TÀI LIỆU KHÓA HỌC “React Ultimate”

Tác giả: Hỏi Dân IT & Eric

Version: 9.0

Note cập nhật:

- Thêm chapter 10, 11, 12, 13 (hiểu React một cách toàn diện)
- Update backend với validate (hiển thị message tiếng việt)

Chapter 0: Giới Thiệu

Giới thiệu và demo kết quả đạt được sau khi kết thúc khóa học này.

#0.1 Demo Kết quả đạt được

Video demo kết quả đạt được: https://www.youtube.com/watch?v=jUOwicA-IQ0

#0.2 Tài liệu khóa học

Đối với mỗi khóa học của Hỏi Dân IT, **học viên đều được cung cấp file tài liệu:**

- Định dạng pdf, được dùng để 'take note' kiến thức quan trọng, link source code, link ví dụ...
- Được đánh số version, link download trực tiếp trong khóa học

Lưu ý về khóa học này:

1. Source code Frontend

Source code Frontend cả khóa học "không được" cung cấp. Bạn cần xem video và code theo.

Cuối mỗi video đều có phần review các files thay đổi.

Chỉ có những video nào, mình nói cho source code (thông thường là các video khó), mình sẽ để link download source code trong tài liệu pdf.

Lý do: 99% những bạn xin full source code, thông thường sẽ không xem video => review và đánh giá khóa học kém chất lượng.

100% code theo videos khóa học sẽ thành công, vì nếu có lỗi, chỉ cần bạn report, mình sẽ làm video update.

2. Source code Backend

- Source code backend bạn sẽ được cung cấp sẵn, dưới dạng mã hóa. Chỉ chạy và không sửa đổi.

Lý do: Đây là khóa học frontend, nên sẽ không học backend. Bạn muốn biết source code này viết như nào, vui lòng học lộ trình backend

3. Chuyện deploy

- Khóa học này chạy localhost, vì hosting free ngoài kia rất "lởm". hàng free mà.

- Deploy chuyên nghiệp cần biết cả frontend và backend
- Trong cv, bạn ghi link github là đủ. nếu cần thiết, thì quay video demo, thay vì deploy free

- Muốn deploy thực tế, bạn cần học kỹ năng backend, sau đấy deploy với server trả phí mình đã có khóa học deploy chuyên làm phần này: https://hoidanit.com.vn/khoa-hoc/ultimate-guide-to-deploy-react-nodejs-640bee82f7099c369b3bc6a4.html

#0.3. Về tác giả

Mọi thông tin về Tác giả Hỏi Dân IT, các bạn có thể tìm kiếm tại đây:

Website chính thức: https://hoidanit.com.vn/

Youtube “Hỏi Dân IT” : https://www.youtube.com/@hoidanit

Tiktok “Hỏi Dân IT” : https://www.tiktok.com/@hoidanit

Fanpage “Hỏi Dân IT” : https://www.facebook.com/askITwithERIC/

Udemy Hỏi Dân IT: https://www.udemy.com/user/eric-7039/

Về chuyện leak khóa học và mua lậu

Mình biết rất nhiều bạn khi học khóa học này của mình, là mua lậu qua bên thứ 3. chuyện này là hoàn toàn bình thường, vì thương hiệu "Hỏi Dân IT" đang ngày càng khẳng định được vị thế của mình.
Nhiều bạn hỏi mình, sao mình không 'chặn việc mua lậu'. nói thật, nếu mình làm, là làm được đấy, cơ mà nó sẽ gây ra sự bất tiện cho học viên chân chính (con sâu làm rầu nồi canh). Với lại, ngay cả hệ điều hành windows, còn bị crack nữa là @@

Mình cũng có 1 bài post facebook về chuyện này:
https://www.facebook.com/askitwitheric/posts/pfbid02gyasktd3semgxat6nevnvwh4c8epzu3i7kpzhr7s7gmmfcvucyz96eb8avnvgwnhl
Với các bạn học viên chân chính, mình tin rằng, những cái các bạn nhận được từ mình khi đã chấp nhận đầu tư, nó sẽ hoàn toàn xứng đáng. vì đơn giản, với cá nhân mình, khách hàng là thượng đế.

VỚI CÁC BẠN MUA LẬU, MÌNH CHỈ MUỐN CHIA SẺ THẾ NÀY:

1. **TRÊN ĐỜI NÀY, CHẲNG CÓ GÌ CHẤT LƯỢNG MÀ MIỄN PHÍ CẢ.**
   VIỆC BẠN MUA LẬU QUA BÊN THỨ 3, LÀ GIÚP BỌN CHÚNG LÀM GIÀU VÀ GÂY THIỆT HẠI CHO TÁC GIẢ.
   NẾU NHÌN VỀ TƯƠNG LAI => CÀNG NGÀY CÀNG ÍT TÁC GIẢ LÀM KHÓA HỌC => NGƯỜI BỊ HẠI CUỐI CÙNG VẪN LÀ HỌC VIÊN

2. **HÃY HỌC THÓI QUEN TRÂN TRỌNG GIÁ TRỊ LAO ĐỘNG**
   NÓ LÀ THÓI QUEN, CŨNG NHƯ SẼ LÀ MỘT PHẦN TÍNH CÁCH CỦA BẠN.
   ĐỪNG VÌ NGHÈO QUÁ MÀ LÀM MẤT ĐI TÍNH CÁCH CỦA BẢN THÂN.
   NẾU KHÓ KHĂN, CỨ INBOX MÌNH, MÌNH HỖ TRỢ. VIỆC GÌ PHẢI LÀM VẬY =))

3. **MÌNH ĐÃ TỪNG LÀ SINH VIÊN GIỐNG BẠN**, MÌNH HIỂU TẠI SAO CÁC BẠN LÀM VẬY. **HÃY BIẾT CHO ĐI. SỐNG ÍCH KỶ, THÌ THEO LUẬT NHÂN QUẢ ĐẤY, CHẲNG CÓ GÌ LÀ NGẪU NHIÊN CẢ**

4. **NẾU BẠN THẤY KHÓA HỌC HAY, HÃY BIẾT DONATE ĐỂ ỦNG HỘ TÁC GIẢ. LINK DONATE: https://hoidanit.com.vn/donate**

Hành động nhỏ nhưng mang ý nghĩa lớn. Hãy vì 1 cộng đồng IT Việt Nam phát triển. Nếu làm như các bạn, có lẽ chúng ta đã không có Iphone, không có Apple như ngày nay rồi @@

Chapter 1: React và Javascript ES6
Ôn tập các kiến thức Javascript hay dùng với React.JS

#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.

```
const 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 $p->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”;
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

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 <= 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 - Giản 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.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;
```

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

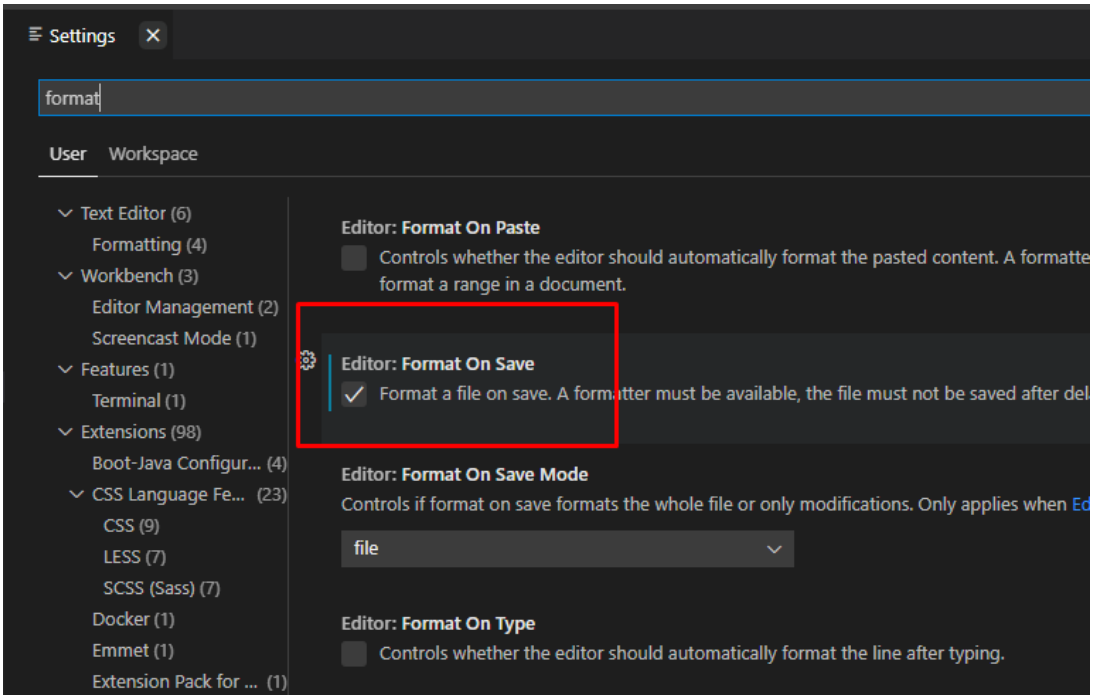
Nắm vững các kiến thức cơ bản và cốt lõi của React

#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

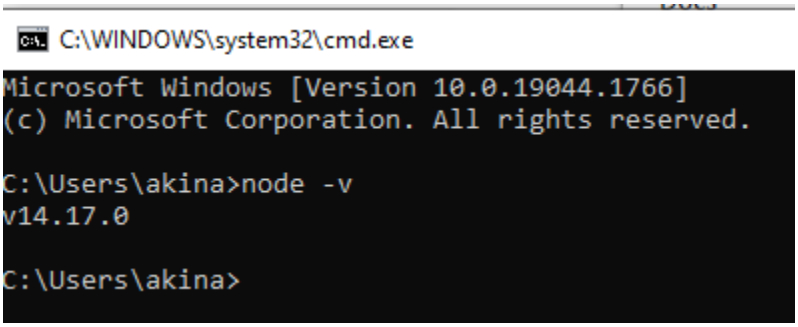
- Yêu cầu: Cài đặt thành công Git tại máy tính cá nhân, có tài khoản Github, biết đẩy code của mình lên Github (xem hết 5 video đầu tiên của link trên)

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

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



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=ccjKHLyo4lM

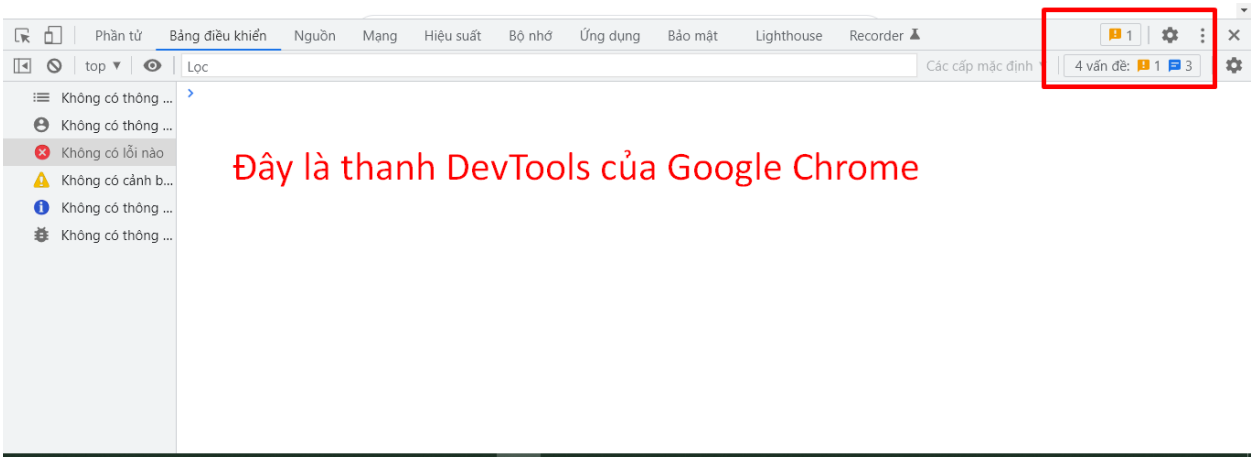
Lưu ý: Chỉ cài đặt nhiều version của Node.Js khi máy tính của bạn chạy nhiều dự án Javascript với các version của Node.JS khác nhau. Nếu như đây là lần đầu tiên bạn làm quen và sử dụng Node.js, thì không cần cài đặt NVM, chỉ cần cài đặt thành công 1 version của Node.JS là được

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

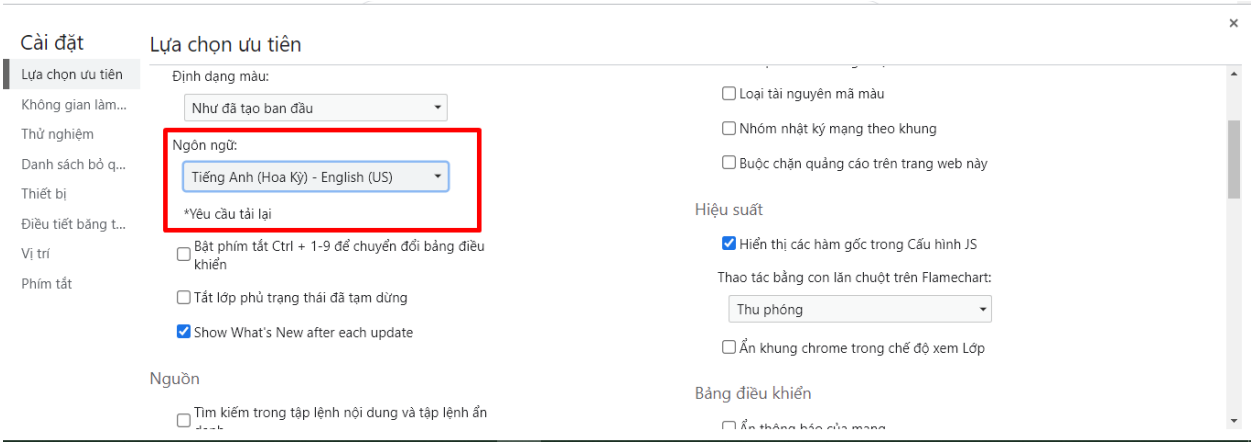
4.Cài Đặt Google Chrome

- Chuyển ngôn ngữ Chrome sang tiếng anh: https://www.youtube.com/watch?v=GVwefuqLniM&list=PLncHg6Kn2JT6E38Z3kit9Hnif1xC_9VqI&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 đớ", và khi các bạn search google nó không ra đâu @@

