 Destructuring assignment - Giảm lược hóa cấu trúc Object/Array.

Cụ thể: hàm `useState ()` là 1 array, trả ra 2 tham số. Tham số đầu tiên chính là 'tên của State' và tham số thứ 2 chính là 'hàm có thể cập nhật giá trị của state' => dùng array destructuring để lấy ra 2 tham số này.

Ví dụ: **const [name, setName] = useState('Eric');**

Cách viết trên, sẽ tương tự: `this.state = { name: 'Eric' }`

Khi muốn **cập nhật giá trị của state 'name'**,
chúng ta dùng hàm **setName('giá trị cập nhật')**

Nó sẽ tương đồng với react class:

this.setState({ name: 'giá trị cập nhật' })

Lưu ý:

- Với React Hook, không còn tồn tại **keyword 'this'**
- Tương tự, không tồn tại **hàm setState** khi sử dụng hook
- **Tư duy khi dùng Hook và Class là giống nhau**, chỉ khác cách khai báo.

#36 Bài Tập: Sử Dụng useState Hook

Yêu cầu: Chuyển đổi Class Component sang sử dụng Function Component với useState Hook

Tài liệu về useState Hook: <https://reactjs.org/docs/hooks-state.html>

#37 Giải Bài Tập useState Hook

Đối với React Hook:

- **Sử dụng Arrow Function** (Function component)
- **Không sử dụng keyword this và hàm setState**
- **Mỗi 1 state Hook sẽ là 1 biến độc lập**, như vậy, sẽ chia nhỏ các state của component ra, **ngược hoàn toàn với React Class** (biến state là object chứa toàn bộ state của component)

#38 useEffect Hook - Sử Dụng LifeCycle với React Function Component

Đối với Function component, code sẽ chạy theo thứ tự từ trên xuống dưới.

Sử dụng useEffect Hook như sau:

useEffect (function_xử_lý, các_biến_phụ_thuộc)

Hàm useEffect nhận vào 2 tham số, tham số đầu tiên là function sẽ khởi tạo khi chúng ta chạy hàm useEffect, và tham số thứ 2, là các biến chúng ta muốn nhờ useEffect 'quan sát' sự thay đổi của nó.

Lưu ý 1:

```
useEffect(  
  () => { //code logic here },  
  []  
);
```

Khi chúng ta truyền vào mảng rỗng (tham số thứ 2 của hàm useEffect), thì hàm **useEffect sẽ chạy đúng 1 lần**, sẽ có **tác dụng như hàm componentDidMount** của React Class.

Lưu ý 2:

```
useEffect(  
  () => { //code logic here },  
  [ tên_biến ]  
);
```

Khi truyền vào tên biến (đối với tham số thứ 2), thì hàm **useEffect sẽ được chạy 'bất cứ khi nào giá trị của 'biến quan sát' bị thay đổi**.

Như vậy, lúc này, **hàm useEffect sẽ có tác dụng như hàm componentDidUpdate**, giúp chúng ta có thể so sánh giá trị quá khứ và giá trị hiện tại của biến (props/state).

Nếu giá trị của biến thay đổi, khối code logic của hàm useEffect sẽ được chạy.

#39 Why Hook ? Tại Sao Chúng Ta Sử Dụng React với Hook

React Hook ra đời vào năm 2018, giúp Function Component có đầy đủ tính năng như Class Component

Hook = Cái móc câu. Nếu như Function Component là 'con cá', thì Hook sẽ 'móc vào' Component, giúp component có thêm cái gì đấy (có thêm state :v)

React Hook chỉ có thể sử dụng với React version ≥ 16.8

React Class tồn tại vấn đề sau:

- **Viết code theo OOP** (lập trình hướng đối tượng). Cần hàm tạo (constructor), khai báo state, khai báo lifecycle...
- **Sử dụng keyword this**
- Với 1 component có nhiều giá trị state, **việc tách riêng 'một vài state' để tái sử dụng là điều không thể.**

React Hook giúp giải quyết các vấn đề trên, đồng thời, giúp component của React, chỉ đơn giản là Function.

Chúng ta dùng Hook, thay vì dùng Class Component vì:

- Hook ra đời sau Class, và đang là xu hướng. Facebook chuyển dịch thì chúng ta chuyển dịch theo.
- Các dự án mới đa phần phát triển với Hook, vì đây là xu hướng chuyển dịch.

Tài liệu tham khảo về React Hook: <https://reactjs.org/docs/hooks-intro.html>

TÀI LIỆU FULL SẼ ĐƯỢC CẤP
KHI CÁC BẠN MUA KHÓA HỌC :)