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

#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_Q40NVP95YIPIhtW6JqnXjkHQZ18Bz/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

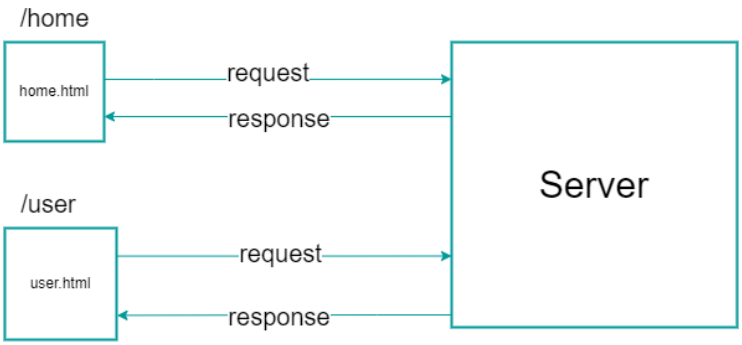
- index.html: file quan trọng nhất của React. Tất cả mã nguồn ứng dụng, sẽ được webpack bundle vào file này (bao gồm file index.js ở thư mục src). Ứng dụng React bản chất chỉ chạy mình file này.
- favicon.ico: file logo của website
- manifest.json: cung cấp thông tin của ứng dụng, như tên, hình ảnh… được sử dụng chủ yếu để hiển thị trên mobile
- robotstxt: phục vụ mục đích SEO, crawl data của Google bot

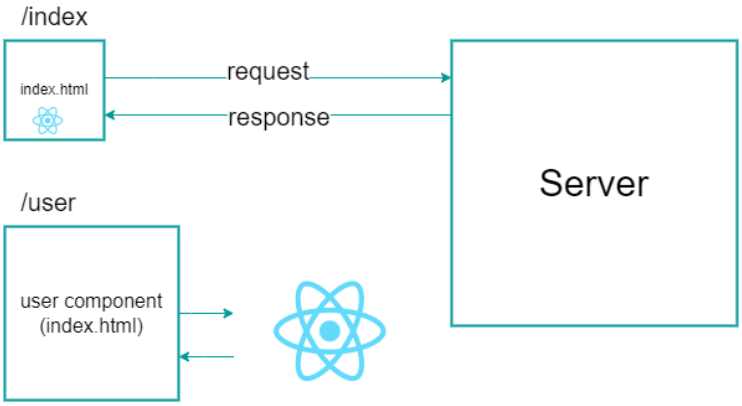
#15 How React Works ?

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 đấy, 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 (thự 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/lmhkpmbekcpmknklioeibfkpmmfibljd?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 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

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 - 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;

#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 toản 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 backend

Ở 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 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

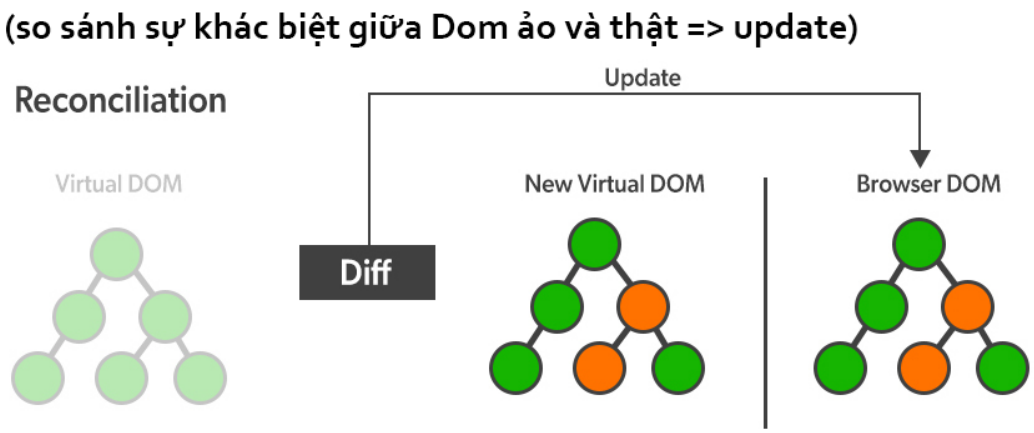
- Hiểu đơn giản về DOM (Document Object Model) - chính là cây HTML giúp hiển thị lên giao diện website

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
Sử dụng React hiện đại 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 dưng 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 đấy 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 - Giản 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

Chapter 4.1: Hướng Dẫn Cài Đặt Backend

Giới thiệu mô hình hoạt động của dự án và cách setup backend

#40.1 Mô Hình Hoạt Động Của Dự Án

Mô hình hoạt động của dự án gồm 3 thành phần:

1. Frontend React (chúng ta sẽ cùng nhau code phần này)

2. Backend (viết bằng javascript với Node.JS)
3. Database SQL

Thành phần 2 và 3 sẽ được mình cung cấp, có nghĩa là, với dự án Frontend, chúng ta sẽ sử dụng backend được cung cấp sẵn (mà không cần học backend)

#40.2 Tổng quan về dự án thực hành

Ý tưởng dự án: Bài test Fresher của FSoft (FTP Software)

```
1. Support user registration function, login is required to perform the survey.
2. Check role for admin, user (response from BE)
3. Use Token Based Authentication with access token, refresh token.
4. Use Ant Design for UI
5. Once logged in, if role is user, the first page displays the first question and the total number of questions to be
answered. If role is admin, the first page displays the question management screen.

******************************USER display******************************
1. User can choose number of questionss, that they want to play
2. Click save and next to save the results and go to the next page, back to return to the previous
question (Optional: Support skip question).
3. Questions can choose only 1 answers.
4. Show a list of selected questions and answers along with the correct answer and total score
after completing the survey.

******************************ADMIN display******************************
1. Can Add, delete, overview any question on the question management screen
2. Click to specific question, which admin want to view detail
3. At (2), admin can edit or delete specific question
```

Cụ thể yêu cầu của dự án chúng ta sẽ thực hành:

- Chức năng đăng ký/đăng nhập người dùng
- Người dùng sẽ được phân quyền dựa vào role (Admin/user)
- Sử dụng JWT (Json web token) để xác thực người dùng
- CRUD bài thi, câu hỏi, câu trả lời, users
- Chức năng làm bài thi, chấm điểm.

#40.3 Lưu Ý về Cài Đặt Backend

Để cài đặt backend cho dự án thực hành, chúng ta có 2 cách:

Cách 1: cài đặt thông thường, gồm có cài đặt backend Node.JS và database SQL

Cách 2: cài đặt thông qua Docker.

Mình khuyến khích các bạn sử dụng cách 2, vì nếu biết Docker, nó sẽ là một công cụ hữu ích phục vụ cho việc đi làm sau này.

Nếu bạn thực hiện Cách 1:
chỉ cần xem Chapter 4.2: Setup Dự Án Backend (Không Dùng Docker)
=> bỏ qua chapter 4.3

Nếu bạn thực hiện Cách 2:
Chỉ cần xem Chapter 4.3: Setup Dự Án Backend (Dùng với Docker)
=> bỏ qua chapter 4.2

Mình chia ra làm 2 cách như vậy, bởi vì một vài bạn không setup được docker =))

p/s: muốn học sâu với mình, ví dụ như học fullstack, thì Docker là bắt buộc các bạn nhé. Ví dụ như website https://hoidanit.com.vn/ là mình triển khai với Docker đấy :v

Chapter 4.2: Setup Dự Án Backend (Không Dùng Docker)
Hướng dẫn cài đặt backend không sử dụng Docker

#40.4 Các Cách Cài Đặt MySQL

Có 2 cách để cài đặt MySQL nhanh nhất (chỉ cần sử dụng 1 trong 2 cách sau):

Cách 1: Cài đặt phần mềm MySQL Workbench

Cách 2: Cài đặt phần mềm XAMPP

Trong quá trình thực hiện khóa học, chỉ cần cài 1 trong 2 phần mềm trên, không cần cài cả hai.

Mình hướng dẫn 2 phần mềm, vì đôi khi có trường hợp ngoại lệ, nhiều bạn không dùng được cách 1, thì chúng ta sẽ sử dụng cách 2 :v

#40.5 Cài Đặt MySQL Workbench

Link tài liệu hướng dẫn:

- Windows: https://dev.mysql.com/doc/refman/5.7/en/windows-installation.html
- MacOS: https://dev.mysql.com/doc/refman/5.7/en/macos-installation.html

Link tải file cài đặt:
https://dev.mysql.com/downloads/installer/

Download file cài đặt sử dụng trong video (windows):
https://drive.google.com/file/d/1bMZv0y07KVRtBuDljpgNeEXmpiDA7E3V/view?usp=sharing

#40.6 Cài Đặt XAMPP

Lưu ý: Nếu bạn đã cài đặt thành công MySQL Workbench, thì không cần cài đặt phần mềm này.

Chỉ xem và làm theo video này, khi và chỉ khi, bạn không thể cài đặt MySQL Workbench

Link download (dành cho windows) file cài đặt trong video:
https://drive.google.com/file/d/1bMZv0y07KVRtBuDljpgNeEXmpiDA7E3V/view?usp=sharing

Link download cho các hệ điều hành:
https://sourceforge.net/projects/xampp/files/

#40.7 Cài Đặt Dự Án Backend (Không Dùng Docker)

Download file cài đặt:
https://drive.google.com/drive/folders/1-WoA-gWaoVN5qGrrrOs7-yL2njC7gkse?usp=sharing

#40.8 Test API với PostMan

Download Postman: https://www.postman.com/downloads/

Lưu ý: File collection tải tại video #40.7 (đã bao gồm trong source code backend)

#40.9 Test Database với DBeaver (MySQL)

Download DBeaver: https://dbeaver.io/download/

Lưu ý: DBeaver không phải là database, nó là công cụ giúp chúng ta xem được data lưu bên trong database

Chapter 4.3: Setup Dự Án Backend (Dùng với Docker)

Hướng dẫn cài đặt backend bằng cách sử dụng Docker (recommend)

#41.1 Về Cách Học Docker Cho Beginners

Học nhanh docker dành cho beginners tại đây:
https://www.youtube.com/playlist?list=PLncHg6Kn2JT4EcU8eTdmojGO1StFDILe2

#41.2 Tạo tài khoản Docker Hub

Docker giúp chúng ta 'đóng gói ứng dụng', tạo môi trường để chạy project, và hạn chế tối đa việc phải cài đặt công cụ.

Docker Hub là nơi lưu trữ file chạy của Docker (images), tương tự như Github để lưu code.

Tìm hiểu về Docker cơ bản, các bạn có thể xem tại đây:
https://www.youtube.com/playlist?list=PLncHg6Kn2JT4kLKJ_7uy0x4AdNrCHbe0n

Cách tạo tài khoản trên Docker Hub:

1. Truy cập https://hub.docker.com/signup để tạo tài khoản
2. Docker Id chính là tên username dùng để hiển thị. Sử dụng Gmail để đăng ký
3. Khi xác nhận Email, nếu không thấy Email xác nhận ở Inbox (Hộp thư đến), các bạn nhớ check trong thư mục Spam nhé ^^

#42 Cài đặt Docker Desktop

Video hướng dẫn cài đặt: https://www.youtube.com/watch?v=Ic48-WLhtHg&list=PLncHg6Kn2JT4kLKJ_7uy0x4AdNrCHbe0n&index=2

Download Docker Desktop và VM Linux mình sử dụng trong video tại đây:
https://drive.google.com/drive/folders/1i2tzQRlTZ7shibrAXzvWAfhoSatvqxH8?usp=sharing

#43. Hướng Dẫn Cài Đặt Dự Án Backend (Dùng Docker)

Các bạn vui lòng xem video và làm theo hướng dẫn để cài đặt backend cho dự án nhé.

Download file cài đặt Docker:
https://drive.google.com/file/d/1a92o_EJa1h5Ciz99lK1borpPqz-XStvL/view?usp=sharing

#44 DBeaver - Kết Nối Database Postgres

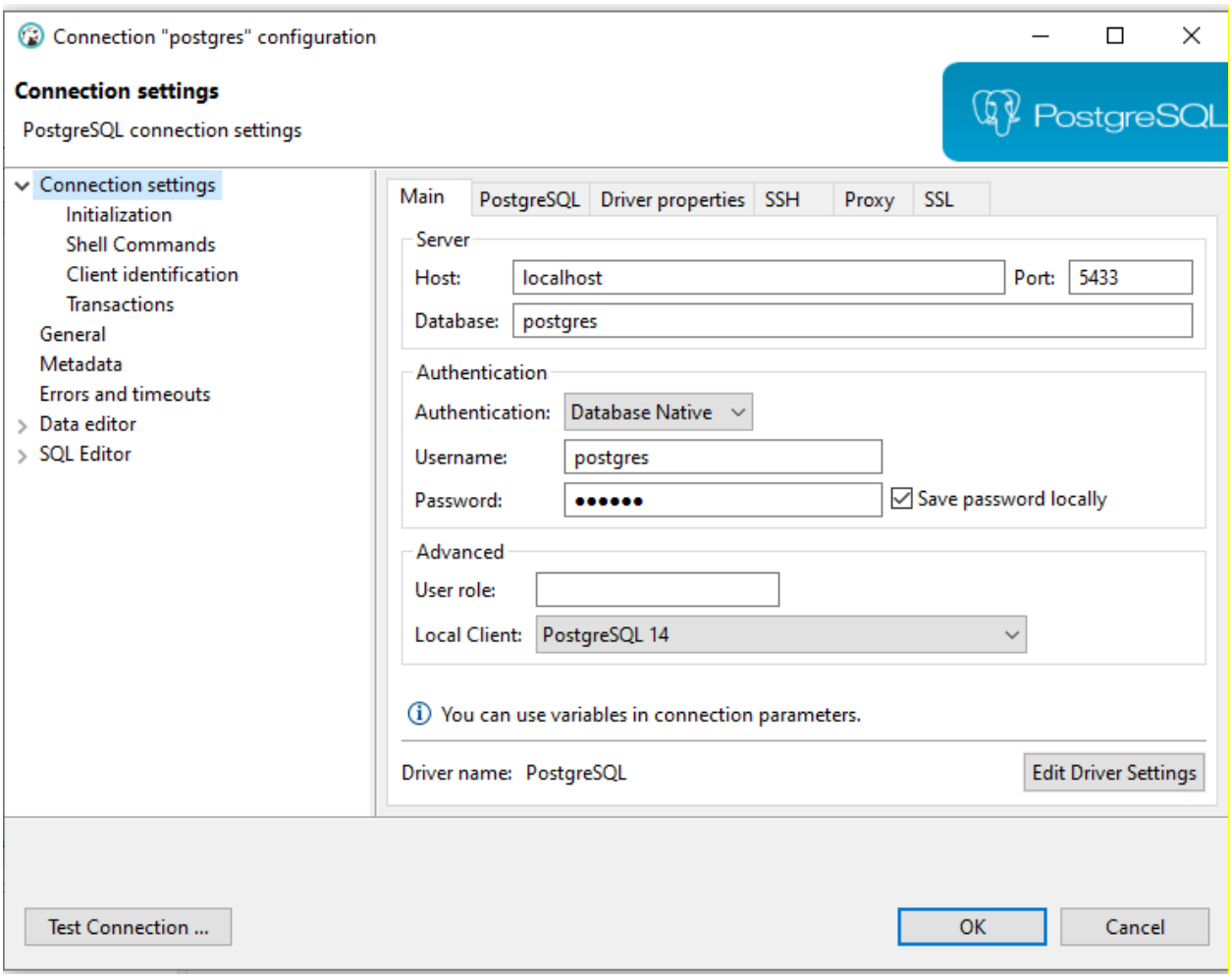
Lưu ý: hình ảnh bên dưới chỉ mang tính chất mình họa. Trong video #43, mình có đề cập cách xem thông tin đăng nhập database với DBeaver

1.Download DBeaver: (dành cho Windows)
https://drive.google.com/file/d/1Tf1wIf-G6gdVu5wWMze1ULNJbht-JlJF/view?usp=sharing

Với mac, linux: https://dbeaver.io/download/

2.Kết nối Database Postgres:

Các tham số mặc định của database, khai báo tại file docker-compose khi cài đặt với Docker.
Mặc định:
Port: 5433
Database: postgres
Username: root
Password: 123456
Lưu ý: trong lần đầu tiên sử dụng, có thể cần download Driver, các bạn cứ nhấn 'Yes/Ok' (nếu có) để phần mềm tự động download nhé


Connection "postgres" configuration
Connection settings
PostgreSQL connection settings
PostgreSQL
Connection settings
Initialization
Shell Commands
Client identification
Transactions
General
Metadata
Errors and timeouts
Data editor
SQL Editor
Main
PostgreSQL
Driver properties
SSH
Proxy
SSL
Server
Host: localhost
Port: 5433
Database: postgres
Authentication
Authentication: Database Native
Username: postgres
Password: ••••••
Save password locally
Advanced
User role:
Local Client: PostgreSQL 14
You can use variables in connection parameters.
Driver name: PostgreSQL
Edit Driver Settings
Test Connection ...
OK
Cancel

3.Phân tích Models Database:

Models:

- Participant: quản lý users sử dụng hệ thống
- Quiz: quản lý các bài thi
- Question: quản lý câu hỏi của bài Quiz
- Answer: quản lý câu trả lời cho từng câu hỏi (Question)

#45 PostMan - Test APIs Backend

Download Postman: https://www.postman.com/downloads/

Link download for Windows: https://drive.google.com/file/d/1MTndb2SdL9J73TFc-hSYXF89Xm8yNmzK/view?usp=sharing

Link download Collections để import vào Postman, đã có sẵn khi tải file setup docker :v

Chapter 5: Điều Hướng Trang với React Router v6

Điều hướng trang trong ứng dụng React sử dụng React-router

#46 Setup Bootstrap 5 & React Router Dom v6

1.Cài đặt React Router Dom :

npm install --save-exact react-router-dom@6.3.0

Tài liệu chi tiết: https://reactrouter.com/

2.Cài đặt Bootstrap 5 sử dụng với React

npm install --save-exact react-bootstrap@2.4.0 bootstrap@5.2.0

Sau khi cài đặt thư viện, cần import CSS của Bootstrap vào file index.js
import 'bootstrap/dist/css/bootstrap.min.css';

Tài liệu chi tiết: https://react-bootstrap.github.io/

3. Tại sao lại cài đặt thư viện react-bootstrap để dùng bootstrap, mà không đơn giản chỉ cần cài đặt mình thư viện bootstrap ?

- Nếu chỉ cài mình thư viện Bootstrap, thì với các Component cần hiệu ứng, ví dụ như Modal, Dropdown… sẽ không chạy được, vì phần hiệu ứng do JQuery và Javascript phụ trách
- Thư viện React-Bootstrap giúp giải quyết vấn đề trên, đồng thời, tối ưu code để dùng tốt với React.

#47 Design Header với Bootstrap Navigation

Tài liệu Bootstrap NavBar: https://react-bootstrap.github.io/components/navbar/#overview

#48 Điều Hướng Trang với Links

Tài Liệu: https://reactrouter.com/docs/en/v6/getting-started/tutorial#connect-the-url

#49 Nested Routes

Tài liệu: https://reactrouter.com/docs/en/v6/getting-started/tutorial#nested-routes

Để share 'dữ liệu HTML' (phần dùng chung) giữa các Routes, chúng ta sử dụng component Outlet.

Outlet cần được định nghĩa ở component Cha. Từ đấy, khi render giao diện giữa các route con, phần component con sẽ được render vào phần Outlet đã định nghĩa.

#50 Active Link - NavLink

Tài liệu: https://reactrouter.com/docs/en/v6/getting-started/tutorial#active-links

Component NavLink khác gì Link ?

NavLink có chức năng 'giống hệt' Link (giúp điều hướng trang), tuy nhiên, cung cấp thêm công cụ để chúng ta CSS cho 'active link' - link chúng ta 'đã click trước đó'.

Hiểu đơn giản: NavLink = Link + CSS

Mặc định, khi user click vào 1 NavLink, thì ngay lập tức NavLink đó sẽ được 'add - thêm vào' class có tên là 'active'

#51 Index Routes

Tài liệu: https://reactrouter.com/docs/en/v6/getting-started/tutorial#index-routes

Index routes không có thuộc tính 'path', vì đơn giản, nó dùng lại path của parent. Thay vào đó, nó cần props 'index'

```
<Route
    index
    element={
      <main style={{ padding: "1rem" }}>
        <p>Select an invoice</p>
      </main>
    }
/>
```

Index routes sẽ được gọi tới, khi tất cả 'route con' còn lại không match, thì nó sẽ được dùng.

#52 Design Homepage

1.Nguồn tài liệu tham khảo

Link website clone: https://www.typeform.com/
Link download video:
https://drive.google.com/file/d/1APqoC_9VKHnHqmIEwImkqjedgKv-S1Bs/view?usp=sharing

2.Lưu ý về sử dụng Video với React

- Import video vào Component như cách sử dụng với image
- Sử dụng html tag video theo cú pháp:

```
<video autoPlay muted loop>
        <source src= '... '  type = '...'/>
 </video>
```

Trong đó:
loop (vòng lặp) : giúp video chạy xong, nó sẽ tự động chạy lại (lặp đi lặp lại, vô hạn).

autoPlay(tự động chạy) : giúp video tự động phát khi load vào website. Ở đây, video sẽ được phát mà không cần nhấn nút 'run'.
muted (câm) : giúp tắt tiếng (audio) của video. Bắt buộc phải sử dụng thuộc tính này kèm với autoPlay để cho tính năng autoPlay có thể hoạt động.

3.Lý do phải dùng song song muted + autoPlay là vì:

Ngày xưa (trước 2018), các website thường tự động chạy video (có âm thanh) và tự động chạy file nhạc khi người dùng vào website, đặc biệt là các website quảng cáo.

Như vậy, tạo sự khó chịu cho người dùng khi không muốn xem/nghe file mà vẫn bị bắt phải làm.

Vì vậy, sau tháng 4/2018, với các version mới của Google Chrome, tính năng autoPlay file nhạc (audio) bị bỏ. Tương tự với video có audio (video không có tiếng vẫn dùng được autoPlay).

Đấy chính là lý do tại sao cần dùng autoPlay + muted.

Nếu muốn chạy file nhạc (audio), hoặc video có tiếng, bắt buộc người dùng phải nhấn nút 'run/play' để thực hiện, không thể dùng javascript/html để tự động chạy 2 loại file trên.

Chi tiết xem tại: https://developer.chrome.com/blog/autoplay/

#53 Design New Header

Tạo Header với background transparent (trong suốt):
background-color: rgba(0, 0 , 0, 0)

Về rgba và rgb xem tại đây: https://www.w3schools.com/css/css_colors_rgb.asp

#53.1 Kinh Nghiệm Đọc Code Quá Khứ - Fix Lỗi Khi Thư Viện Update

Lưu ý: trong quá trình xem video thực hành khóa học, các bạn vui lòng cài đặt đúng 'version thư viện' mình hướng dẫn (cho dù thư viện nó có update)

Lý do: là để hạn chế lỗi thôi. Chứ chạy theo công nghệ, lắp tên lửa vào người bay theo cũng không kịp.

Còn sau này, khi đã 'cứng rồi', các bạn có khả năng 'tự fix bug' thì làm gì cũng được nhé. Thích gì thì quất vậy.

Link github thực hành: https://github.com/azouaoui-med/react-pro-sidebar

1.Câu lệnh git checkout

https://git-scm.com/docs/git-checkout

Check code tại 1 branch (đã có sẵn)
git checkout branch_name

Check code sang 1 branch hoàn toàn mới (chưa có)
git checkout -b branch_name

Check code tại 1 commit (1 mốc thời gian trong quá khứ)
git checkout commit_hash

#54 Design Admin SideBar

Tích hợp thư viện
npm install --save-exact react-pro-sidebar@0.7.1

Nếu cài đặt bị lỗi, thử câu lệnh này:
npm install --save-exact react-pro-sidebar@0.7.1 --legacy-peer-deps

Link github: https://github.com/azouaoui-med/react-pro-sidebar
Link demo (preview): https://azouaoui-med.github.io/react-pro-sidebar/

Để sử dụng icon, cài đặt thêm thư viện sau:
npm install --save-exact react-icons@4.4.0

Link source code video này (phần sidebar):
https://drive.google.com/file/d/1jw2Ie9poPA24BMNf0_nqP_GK6MBmM5nh/view?usp=share_link

Link file github mình copy để làm Sidebar: **(lưu ý, xem video #53.1 để biết cách fix lỗi - nên xem để biết)**
https://github.com/azouaoui-med/react-pro-sidebar/blob/master/demo/src/Aside.js

Link download file ảnh background sidebar:
https://drive.google.com/file/d/1GllVesIPO2cLVtvobRalEhJHYEZAlfPa/view?usp=sharing

#55 Setup Axios, React Toastify, React Paginate

Tài liệu về React-Icons:
Github: https://github.com/react-icons/react-icons

Trang chủ: https://react-icons.github.io/react-icons/

Download Sidebar (code của mình):
https://drive.google.com/file/d/1JkqrgmBERd8s9E44eUYYsymtUQK3Ld1D/view?usp=sharing

Cài đặt thư viện cần thiết:

npm install --save-exact axios@0.27.2 react-toastify@9.0.7 react-paginate@8.1.3

Chapter 6: CRUD Users - Thêm/Hiển Thị/Cập Nhật/Xóa Người Dùng

Luyện tập chức năng thêm, sửa, xóa người dùng với React và APIs của Backend

#56 Modal Thêm Mới Người Dùng

Tài liệu:

1. Modal Reactrap: https://react-bootstrap.github.io/components/modal/#live-demo

2. Form Bootstrap: https://getbootstrap.com/docs/5.0/forms/layout/#gutters

#57 State Hóa Modal Add New User

1.Lưu ý khi CSS cho Modal:

Mặc định, Modal không được render vào div 'root', mà thường chèn vào trước tag </body>

=> khi CSS, nếu CSS theo kiểu cha lồng con của app React, nó sẽ không hoạt động.
=> cách dễ nhất để fix bugs trên là đặt className cho Modal, sau đấy CSS global.

Phải đặt className, tránh việc CSS trực tiếp vì nó có thể làm vỡ giao diện, do bị trùng (ghi đè) CSS của nhau.

2. CSS Image Fit Div: https://stackoverflow.com/a/3029434

3. Customize Upload File button

Sử dụng:
<label htmlFor="your-input-id" > </label>
<input hidden id="your-input-id" />

4.Upload File với React

Để lấy file thông qua event, sử dụng event.target.files[0]

Ở đây, khi <input type='file'/>, sử dụng event.target sẽ trả ra FileList (một array chứa File), thành ra khi upload 1 file (single), chúng ta cần lấy phần tử đầu tiên của FileList (files[0])

Lưu ý: File là 1 loại dữ liệu đặc biệt của javascript

Tài liệu về File với javascript: https://developer.mozilla.org/en-US/docs/Web/API/File_API/Using_files_from_web_applications
5.Preview Image với React

File ảnh upload được lấy dưới dạng File object => không thể hiển thị hình ảnh dưới dạng File object, cần phải sử dụng URL.createObjectURL( )

Tài liệu:
https://developer.mozilla.org/en-US/docs/Web/API/URL/createObjectURL

#58 API Thêm Mới Người Dùng (CREATE)

API thêm mới user:

POST http://localhost:8081/api/v1/participant

Tài liệu Axios với Form Data: https://github.com/axios/axios#using-multipartform-data-format

#59 Validate data và React Toastify

Validate Email: https://stackoverflow.com/a/46181

Tài liệu React Toastify: https://fkhadra.github.io/react-toastify/introduction/

```
toast.info("Lorem ipsum dolor"); // same as toast(message, {type: "info"});
toast.error("Lorem ipsum dolor")
toast.success("Lorem ipsum dolor")
toast.success("Lorem ipsum dolor", {
  theme: "colored"
})
toast.warn("Lorem ipsum dolor")
toast.warn("Lorem ipsum dolor", {
  theme: "dark"
})
```

#60 API Services - Customize Axios

1.Về cách đặt tên thư mục:

- Thông thường, trong các dự án, thư mục Services thường dùng để ám chỉ các logic liên quan tới gọi APIs. Nên tách riêng phần này khỏi component để dễ quản lý code.
- Thư mục Utils (là viết tắt của utility - hữu ích), nơi chứa những function dùng chung (helper)

2.Axios Customize

- Tạo Instance Axios: https://github.com/axios/axios#creating-an-instance

- Tạo customize của Axios, để trước khi gọi API, chúng ta sẽ chỉnh sửa thư viện axios cho phù hợp với nhu cầu. Hiểu nôm na, những phần nào dùng chung giữa các lần gọi API với Axios, chúng ta sẽ 'cấu hình' tại 1 nơi duy nhất.

  Ví dụ: mỗi lần gọi API, chúng ta cần truyền access_token vào APIs. Thay vì làm thủ công, là khi nào gọi APIs thì thêm trường access_token, chúng ta sẽ cấu hình ở file axios_customize, và thêm access_token ở đó (1 nơi duy nhất).
  Như vậy, mặc định, chúng ta sẽ không cần truyền access_token khi gọi APIs nữa.
  (trong khóa học, chúng ta sẽ thực hành ví dụ trên)

- Instance, hiểu đơn giản là 1 object (thực thể/đối tượng). Giống như khi học lập trình hướng đối tượng, chúng ta có:

  const car = new Car (...) thì ở đây là const instance = axios.create(...)

- Tạo Interceptor: https://github.com/axios/axios#interceptors
  You can intercept requests or responses before they are handled by then or catch. (chúng ta có thể can thiệp vào request và response trước khi client nhận được phản hồi)

  Mã phản hồi của backend (http status code) : https://developer.mozilla.org/en-US/docs/Web/HTTP/Status

  Interceptor (người can thiệp) được sử dụng để customize request/response.

Trước khi request (từ client gửi lên server), Interceptor có thể can thiệp (ví dụ như việc thêm acceess_token ở trên)
Trước khi response (từ server gửi về client), Interceptor cũng có thể can thiệp.
Mặc định, khi sử dụng axios, axios sẽ trả về response theo dạng sau:
https://github.com/axios/axios#response-schema

=> response có nhiều hơn những cái chúng ta quan tâm => dùng interceptor để can thiệp và lấy cái chúng ta muốn

The response for a request contains the following information.

```
{
  // `data` is the response that was provided by the server
  data: {},

  // `status` is the HTTP status code from the server response
  status: 200,

  // `statusText` is the HTTP status message from the server response
  statusText: 'OK',

  // `headers` the HTTP headers that the server responded with
  // All header names are lowercase and can be accessed using the bracket notation.
  // Example: `response.headers['content-type']`
  headers: {},

  // `config` is the config that was provided to `axios` for the request
  config: {},

  // `request` is the request that generated this response
  // It is the last ClientRequest instance in node.js (in redirects)
  // and an XMLHttpRequest instance in the browser
  request: {}
}
```

#61 Hiển Thị Danh Sách Users (READ)

1.APIs lấy danh sách người dùng

GET http://localhost:8081/api/v1/participant/all

2.Tài liệu

- Về cách sử dụng hàm map ( ), xem tại video #5 ES6 Array Methods - Map và Filter

- Table Bootstrap 5: https://getbootstrap.com/docs/5.0/content/tables/

- Button Bootstrap 5: https://getbootstrap.com/docs/5.0/components/buttons/

#62 Cập Nhật Danh Sách Users Khi Thêm Mới Thành Công

1.Lift-up state

Tài liệu: https://reactjs.org/docs/lifting-state-up.html

Lift-up (nâng lên), ở đây chúng ta hay gọi là 'lifting state up', một kỹ thuật hay được dùng để chia sẻ data trong ứng dụng React.

Cụ thể, chúng ta sẽ đưa 'state của component con' sang cho 'component cha' quản lý. Component con sẽ dùng data thông qua 'props truyền từ cha xuống'

Như vậy, 'lift-up state' ám chỉ việc đưa state của 'component con' lên 'component cha'. Do React thiết kế theo mô hình cha-con, data chảy từ trên (cha) xuống dưới (con), nên gọi là lift-up (đẩy lên/nâng lên).

Khi nào cần lift-up state ?

Chúng ta cần lift-up state khi 'component cha có nhiều con', và các con của nó có quan hệ với nhau. Ở đây, chúng ta muốn 1 hành động xảy ra ở '1 đứa con này', sẽ cần làm gì đấy 'ở 1 đứa con khác', thành ra, chúng ta sẽ nhờ 'cha của các con' làm (thông qua việc lift-up state). Cha có thể nói chuyện với con thông qua 'props', còn các con không thể nói chuyện với nhau.

Ví dụ:

```
<Parent>
    <Child1>...</Child1>
    <Child2> ...</Child2>
```

```
</Parent>
```

#63 Design Modal Update User

Cài đặt thư viện lodash: **npm install --save-exact lodash@4.17.21**

Hiển thị Image base64: https://stackoverflow.com/a/42399865

Lưu ý về code:

- Component sẽ được re-render (vẽ lại giao diện) khi có sự thay đổi của state hoặc props

- Hàm useEffect của react Hook sẽ bằng các hàm sau của React Class:
  1.ComponentDidMount (component đã được render và chèn vào DOM)
  2.ComponentDidUpdate (có component đã được cập nhật, thông qua sự thay đổi của props/state)
  3.ComponentWillUnmount của React Class

Khi useEffect === ComponentDidMount, chúng ta sẽ dùng nơi đây để gọi API. Lý do chúng ta gọi API nơi này, vì lúc này DOM đã sẵn sàng (đã có HTML), nên việc thao tác với DOM sẽ không có lỗi. (tránh bugs không cần thiết).
Lúc này, chúng ta sẽ dùng cú pháp:
useEffect(( ) => {....}, [ ]); để ý là hàm useEffect chúng ta truyền vào 1 mảng rỗng [ ] , mục đích để cho chạy duy nhất 1 lần.

Khi useEffect === ComponentDidUpdate, chúng ta dùng nơi này để xử lý logic khi có bất kỳ sự thay đổi nào của 'biến quan sát - state/props'.
Cú pháp:
useEffect(( ) => {....}, [biến_quan_sát]); biến quan sát có thể là state của Component hiện tại, hoặc là props truyền từ cha xuống.

Ở đây, mỗi khi 'biến quan sát' được thay đổi giá trị, ngay lập tức hàm useEffect sẽ được chạy, như vậy hàm useEffect lúc này sẽ được chạy nhiều lần (> 1)

#64 API Cập Nhật User (UPDATE)

APIs cập nhật người dùng
PUT http://localhost:8081/api/v1/participant

#65 Bài tập: Chức Năng Xem Chi Tiết User

Yêu cầu:

- Nhấn vào button View, mở Modal Xem Chi Tiết Người Dùng
- Thông tin hiển thị giống hệt như Modal Update User, tuy nhiên, tất cả các trường sẽ disabled (không cho sửa)
- Modal này không cần nút Save, chỉ cần nút Close (mục đích chỉ cho user xem thông tin, không thể thao tác sửa đổi cập nhật)

#66 Design Modal Delete User

Modal Bootstrap: https://react-bootstrap.github.io/components/modal/#live-demo

#67 APIs Delete User (DELETE)

APIs xóa người dùng
DELETE http://localhost:8081/api/v1/participant

#68 Hiển Thị Danh Sách User Phân Trang (Paginate)

Thư viện React-paginate: https://www.npmjs.com/package/react-paginate

Github: https://github.com/AdeleD/react-paginate

Link ví dụ: https://codepen.io/monsieurv/pen/abyJQWQ

APIs lấy danh sách người dùng phân trang:
GET http://localhost:8081/api/v1/participant?page=1&limit=2
(Truyền động tham số page và limit)
Về dấu nháy chéo backticks `, xem tại video #6 Template literals (Template strings) - Dấu nháy chéo ` backticks

#69 Design Login

Về thuộc tính 'forcePage' xem tại đây: https://github.com/AdeleD/react-paginate#props

useNavigate Hook : https://reactrouter.com/docs/en/v6/hooks/use-navigate

#70 API Login - Đăng nhập
APIs login:
POST http://localhost:8081/api/v1/login

HTML entity: https://www.w3schools.com/html/html_entities.asp

#71 Bài tập: Chức năng Register - Đăng Ký Người Dùng
APIs register:
POST http://localhost:8081/api/v1/register

Yêu cầu: xem tại cuối video trên :v

#72 Chức Năng Register

File Register (component + css) : https://drive.google.com/file/d/1WM_bA-BIEbY18wNtBIka_yuh9GnsA-y3/view?usp=sharing

CSS position: https://www.w3schools.com/css/css_positioning.asp

Chapter 7: Quản Lý State Application với Redux

Tìm hiểu các quản lý global state của ứng dụng React với thư viện Redux

#73 Why Redux ? Tại Sao Lại Cần Redux

1.Vấn đề gặp phải với React (không dùng Redux)

- Nguyên tắc chia sẻ data: Truyền từ cha (parent) xuống con (child) thông qua props. Truyền từ trên xuống dưới (top to bottom), thành ra component cha sẽ nằm ngoài bọc component con.

- Muốn chia sẻ data giữa 2 component cùng cấp, cách đơn giản nhất là lift-up state, đưa data cho component cha quản lý (xem video #62 về lift-up state)

- Chỉ có thể dùng props khi và chỉ khi 2 component cùng tồn tại trên màn hình, tức là có tồn tại component cha và con trên DOM

2.Tại sao dùng Redux

- Redux giúp quản lý trạng thái ứng dụng React (managing state) và không phụ thuộc vào nguyên tắc chia sẻ data (từ cha xuống con)
- Redux khác gì so với Context API: https://reactjs.org/docs/context.html

  Về chức năng, Redux không khác gì Context API, giúp chia sẻ data giữa các component không có quan hệ với nhau.

  Về hiệu năng, Redux vượt trội, vì 2 lý do chính.
  1.Redux hỗ trợ devtool dành cho lập trình viên, giúp việc kiểm tra data lưu trong redux cực kỳ thuận tiện
  2.Với context API, mỗi lần nó thay đổi, thì giao diện thay đổi. Không thể pause (dừng lại việc re-render). Với Redux, bạn có thể làm được, có thể quyết định xem có cho component re-render khi redux thay đổi hay không ?
  3.Thư viện sinh ra là có lý do của nó, nếu không, nó không tồn tại. Context API là level thô sơ, còn Redux là level đã cài tiến từ Context API :v

3.Lưu ý

- Với redux, hiện nay có 2 cách sử dung là: sử dụng redux thuần hoặc redux/toolkit Trong khóa học này, mình sử dụng redux thuần.

Về redux/toolkit, cũng như chuyên sâu về Redux, mình đã có 1 khóa học riêng về nó: https://www.udemy.com/course/hoidanit-redux-and-redux-toolkit-ultimate/

#74 Store - Lưu Trữ Data Redux

Redux diagram: https://redux.js.org/tutorials/fundamentals/part-2-concepts-data-flow#redux-application-data-flow

Store: (nơi lưu trữ) data Redux . Data sẽ được lưu bên trong thư viện Redux (với google Chrome thì sử dụng Redux devtool để xem data Redux - video #18)

Khi chạy ứng dụng React, thì Redux đã được cấu hình để chạy song song với ứng dụng React, cụ thể như sau:

- Tất cả logic liên quan về Redux được viết trong thư mục src/redux.
- Khi chạy ứng dụng React lên (file index.js), đã nhúng Store của Redux vào, thông qua thư viện react-redux.
  File index.js:
  <Provider store={reduxStore} > <= nạp data Redux tại đây
  <App/>
  </Provider>

- File store: quy định các thông tin sẽ lưu (thông qua reducer) và cấu hình cho redux (middleware như redux devtool, redux thunk...)

#75 Actions/Dispatch

Download project thực hành Redux:
https://drive.google.com/file/d/17u_Q40NVP95YIPIhtW6JqnXjkHQZ18Bz/view?usp=sharing

1.Actions

Tài liệu: https://redux.js.org/tutorials/fundamentals/part-3-state-actions-reducers#designing-actions

Actions (tên hành động) là cách ứng dụng React và Redux nói chuyện với nhau. Ứng dụng React sẽ yêu cầu Redux làm việc cho mình, thông qua việc sử dụng actions.

Cú pháp: Actions là 1 javascript object, giống với state/props của React.

Để 1 object javascript là 1 actions, thì phải khai báo theo cấu trúc:

```
const your_action = { type: 'action_name', payload }
```

Type: thuộc tính này bắt buộc, dùng để miêu tả hành động (action), nó giống như id để phân biệt các action với nhau.
Payload: data muốn truyền theo hành động, thuộc tính này KHÔNG bắt buộc phải có với action

Ví dụ về actions:

- Actions có payload (truyền data cùng actions)

  { type: 'todos/todoAdded', payload: todoText }

  { type: 'todos/colorSelected, payload: { todoId, color } }

- Action không truyền payload

  { type: 'todos/allCompleted' }

  { type: 'todos/completedCleared' }

2. Dispatch

Dùng keyword ‘dispatch’ để fire (thực hiện) 1 action.

React => dispatch(action) => Redux

Dispatch là 1 keyword đặc biệt, chỉ có thể hoạt động trong môi trường react-redux, cũng như nhờ có dispatch, chúng ta từ React, bắn actions vào Redux.

Redux sau khi nhận được actions, sẽ cần dùng Reducer để xử lý tiếp.

#76 Reducer

Reducer (công nhân của Redux): sau khi nhận được actions gửi từ React, Redux sẽ gọi tới Reducer để xử lý.

- Reducer sẽ dựa vào type (tên của actions) để biết nó cần làm gì.
  Đầu vào của Reducer là actions (gồm tên (type) hoặc có thêm data (payload))
  Đầu ra của Reducer là cập nhật 'state của Redux - thứ lưu trong Store Redux'

Reducer sẽ được cấu hình (theo template của Redux) gồm 2 tham số: initState (state mà Reducer quản lý) và actions đầu vào.

Reducer giống hệt như vòng switch - case (if/else), nó sẽ quét theo 'tên action' để biết cần xử lý như thế nào.

Tại sao khi khai báo action trong Reducer code lại chạy được ?
Vì tất cả reducer đã được khai báo ở file store.js, khi ứng dụng React chạy lên, thì nó đã biết có bao nhiêu actions cần xử lý rồi :)

#77 useSelector, useDispatch Hook

Đối với Redux/toolkit, thay vì dùng Reducer, thì sử dụng Slide. Tuy nhiên, tác dụng của 2 cái này là giống nhau (khác nhau về cú pháp viết)

1.useSelector Hook

useSelector cho phép chúng ta truy cập 'state của Redux', bằng cách cung cấp 'state' như là tham số đầu vào, cụ thể:

const count = useSelector(state => state.counter.count)

state: chính là state của Redux, được thư viện cho như là tham số đầu vào.
couter: tên gọi vắn tắt của Reducer (khai báo trong file store.js)
count: tên của state-redux được lấy ra từ counter Reducer

Cách viết trên là cách viết tắt của arrow function, cách viết đầy đủ là:
const count = useSelector((state) => { return state.counter.count })

Như vậy, khối code này sẽ trả ra giá trị, (state) => { return state.counter.count }, sau đấy gán ngược vào biến const count

Mỗi một khi giá trị lấy từ useSelector() thay đổi, thì ngay lập tức giao diện React bị re-render (vẽ lại). Amazing :)

2. useDispatch Hook

Công cụ để lấy ra keyword 'dispatch', dùng để fire actions từ React tới Redux

#78 Sử Dụng Redux Lưu Thông Tin Người Dùng

3 bước thực hiện:

Step 1: Khai báo dispatch + actions (tại component của React)

Step 2: Khai báo reducer + logic xử lý của nó (tại thư mục reducer)

Step 3: Sử dụng state của Redux (tại component React)

Video bổ trợ:
#7.Spread syntax (...) - Cú pháp toán tử mở rộng
#10 Optional chaining (?.)

#79 Loading Bar - Hiển Thị Thanh Loading Khi Gọi APIs

Tài liệu:

- React Icons: https://react-icons.github.io/react-icons/search?q=spinner
- CSS Make an icon spins: https://stackoverflow.com/questions/65298589/how-to-make-an-icon-spin-in-react
- CSS Disabled button: https://www.w3schools.com/cssref/sel_disabled.asp

Cài đặt thư viện: **npm install --save-exact nprogress@0.2.0**
Link github: https://github.com/rstacruz/nprogress

Customize thư viện: https://learnjsx.com/category/4/posts/nextjs-nprogress

#80 Redux persist - Xử lý Data Khi F5 Refresh

Thư viện redux-persist giúp ghi data Redux vào local Storage, đồng thời, khi F5 lại trang, nó sẽ tự động nạp data từ local Storage vào ứng dụng Redux.

Tài liệu:
Cài đặt thư viện: **npm install --save-exact redux-persist@6.0.0**

Github: https://github.com/rt2zz/redux-persist

Chapter 8: Doing Quiz - Chức năng Bài Thi

Thực hiện các chức năng của bài thi

#81 Design Danh Sách Bài Thi Của User - Display List Quiz

Bootstrap 5 Card: https://getbootstrap.com/docs/5.0/components/card/#example

API Lấy tất cả bài Quiz của User:
GET http://localhost:8081/api/v1/quiz-by-participant

Để lấy State của Redux, chúng ta sẽ sử dụng store và hàm getState của nó:
https://redux.js.org/api/store#getstate

Axios với header token: https://stackoverflow.com/a/55259078

Display a base64 image: https://stackoverflow.com/questions/8499633/how-to-display-base64-images-in-html

Lưu ý: các bạn có thể dùng .... Hoặc

...
Thì browser nó vẫn hiển thị được ảnh, thành ra chúng ta không cần phải quan tâm ảnh gốc là .jpeg hay .png nhé :)

#82 Chi Tiết Bài Quiz - Sử dụng URL Params

Not Found Route:
https://reactrouter.com/docs/en/v6/getting-started/overview#not-found-routes

React Router lấy tham số trên URL: https://reactrouter.com/docs/en/v6/getting-started/tutorial#reading-url-params

API Lấy Câu Hỏi của Bài Test:
GET `http://localhost:8081/api/v1/questions-by-quiz?quizId=${id}`

#83 Process Data - Xử Lý Data Phía Frontend
Lodash là thư viện giúp công việc xử lý data với Array và Object trở nên dễ dàng hơn rất nhiều (và hiệu năng cao)
Lodash Group By: https://stackoverflow.com/a/23600960

#84 Design Quiz Layout - Tạo Base Giao Diện Bài Thi

Ý tưởng design giao diện: https://www.anhngumshoa.com/test/mini-test-c3409.html
Các bạn vào link trên, đăng nhập bằng tài khoản Google để test nhanh nhé :)

Navigate với State (sử dụng React Router để lưu data) :
https://stackoverflow.com/a/52138179

#85 Design Question - Tạo Giao Diện Hiển Thị Question
File PDF vẽ quan hệ Parent-Child:
https://drive.google.com/file/d/11D96HwP40FqJoHNy3QdBeAbYGhf8Rksc/view?usp=sharing

Bootstrap 5 Checkbox: https://getbootstrap.com/docs/5.0/forms/checks-radios/#checks

#86 Xử Lý Data Khi Chọn Câu Trả Lời
React get checkbox value: https://stackoverflow.com/a/39270148

Chúng ta dùng Lodash để cloneDeep state, chứ không thao tác trực tiếp với State React.

Sự khác nhau giữa việc clone (deep copy) hay là gán biến state thành biến khác (shallow copy):
https://stackoverflow.com/a/184780

Ví dụ: const [questions, setQuestions] = useState(...) <- questions ở đây là 1 biến State.

Nếu viết: const myVar = questions ;
thì đây là shallow copy. Và khi chúng ta thay đổi biến myVar, dẫn tới biến questions thay đổi => thay đổi state thì giao diện re-render => dẫn tới bugs :)

Nếu viết: const myClone = _.cloneDeep(questions) ;

Thì đây là deep copy. Khi chúng ta thay đổi biến myClone thì không ảnh hưởng gì tới biến state questions ban đầu.

#87. Build Data Trước Khi Submit API

Bài này dễ quá, chẳng có gì để note cả :)

#88. Submit Quiz - Nộp Bài Test

API:
POST http://localhost:8081/api/v1/quiz-submit

Về toán tử 3 dấu chấm … , các bạn xem tại video #7.Spread syntax (...) - Cú pháp toán tử mở rộng

#89. Design Giao Diện Thêm Mới Bài Test

Floating label: https://getbootstrap.com/docs/5.0/forms/floating-labels/#example

Fix lỗi tag legend/fieldset của Bootstrap:
https://github.com/twbs/bootstrap/issues/32548#issuecomment-999596377

Kinh nghiệm đọc issue trên Github là cứ chọn câu trả lời nào có nhiều react (like, heart…) thì chúng ta test theo.

Cài đặt thư viện Select: **npm install --save-exact react-select@5.4.0**

#90. API Thêm Mới Bài Thi

API:
POST http://localhost:8081/api/v1/quiz

#91. Hiển Thị Danh Sách Bài Thi Admin

API:
GET http://localhost:8081/api/v1/quiz/all

Bootstrap Accordion: https://react-bootstrap.github.io/components/accordion/#examples

#92. Fix Lỗi ScrollBar

Cài đặt thư viện: **npm install --save-exact react-perfect-scrollbar@1.5.8**

#93. Bài Tập Sửa/Xóa Bài Thi

Link source code video này: https://drive.google.com/file/d/1wcit01Pk-oJ46-qs_lU3sE3n5t9MR8hM/view?usp=sharing

#94. Design Base Giao Diện Thêm Mới Questions/Answers

React icons: https://react-icons.github.io/react-icons/

#95. Tạo Fake Data Cho Giao Diện

Cài đặt thư viện: **npm install --save-exact uuid@8.3.2**

Source code video 95:
https://drive.google.com/file/d/1zNoqaRK9-6A5-ybd8GLQUO100ZHAREr5/view?usp=sharing

#96 State Hóa Data Questions

Todo…

#97 Preview Image

Tài liệu: https://www.npmjs.com/package/react-awesome-lightbox

Cài đặt thư viện: **npm install --save-exact react-awesome-lightbox@1.8.1**

Nếu cài đặt bị lỗi, thử câu lệnh:
npm install --save-exact react-awesome-lightbox@1.8.1 --legacy-peer-deps

#98 Lưu Questions/Answers

Tài liệu về hàm Promise.all : https://developer.mozilla.org/en-US/docs/Web/JavaScript/Reference/Global_Objects/Promise/all

Promise.all giúp gọi APIs (async/await) trong vòng lặp, và việc gọi APIs sẽ được thực hiện parallel (song song).

Việc dùng promise.all sẽ giúp tăng hiệu năng, và đảm bảo rằng tất cả các apis sẽ được gọi. Còn khi chúng ta dùng vòng lặp for, thì APIs sẽ được gọi tuần tự, tức là cái này xong, thì cái kế tiếp mới được thực hiện.

#99 Validate Questions/Answers

Để gọi APIs trong vòng lặp, chúng ta sử dụng For, thay vì map, forEach...

for ... of khác gì so với for ... in ?
https://stackoverflow.com/a/62328579

for...of lặp các phần tử (đối tượng), giống hệt map, forEach
for...in lặp theo index của mảng

#100 Design Update/Delete Quiz

Todo...

#101 Assign Quiz to User

API assign quiz to user:
POST http://localhost:8081/api/v1/quiz-assign-to-user

API get quiz data with questions/answers:
GET http://localhost:8081/api/v1/quiz-with-qa/${quizID}

Cần convert từ base64 về file để hiển thị:
https://stackoverflow.com/questions/35940290/how-to-convert-base64-string-to-javascript-file-object-like-as-from-file-input-f

Chapter 9 : Complete Project - Hoàn Thiện Dự Án
Hoàn thiện các chức năng của dự án

#102 API Update/Delete Questions/Answers

API upsert quiz:
POST http://localhost:8081/api/v1/quiz-upsert-qa

Convert từ file sang base64 rồi truyền vào APIs
https://stackoverflow.com/questions/36280818/how-to-convert-file-to-base64-in-javascript

#103 Design Base Questions - Right Content

Todo…

#104 CountDown Timer

1.setTimeout (chạy duy nhất 1 lần)
https://developer.mozilla.org/en-US/docs/Web/API/setTimeout
setTimeout(code, delay)

Hàm này cần truyền vào 2 tham số.

- Tham số thứ 2 (delay), tức là mình muốn sao bao lâu, tham số 1 sẽ được chạy, nên thông thường, tham số 1 là function

Ví dụ:

```
setTimeout(
 ( ) => {
        console.log('timeout');
 }, 3000 );
```

Ở trên, chúng ta truyền vào 1 arrow functions, và function đấy sẽ được thực thi sau 3s (đơn vị là ms)

2. setInterval (chạy lặp vô hạn)
https://developer.mozilla.org/en-US/docs/Web/API/setInterval
setInterval(code, delay);

Hàm này giống hàm setTimeout, cơ mà nó không phải chạy 1 lần, mà chạy vô hạn.

3.clearInterval (xóa setInterval)
https://developer.mozilla.org/en-US/docs/Web/API/clearInterval

clearInterval(intervalID)

Chúng ta dùng hàm này, để xóa sự lặp vô hạn của hàm setInterval theo Id của nó.

const myInterval = setInterval(code, delay); <= ở đây, chúng ta gán myInterval, như là 1 cách định danh cho hàm setInterval, vì trong code, chúng ta có thể code nhiều hàm setInterval, thì làm sao để phân biệt hàm nào là hàm nào, right ?

Tại sao khi code setState bên trong hàm setInterval, code chạy không đúng ?

```
useEffect ( () => {
        setInterval( () => {
                setCount(count - 1) ; <= code như này không chạy đúng ? why ?
         }, 1000);
}, [ ]);
```

Lý do code ở phía trên không chạy đúng, chỉ chạy đúng 1 lần duy nhất, vì hàm useEffect rất đặc biệt, cũng như là đặc thù của React Hook, là nó không thể truy cập được giá trị quá khứ của biến.
https://stackoverflow.com/a/53024497

setCount(count - 1) <= code sai ở đây
Sửa thành: setCount(count => count - 1) , nó sẽ đúng :), cơ mà mình không muốn code kiểu này, vì nó sẽ gây ra sự khó hiểu cho các bạn.

Tại sao lại cần cleanUp ?
Giai đoạn cleanUp giúp chúng ta thực thi code, trước 1 lần render mới.

Hàm convert Time: https://stackoverflow.com/a/34841026

#105 Select Questions - Thêm Hiệu Ứng

Về useRef, mình sẽ đề cập trong khóa React Nâng Cao :)

#106 Private Route

https://www.robinwieruch.de/react-router-private-routes/

Về children, mình đề cập trong khóa React Nâng Cao :)

#107 Chức năng Logout

API logout:
POST http://localhost:8081/api/v1/logout

#108 Design Header - Cài Đặt Thư Viện Cho Languages

Bài post trong video: https://dev.to/adrai/how-to-properly-internationalize-a-react-application-using-i18next-3hdb

Cài đặt thư viện:
npm install --save-exact i18next@21.8.16 react-i18next@11.18.3 i18next-browser-languagedetector@6.1.4 i18next-http-backend@1.4.1

#109 Tích Hợp Chuyển Đổi Ngôn Ngôn

Download file cấu hình (i18n.js)
https://drive.google.com/file/d/1Ur9bvKgCpHIIJ_BVfG9HE6WrcYwsrkC5/view?usp=sharing

Về Suspense sẽ được đề cập trong khóa react nâng cao

#110 Design DashBoard

Cài đặt thư viện: **npm install --save-exact recharts@2.1.13**

#111 Tích hợp API Dashboard

Set height, width chart: https://stackoverflow.com/a/55839553

API dashboard:
GET http://localhost:8081/api/v1/overview

#112 Bài Tập Chuyển Đổi Ngôn Ngữ

Todo…
Bootstrap Breadcrumb: https://react-bootstrap.github.io/components/breadcrumb/

#113 Bài Tập Update Profile

Todo…
Bootstrap Tabs: https://react-bootstrap.github.io/components/tabs/

#114 Bài Tập Hiển Thị Kết Quả Làm Bài Quiz

Tài liệu trong video :
https://drive.google.com/file/d/1k0kuSEyV3S0g_5MKnVXy0gWLrWJAC_kO/view?usp=sharing

#115 Nhận Xét Về Dự án Bài Quiz

Ưu điểm:

- Hiểu được 1 ứng dụng React thực tế cần có 'tối thiểu' những tính năng như thế nào.
- Hiểu được cách hoạt động của React thông qua sử dụng State/Props và các hook cơ bản (useState, useEffect)

Nhược điểm:

- Chưa tối ưu hóa code 1 cách tốt nhất có thể. Phần tối ưu này, đòi hỏi các bạn cần nắm vững React cơ bản, vì vậy, mình sẽ đề cập trọng khóa học React Advance.

- Trong khóa học này, chúng ta đang dừng lại ở chỗ, code làm sao cho hiểu React, và code của chúng ta 'chạy được', chứ hiệu năng chưa cao :)

What's next ?

Level tiếp theo sau khóa học này, chính là các chapter 10, 11, 12, 13.

Enjoy :v

#116. Giới thiệu về lộ trình mới (NEW)

Khóa học được chia làm 2 phần nội dung chính:
Từ chapter 0 tới chapter 9:

- Học kiến thức cơ bản của React và thực hành dự án làm bài quiz
- Kiến thức sử dụng trang tài liệu: https://legacy.reactjs.org/

Từ chapter 10 tới chapter 13:

- Học sâu hơn về các kiến thức cốt lõi của React
- Học công cụ để tối ưu hóa source code dự án bài quiz
- Kiến thức sử dụng trang tài liệu: https://react.dev

Lưu ý: với chapter từ 10 tới 13, sử dụng chung backend với dự án Quiz (nên sẽ không cần setup backend nữa)

Chapter 10: Think In Modern React (NEW)

Không quan trọng làm như thế nào, mà điều quan trọng là làm đúng hướng. (architecture)

#1. Setup Environment

VSCode - Công cụ để code Javascript:

- Tải VSCode: https://code.visualstudio.com/download
- Sử dụng VSCode vì nó free, và support mạnh mẽ cho javascript

Node.js - Môi trường thực thi Javascript

- Download Node.js: https://nodejs.org/en/download/
- Node.js là môi trường để chạy code, chứ không phải là framework hay library. Nó là platform. Tương tự như windows là môi trường để chạy ứng dụng Microsoft Words (ở đây, node.js là môi trường để chạy ứng dụng React - viết bằng ngôn ngữ js)

Setup project thực hành

- Download project react with Vite: (v3.0)
  https://drive.google.com/file/d/1kar90KNWOHjRNItz5yBWtkzcqj3CgWwY/view?usp=sharing

- Các bước cần thực hiện để cài đặt project thực hành:
  B1: Đảm bảo rằng máy tính các bạn đã cài đặt Node.js

  B2: Download project từ link ở trên

  B3: Chạy câu lệnh npm i để cài đặt các thư viện cần thiết (super fast :))

  B4: Chạy câu lệnh **npm run dev** để chạy project. Mặc định, project sẽ chạy trên cổng 5173

  Lưu ý: với React Vite, project chạy trên port 5173, chứ không phải 3000 như khi khởi tạo với create-react-app.

SOS - Cần support thì làm thế nào ?

Để đảm bảo quyền lợi khi học trên Udemy, các bạn học viên vui lòng inbox trực tiếp qua Fanpage https://www.facebook.com/askITwithERIC/ để được hỗ trợ, cung cấp đầy đủ tài liệu và giải đáp thắc mắc nhé !

#2. Component

Why component ? Tại sao lại sử dụng component

- Component giúp chia UI (user interfaces) thành các thành phần độc lập, và có thể tái sử dụng được.
- Ví dụ về tổ chức components đơn giản trong ứng dụng React (source code)

Cách chia bố cục website (layout) theo components:

Gồm 2 cách chính:

+ Freestyle : code trước tính sau, đến đâu thì đến :) . Cứ code theo cách các bạn hiểu, điều quan trọng là chúng ta lên được bố cục của 1 website bằng cách sử dụng React .
  Cách làm này thường được sử dụng bởi các bạn mới làm quen với React :)

+ Chia component theo top/down hoặc bottom/up : Với cách làm này, chúng ta sẽ cần hình dung bố cục website trước, và chia layout.

  Top/down tức là chia từ trên xuống dưới, từ component Cha cho tới component con. Các bạn có thể code freestyle, sau đấy, tách dần component cho nó nhỏ ra. Từ 1 cha ban đầu, phân thành N con nhỏ hơn.

  Bottom/up tức là chia từ dưới chia lên, từ component con cho tới cha. Ở đây, chúng ta sẽ đi từ những thành phần nhỏ nhất, những phần có điểm chung thì gộp lại thành 1 component cha.

#3. State và Props

Có 2 loại 'model' cho data trong ứng dụng React: props và state. Và chúng khác hẳn nhau:

Props (viết tắt của properties - tài sản thừa hưởng (belong to someone):

- Cho phép parent component truyền data xuống child component (parent to child)

ex: pass 'color' props to a Button

State (component's memory) :

- Giúp component biết nó đang kiểm soát thông tin gì (keep track of informations)
- Thực hiện thay đổi tương ứng khi có sự tương tác của người dùng

ex: form with input state

#4. Data Flow - Luồng Chảy Của Dữ Liệu

React data flow: top to bottom, parent to child (with props)
=> one-way data binding.

one-way vs two way data binding:

ex: two way binding with angular 2: https://angular.io/guide/two-way-binding
- View (html) => Update Model (js/ts)
- Model (js/ts) => Update View (html)

https://codesandbox.io/s/two-way-data-binding-example-forked-kxbrcp?file=/src/app/app.component.html

ex: one way binding with react: form with input
state => control
Model (js/ts) => Update View (jsx)

#5. Control Component/Uncontrolled Component

- Đa số các trường hợp được sử dụng với Form.

- Với controlled components: dữ liệu của form được React kiểm soát (thông qua state).
=> thao tác với state

- Với uncontrolled components: dữ liệu của form được DOM quản lý.
Muốn lấy dữ liệu form, phải thao tác trực tiếp với DOM (html)
=> thao tác với Ref

feature	uncontrolled	controlled
one-time value retrieval (e.g. on submit)	✅	✅
validating on submit	✅	✅
instant field validation	❌	✅
conditionally disabling submit button	❌	✅
enforcing input format	❌	✅
several inputs for one piece of data	❌	✅
dynamic inputs	❌	✅

Ví dụ về controlled component:
Form with 100 inputs

Ví dụ về uncontrolled component
Thao tác với ref

#6. Form Mik vs React Form Hook

Form Mik và React Form Hook là 2 thư viện rất phổ biến về xử lý Form với React. Cả 2 thư viện này, đều sử dụng nguyên tắc của 'Uncontrolled component' để cải thiện hiệu năng.

Thông tin nhanh về 2 thư viện này, các bạn xem tại link:
https://blog.logrocket.com/react-hook-form-vs-formik-comparison

https://apiumhub.com/tech-blog-barcelona/react-form-libraries-comparison-formik-vs-react-hook-form/

Lưu ý: đối với form thông thường, không cần thiết phải sử dụng tới 2 thư viện trên. Gần như tới 90% các trường hợp các bạn đi làm, sử dụng 'controlled component với state' là đã ok và đáp ứng được nhu cầu làm việc rồi.

Chỉ sử dụng thư viện, khi và chỉ khi, các bạn cần cải thiện hiệu năng, "chống hiện tượng" giật và lag. Còn sử dụng thư viện nào, là quyền của các bạn, chứ không có khái niệm thư viện nào là tốt nhất ở đây.

Ưu, nhược điểm khi sử dụng thư viện:

- Pros: Code ngắn hơn và tối ưu hóa performance (số lần render ít hơn)
- Cons: Phải học cú pháp của thư viện để hiểu nó hoạt động như thế nào.

#7. Anonymous Functions (Hàm - vô danh)
Tài liệu: https://www.javascripttutorial.net/javascript-anonymous-functions/

Anonymous functions:
- Là function 'không có tên' (anonymous - vô danh).

Function truyền thống:
function pickAName() {...}

Anonymous function:

(function() {...})

=> lưu ý điểm đặc biệt của function trên:
- Không có tên
- Cặp dấu ( ) bọc lại function

Mặc định, sau khi khai báo anonymous function, chúng ta không thể thực thi được nó (vì nó không có tên)

ví dụ: khai báo 2 anonymous functions, thì làm sao biết để thực thi functions nào, hay thực thi cả 2 :)

Có 2 cách để thực thi anonymous function:
Cách 1: assign to a variable (gán tên cho function, để biết nó là ai, tương tự như là ID)
ex:
const doSomething = function() { ... } <= lưu ý không có () bọc functions. nếu có, js sẽ thực thi function luôn
doSomething(); <= thực thi functions

vd trong thực tế: anonymous functions được sử dụng như là tham số truyền vào function:
```
setTimeout(function() {
   console.log('Execute later after 1 second')
}, 1000);
```

Cách 2: self-running function (Self-Executing Anonymous Function)

```
(function() {
   console.log('IIFE');
})();   <= có thêm () ở cuối functions => dấu hiệu để thực thi functions
```

Tại sao lại cần self-running function, thay vì viết 1 loạt code
https://stackoverflow.com/questions/592396/what-is-the-purpose-of-a-self-executing-function-in-javascript

Arrow functions:
- Được giới thiệu từ ES6, arrow functions là cách ngắn gọn để định nghĩa 1 anonymous functions.

```
const doSomething = function () { ... }
```

có thể được viết ngắn gọn thành:

```
const doSomething = ( ) => { ... }
```

anonymous functions: function() { }
 => chuyển thành :

```
( ) => {...}
```

#8. Adding Event Handlers

- Thường định nghĩa bên trong component
- Thường bắt đầu với keyword 'handle' + 'tên event'
ex: handleClick, handleOnChange

3 cách định nghĩa 1 event handler:
- tham chiếu (ko có dấu ())
- inline function
- arrow function

Phân biệt truyền tham chiếu (passing a function) và gọi 1 function (calling a function)
<button onClick={handleClick}>
=> function này (không có dấu ( )) được truyền (pass) vào eventHandler onClick.

React sẽ nhớ hàm này, và chỉ thực hiện khi user click vào button (event onClick fired)

<button onClick={handleClick( )}>
=> function này có dấu ( ), nó sẽ nói với React thực thi hàm này 'ngay lập tức' khi giao diện được render (mà không cần click vào button)

passing a function (correct)	calling a function (incorrect)
<button onClick={ ( ) => alert('...') }>	<button onClick={alert('...')}>

=> truyền vào anonymous functions (passing), chứ không phải calling (execute/running)

<button onClick={handleClick}> // passes the handleClick function.

<button onClick={ ( ) => alert('...')}>   // passes the ( ) => alert('...') function.

#9. Stop Something (Event Propagation/PreventDefault)

1.Preventing default behavior

Một vài trình duyệt có các hành động mặc định, ví dụ:
Sử dụng <form> với submit event. Khi 1 button trong form click, nó sẽ reload toàn bộ page (f5/refresh)
eg:...
enter => auto submit form :v

Để ngăn chặn tình trạng trên, sử dụng: event.preventDefault()

đôi khi viết là 'e', viết tắt của event.
dùng 2 cách viết.
ví dụ về passing functions
or inline function with arrow function

2. Event propagation

Minh họa ví dụ:
 có 2 button...

Mặc định, Event handler sẽ 'bắt -catch' events từ bất kỳ 'children' nào trong component.
chúng ta gọi nó là 1 event "bubbles" or "propagates" up the tree - event nổi lên từ 'gốc' tới 'ngọn' cây.
event bắt đầu tại nơi nó xảy ra, sau đấy đi ngược lên phía trên (parent)

-stopping propagation:
e.stopPropagation( ) => ngăn việc event 'bubbles' further

Đừng nhầm lẫn event.stopPropagation( ) và event.preventDefault(). Chúng đều hữu ích, tuy nhiên, không liên quan gì tới nhau:
- event.stopPropagation( ) ngăn việc event handler được gọi ở lớp phía trên (lớp cha).
- event.preventDefault( ) ngăn việc hành động mặc định của browser đối với 1 số event, ví dụ onSubmit form thì refresh - f5 website

#10. Keeping Components Pure (Nguyên Chất)

- A Pure Function (trong thế giới JS) là 1 function (1 block code) "luôn trả về cùng kết quả" với "cùng input đầu vào"

ex pure function:

```
function calculateVAT( productPrice ) {
   return productPrice * 0.05;
}

console.log(">>> VAT === ", calculateVAT(100))
```

ex not pure function (không phụ thuộc vào input đầu vào)

```
 var tax = 5;
 function calculateVAT(productPrice) {
  return productPrice * tax / 100;
 }

==========================
 function doSomething() {
  //..whatever
  tax = 100;
 }
 doSomething();
 console.log(">>> VAT === ", calculateVAT(100))
```

---

với React:

==== impure function:

```
let price = 100, tax = 0;
const CalculateVAT = (props) => {
        //mutate
        tax = tax + 5;

        return (
          <div> VAT = { price * tax/100 } with PRICE = {price}</div>
        )

}
```

```
<CalculateVAT />
<CalculateVAT />
<CalculateVAT />
```

```
===== pure component
const CalculateVAT = (props) => {
        const {tax, price} = props;
        return (
          <div> VAT = { price * tax/100 } with PRICE = {price}</div>
        )
}
<CalculateVAT tax={5} price={100} />
<CalculateVAT tax={5} price={100} />
<CalculateVAT tax={5} price={100} />
```

- Tại sao cần nên viết pure functions:
+ Cùng input, cùng kết quả trả ra => hạn chế bugs và tiên đoán được kết quả trả ra.
+ Với React:
- Nếu biến bị thay đổi là state/props => dẫn tới sự render không cần tiết (không tối ưu hóa hiệu năng)
- Nếu cần thay đổi giá trị của biến => sử dụng setState

#11. Strict Mode
https://reactjs.org/docs/strict-mode.html

- Kỹ thuật render 1 lần với Ref

Video từ #12 tới #16 là hướng dẫn cài đặt backend, tuy nhiên, không cần thiết nữa, vì backend của các chapter này chính là backend của dự án bài Quiz

=> Các bạn tiếp tục xem #17.1 bình thường.

Chapter 11: Copy với Javascript (NEW)

Hiểu Javascript, Hiểu React một cách dễ dàng hơn

#17.1 Primitive data type

Tài liệu:
https://developer.mozilla.org/en-US/docs/Web/JavaScript/Data_structures
https://www.javascripttutorial.net/javascript-data-types/

I.Primitive data type: data nguyên thủy, không phải là object,
không có phương thức (methods) hay thuộc tính (properties)

Có 7 loại primitive data types:
1.string
- là tập hợp của chuỗi ký tự, bắt đầu và kết thúc với dấu nháy đơn ' hoặc dấu nháy đôi "
let greeting = 'Hi';
let message = "Bye";

- Nếu muốn dùng dấu nháy đơn ' và dấu nháy đôi trong 1 chuỗi string cùng lúc, sẽ cần phải 'escape' character.
let message = 'I\'m also a valid string'; // use \ to escape the single quote (')

- Đơn giản hơn, có thể sử dụng Template Strings với dấu nháy chéo(backticks) (ES6)
https://developer.mozilla.org/en-US/docs/Web/JavaScript/Reference/Template_literals

let message = `I'm also a valid string use "" and '' `;

2. number
- javascript sử dụng kiểu dữ liệu 'number' để hiển thị cho số nguyên (integer) và số thực (float)
let num = 100;
let price = 12.5; <= sử dụng dấu . để định nghĩa số thực
typeof price; // number

- Khoảng giá trị an toàn để sử dụng number (integer): -(2^53 – 1) and 2^53 – 1 (9,007,199,254,740,991 or ~9 quadrillion)
- với float; 2^1024
3. boolean
- Kiểu dữ liệu boolean có 2 giá trị true và false (viết thường)

let inProgress = true;
let completed = false;

console.log(typeof completed); // boolean

4.undefined

- Khi một biến được khai báo, tuy nhiên chưa 'gán/khởi tạo' giá trị cho nó, JS sẽ gán (assign)
giá trị 'undefined - chưa được định nghĩa' cho nó.

let counter;
console.log(counter); // undefined
console.log(typeof counter); // undefined

5.null

- Khi một biến được khai báo, tuy nhiên không có kiểu dữ liệu hoặc giá trị => null;
const foo = null;
console.log(foo); //null

6.bigint (ES2020)

- Dùng để hiển thị số có giá trị lớn hơn 2^53 -1
- Để khởi tạo 1 biến là BigInt, thêm ký tự 'n' vào cuối number:
let bigInt = 9007199254740991n;
hoặc gọi hàm khởi tạo: let bigInt = BigInt(9007199254740991);

console.log(typeof bigInt); // bigint

===before với number
let x = Number.MAX_SAFE_INTEGER;
x = x + 1; // 9007199254740992
x = x + 1; // 9007199254740992 (same as above => bugs)
===
let x = BigInt(Number.MAX_SAFE_INTEGER);
x = x + 1; // 9007199254740992n
x = x + 1; // 9007199254740993n (correct now)

7.symbol (ES6)

- Để tạo 1 biến symbol, sử dụng hàm Symbol()
let s = Symbol('foo');
- Symbol() function tạo ra giá trị 'độc nhất' (unique) mỗi lần gọi nó.
console.log(Symbol() === Symbol()); // false

https://www.javascripttutorial.net/es6/symbol/

#17.2 Objects data type

II. Objects

"Normal" object: { key: value }
const person = { name: 'ABC', address: 'HN' }

Dates

Indexed collections: Arrays

Keyed collections: Maps, Sets, WeakMaps, WeakSets

#18. Convert to a Boolean

Kiểu giá trị boolean có 2 loại giá trị: true và false (viết thường)

```
let inProgress = true;
let completed = false;

console.log(typeof completed); // boolean
```

Javascript cho phép chuyển đổi (convert) các kiểu dữ liệu khác sang boolean.
Để làm được điều đấy, chúng ta sử dụng hàm Boolean().

ví dụ:

```
console.log(Boolean('Hi'));// true
console.log(Boolean(''));  // false

console.log(Boolean(20));  // true
console.log(Boolean(Infinity));  // true
console.log(Boolean(0));  // false

console.log(Boolean({foo: 100}));  // true on non-empty object
console.log(Boolean(null));// false
```

Quy luật để convert sang boolean:

Type	true	false
string	non-empty string	empty string
number	non-zero number and Infinity	0, NaN
object	non-null object	null
undefined		undefined

Với javascript và React, nó có ý nghĩa gì ?

- Sử dụng với câu điều kiện if...else

```
if( condition ) {
  // ...
} else {
  // ...
}
```

thông thường, condition sẽ là 1 biến có giá trị có giá trị true/false. Tuy nhiên, nếu truyền vào 1 biến không là boolean, if..else sẽ cố gắng convert 'biến đấy' sang boolean theo quy luật trong bảng trên.

ex:

```
let name = 'eric'; //datatype: string
if(name){ //do something}
=== if(name === true){ //do something}

let address;
if(address){//do something}
```

- Sử dụng với toán tử điều kiện trong JSX
(tự động convert biến thành boolean, giống việc sử dụng condition/Boolean( ) function)

```
let x = 10;
let y = true;
<>
        { x && x >  5 && <span> I'm greater than 5</span> }
        { y && x >  5 && <span> I'm greater than 5</span> }
        { y === true && x >  5 && <span> I'm greater than 5</span> }
</>
```

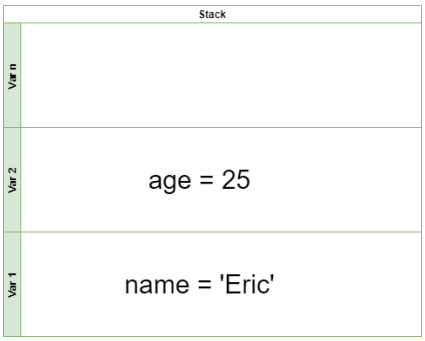
#19. Storing Variables JS

Javascript có 2 loại 'kiểu giá trị' (data types):
- primitive values: giá trị nguyên thủy
    + là các phần tử thô sơ cấu thành nên data (atom)
    + bao gồm: string, number, boolean, bigint, null, undefined, symbol
- reference values: giá trị tham chiếu
    + là đối tượng có thể bao gồm nhiều giá trị nguyên thủy
    + bao gồm object (object truyền thống, array...)

Để lưu trữ biến, JS sử dụng 2 vùng bộ nhớ: stack và heap.

Với data tĩnh (static data - size không đổi tại thời điểm dịch code), bao gồm:
- primitive values (7 loại datatype ở trên)
- Các biến tham chiếu để trở tới 'objects'

```
let name = 'eric';
let age = 25;
```
(static data - primitive data types)



Lưu ý: với nhiều ngôn ngữ lập trình khác (ví dụ java và C#), string là object.
tuy nhiên, với javascript, string là giá trị nguyên thủy (primitive values)

- Khác với stack, heap là nơi để lưu trữ các giá trị 'dynamic' (giá trị động về size) thường lưu trữ objects và functions.

```
let name = 'eric';
let age = 25;
let person = { name: 'eric',  age: 25,};
```

person là biến tham chiếu tới object
=> person lưu tại stack, còn giá trị của nó, lưu tại heap.

- Const khi sử dụng với object/array:

```
const person = {
  name: 'Eric'',
  age: 25,
};
```

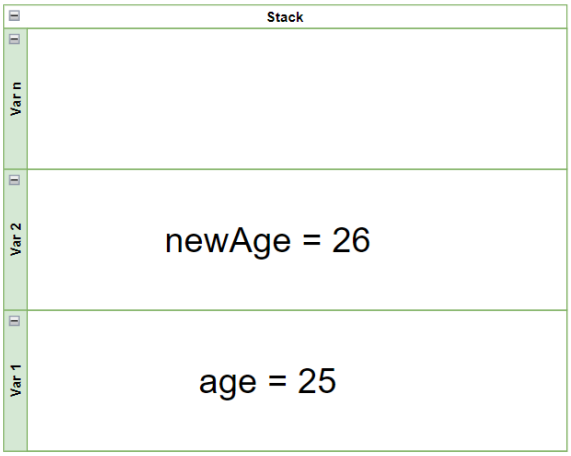
```
// add the ssn property
person.phone = '123-45-6789';

// change the name
person.name = 'Hỏi Dân IT'';

// delete the age property
delete person.age;
console.log(person);
```

#20. What's wrong with 'normal' assign/copy ?

- Copying values:
 + Với primitive values:

```
let age = 25;
let newAge = age;
```

javascript sẽ:
- Định nghĩa biến age, gán giá trị là 25.
- Định nghĩa biến newAge, copy giá trị biến age, và gán (assign) cho biến new Age



- ở bên trong stack, 2 biến age/newAge là 2 biến riêng biệt, không liên quan gì tới nhau, và sẽ không ảnh hưởng tới nhau.

ví dụ, thay đổi biến này, không làm ảnh hưởng tới biến khác

```
let age = 25;
let newAge = age;
```

```
newAge = newAge + 1;
console.log(age, newAge);
```

với reference values:

```
let person = {
  name: 'Hỏi Dân IT'',
  age: 25,
};

let member = person;

member.age = 26;

console.log(person);
console.log(member);
```

thay đổi biến này, dẫn tới thay đổi biến kia :)

#21. Shallow copy và Deep copy

Với javascript, chúng ta có 2 dạng datatypes: primitive và references.
primitive: number, string, boolean… (video #17.1)
references: objects (video #17.2)

```
let counter = 1;
let copiedCounter = counter;
copiedCounter = 2;

console.log(" counter = ", counter, " and copiedCounter = ",copiedCounter)


let person = {
   firstName: 'John',
   lastName: 'Doe'
};
let copiedPerson = person;

copiedPerson.firstName = 'Jane';
console.log(person);
```

==== deep copying nghĩa là 'các giá trị của biến mới' không liên quan gì tới biến cũ.
2 biến tạo ra tách biệt nhau. thay đổi biến này không làm thay đổi biến còn lại.
(ví dụ counter)

deep === sâu

shallow copying nghĩa là 'copy một phần giá trị của biến'. thay đổi giá của biến copy khiến thay đổi biến gốc.
(vì 2 biến đang trỏ tới cùng 1 object)
Shallow = not deep

Để copy object trong JS, chúng ta có các lựa chọn sau:
- sử dụng spread (...) syntax
- sử dụng Object.assign() method
- sử dụng JSON.stringify() và JSON.parse() methods
- sử dụng clone/cloneDeep() methond (thư viện lodash)

#22. Spread syntax (...) được sử dụng để copy array/object

- shallow copy (1 level deep) => lưu ý khi mutate

ví dụ về ghi đè object:

```
const obj = {
  name: 'abc',
  address: 'hn',

  //
  name: 'def'
}

console.log(">>> ", obj)
```

ví dụ về copy:

```
const person = {
  name: 'abc',
  address: {
    city: 'hn',
    country: 'vn'
  },
  job: {
    title: 'dev',
    detail: {
      position: 'boss',
      salary: '5k'
    }
  }
}

// modify directly
// const clone = { ...person };
// clone.name = 'update name' // ok
// clone.address.city = 'hcm'; //2 level deep => not ok

// spread syntax
```

```
const clone = {
  ...person,
  address: {
    ...person.address,
    city: 'hcm'
  }
}

console.log(">>> check person: ", person, ' clone= ', clone)
```

#23. Object.assign() method

```
const person = {
  name: 'abc',
  address: {
    city: 'hn',
    country: 'vn'
  },
  job: {
    title: 'dev',
    detail: {
      position: 'boss',
      salary: '5k'
    }
  }
}

const education = {
  edu: {
    school: 'bk',
    degree: 'doctor'
  },
  // name: 'my education info'
}

//Object.assign(target, sources);
// const mine = Object.assign(person, education); // let x = y; x <= y
// console.log(">>> check person: ", person, ' mine:', mine)
```

```
//copy object
const clonePerson = Object.assign({}, person); // let x = y; x <= y
console.log(">>> check person: ", person, ' clonePerson:', clonePerson);

const mine = Object.assign({}, person, education);
console.log(">>> check person: ", person, ' mine:', mine);


//mutate object
mine.name = 'update name';
console.log(">>> check person: ", person, ' mine:', mine); // 1 level deep => ok


mine.job.title = 'Engineer';
console.log(">>> check person: ", person, ' mine:', mine); // 2 level deep => NOT ok

//shallow copy :)
```

#24. JSON.stringify() và JSON.parse() methods

JSON stands for JavaScript Object Notation

JSON is a "text format" for storing and transporting data

JSON (key phải có dấu " ")
{"name":"John"}

Javascript:
{name:"John"}

==========
JSON.parse() //convert to something

ví dụ: chúng ta nhận được 1 dạng text (từ server):

```
//data types: string
 let data = '{"name":"John", "age":30, "city":"New York"}'
```

sử dụng JSON.parse sẽ convert text thành javascript object

```
const obj = JSON.parse(data);
const obj = JSON.parse('{"name":"John", "age":30, "city":"New York"}');
```

//Lưu ý: text truyền vào hàm JSON.parse() phải ở dạng JSON format. nếu không ERROR.

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

==========
JSON.stringify()
- giúp convert từ javascript object thành JSON format (string)

(json.parse() => convert từ string - JSON format, thành js object)

```
const obj = {name: "John", age: 30, city: "New York"};
const myJSON = JSON.stringify(obj);

//typeof myJSON === 'string
```

```
============
clone object
  const person = {
    name: 'abc',
    address: {
      city: 'hn',
      country: 'vn'
    },
    job: {
      title: 'dev',
      detail: {
        position: 'boss',
        salary: '5k'
      }
    }
  }

  //convert object to string => tạo 1 vùng bộ mới trong stack (primitive values)
  //convert string to object => tạo 1 bộ mới trong heap (object values)
  const clone = JSON.parse(JSON.stringify(person));

  console.log(">>> check person: ", person, ' clone:', clone);


  //mutate
  clone.address.city = 'update city';
  console.log(">>> check person: ", person, ' clone:', clone);
  // deep copy
```

#25.1 clone/cloneDeep() methond (thư viện lodash)

npm install --save-exact lodash@4.17.21
//import _ from 'lodash';

_.clone(value)
// supports cloning arrays, array buffers, booleans,
 date objects, maps, numbers, Object objects, regexes, sets, strings, symbols, and typed arrays.

//shadow copy

_.cloneDeep(value)
//deep copy

#25.2 Immutable.js

Tài liệu:
https://immutable-js.com/
https://github.com/immutable-js/immutable-js#the-case-for-immutability

#25.3 Immer.js

Tài liệu:
https://immerjs.github.io/immer/

Chapter 12: Deep Understand about React's State (NEW)

Nếu bạn đã hiểu tại sao chúng ta cần copy State (chương 3), thì chương này sẽ hướng dẫn bạn đi đúng hướng để trở thành 'pro' với React State

#26.1 Bài tập Tạo Component Hiển Thị Thông Tin

Yêu cầu: tạo giao diện theo yêu cầu trong video và APIs được cung cấp

#26.2. State is isolated and Private

- Mỗi 1 component trên màn hình, luôn 'tự quản lý' state của nó.

Điều này có nghĩa là, nếu có 1 component được render 2 lần, mỗi bản copy (component) sẽ có state riêng và không hề liên quan tới nhau. Thay đổi component này (state) sẽ không ảnh hưởng tới component kia.

Ví dụ:
=== Show/hide component
Todo: adding code sample

- Lợi ích của việc giữ state 'riêng biệt' (isolated) và 'riêng tư' (private):

+ Parent không thể thay đổi 'trực tiếp' state của component con.
Nếu muốn, thì phải làm gián tiếp, thông qua component con tự cập nhật state của nó (sẽ đề cập khi sử dụng useImperativeHandle hook)

+ Thay đổi state của 1 component bất kỳ, không làm ảnh tới 'tất cả' các components còn lại.

What if ? Chúng ta muốn 2 component (copy) thay đổi 'cùng lúc' - sync state. Hide/show components cùng lúc ?

#27.1 Bài Tập Tạo Component Add New User

Bài tập: Tạo component Add new user

#27.2. Sharing State Between Components (Kỹ Thuật Lift-up State)

- Đôi khi, chúng ta muốn 'state' của 2 components luôn thay đổi cùng nhau.
Vd: thay đổi State 'a' của component X sẽ dẫn tới thay đổi State 'b' của component Y.

Để làm được điều này, chúng ta sẽ 'remove state' tại 2 component trên, và 'chuyển' tới parent 'gần nhất của chúng',
sau đấy, pass state từ component parent xuống children thông qua props.

Kỹ thuật này được gọi là 'lifting state up' (lift-up state), một kỹ thuật rất hay được dùng khi viết code React.

Ví dụ về lift-up state...

#28.1 Render as snapshot

Settings state triggers renders (setState)
nguyên nhân - kết quả => khiến hành động khác xảy ra (triggers)

UI thay đổi ngay lập tức khi '1 event' xảy ra, ví dụ như use click button.
Với mô hình 'one way binding data', React sẽ cần phải 'update state'.

Quá trình Rendering (re-render):

"Rendering" có nghĩa là React gọi 'component' của bạn (1 js function), sau đấy trả JSX (từ hàm return của component) - snapshot

Tuy nhiên, không giống như 'a photograph", React snapshot có thêm sự tương tác.

Nó sẽ bao gồm tất cả event handler bên trong JSX (snapshot = html + js)

sau đấy, React sẽ update screens để cho 'nó khớp' (match) với snapshot vừa được tạo (bao gồm html, js - các event handlers)

3 bước:
- React executing the function
- Calculating the snapshot
- Updating the DOM tree

#28.2 Bài tập Vận Dụng Khái Niệm Render Snapshot

Bài tập : Viết 1 hàm để tăng giá trị lên 3 đơn vị

- Đoạn code sau sai ở đâu ?

ví dụ về Render của React:

```
<button onClick={() => {
  setNumber(number + 1);
  setNumber(number + 1);
  setNumber(number + 1);
}}>+3</button>
```

#28.3 Giải thích cơ chế tạo Snapshot của React

ví dụ về Render của React:

```
<button onClick={() => {
  setNumber(number + 1);
  setNumber(number + 1);
  setNumber(number + 1);
}}>+3</button>
```

State là 1 biến js đặc biệt, và nó chính là 'component memory'. Điều đặc biệt ở đây, đấy chính là state thực sự 'không sống bên trong component', skeptical :)

State sống ở ngoài component, bên trong React.
- User click event (onClick button)
- React tính toán State (bên trong nó)
- React copy 1 bản snapshot của state (bên trong nó), kết hợp JSX để tạo ra '1 version'.
- React sử dụng version vừa tạo để update DOM

(Cái gì thay đổi theo thời gian thì tạo ra snapshot) - state :v

setNumber(number + 1): number is 0 so setNumber(0 + 1).
React prepares to change the number to 1 on the next render.

Mặc dù setNumber (number + 1) được gọi 3 lần, number luôn = 0

== Trong lần render tiếp theo (sau khi number = 0 + 1), nó sẽ trông như là:

```
<button onClick={() => {
  setNumber(1 + 1);
  setNumber(1 + 1);
  setNumber(1 + 1);
}}>+3</button>
```

========= Nếu hiểu theo cách : setState => re-render
tuy nhiên, react sẽ 'batching update', tức là gộp lại update 1 lần
(về batching update sẽ đề cập sau)

#29.1 Bài tập về Thay Đổi State Theo Thời Gian

```
======
import { useState } from 'react';

export default function Counter() {
  const [number, setNumber] = useState(0);

  return (
    <>
      <h1>{number}</h1>
      <button onClick={() => {
        setNumber(number + 5);
        alert(number);
      }}>+5</button>
    </>
  )
}

==================
```

tương tự:

```
 <>
      <h1>{number}</h1>
      <button onClick={() => {
        setNumber(number + 5);
        setTimeout(() => {
          alert(number);
        }, 3000);
      }}>+5</button>
    </>

=======Fix timer với snapshot
import { useEffect, useState } from "react";

export function App() {
  const [count, setCount] = useState(10);

  useEffect(() => {
```

```
    setInterval(() => {
      console.log(">>> run timeout");
      setCount(count - 1);
    }, 1000);
  }, []);
  return <div>count = {count}</div>;
}
```

#29.2 State Overtime - Thay Đổi Theo Thời Gian

========== alert trả ra kết quả 0 vì:
- B1: Tính toán các function handler (truyền vào giá trị hiện tại count = 0)
- B2: React tính toán new state (tăng count = 1)
- B3: tạo ra 1 bản snapshot (copy) của state hiện tại (đang = 1)
- B4: Chèn giá trị này vào JSX
- B5: Thực hiện re-render (vẽ lại giao diện)

Tại bước 4, giá trị của count hiển thị ra màn hình sẽ là 1, tuy nhiên, giá trị alert sẽ là 0 (lý do do bước 1)

=> lỗi là do React tạo ra snapshot

#30. Get Next State với setState

Với react class, chúng ta có hàm setState, đồng thời, sử dụng callback để lấy nextState
ví dụ:

```
  //init state: count = 0:
  this.setState(
    { count: this.state.count + 1},
    () => {
      console.log(">>> check count = ", this.state.count); //count = 1
    });
```

Với React Hook, chúng ta không có callback của setState(), tuy nhiên, chúng ta có thể tính toán nextState :)

```
setNumber(number + 1) //ko trả ra nextState
---
setNumber(n => n + 1) // nói với React 'tính toán giá trị của state',
thay vì đơn giản là 'thay thế - replace với snapshot của state'

===========
ví dụ về click button
export default function Counter() {
  const [number, setNumber] = useState(0);

  return (
    <>
      <h1>{number}</h1>
      <button onClick={() => {
        setNumber(n => n + 1);
        setNumber(n => n + 1);
        setNumber(n => n + 1);
      }}>+3</button>
    </>
  )
}

=> React tạo queue để update
```

Queued update	n	returns
n => n + 1	0	0 + 1 = 1
n => n + 1	1	1 + 1 = 2
n => n + 1	2	2 + 1 = 3

fixing timer:

```
 useEffect(() => {
   setInterval(() => {
     console.log(">>> run timeout");
     setCount(count => count - 1);
   }, 1000);
 }, []);
```

#31. Queueing a Series of State Updates - Đợi Để Update 'Hàng Loạt'
'batching' ~ bulk create/update/delete...

React 18 'batching' updates:

Ví dụ về 'batching' updates 'inside event handler': (với react <18)
https://github.com/reactwg/react-18/discussions/21

- Mặc định, React 18 hỗ trợ 'batching updates' out-of-the-box (xài thôi :)

Với 'batching update', React tối ưu hóa hiệu năng hơn, tránh render không cần thiết.

- Cách React 'batching' update:
setNumber(number + 1);=> đưa vào queue
setNumber(number + 1);=> đưa vào queue
setNumber(number + 1);=> đưa vào queue

=> React gộp tất cả hàm setState (setter) để cập nhập 1 lần

#32. Don't mutate the state.

State của React có thể lưu trữ giá trị của Javascript (number, string, array, object...)

Mutate state directly ? (Modify/Change)

```
import { useState } from "react";

export function App() {
  const [info, setInfo] = useState({
    name: "abc",
    age: 30
  });

  const handleModify = () => {
    // info.name = "modify";
    // info.age = "35";

    setInfo({
      name: "modify",
      age: 35
    });
  };

  const handleDoSth = () => {
    console.log(">>> do sth with info = ", info);
  };

  return (
    <div>
      'Hello world!'
      <div>userInfor: {JSON.stringify(info)}</div>
      <button onClick={() => handleModify()}>Modify state</button>
      <br /> <br />
      <button onClick={() => handleDoSth()}> Do sth </button>
    </div>
  );
}
```

#33.1 Update object

```
const Person = () => {
  const [person, setPerson] = useState({
    username: '',
    email: '',
    password: ''
  })

  const handleOnChangeUsername = (event) => {
    // person.username = event.target.value;
    setPerson({ username: event.target.value })
  }

  return (
    <div>
      <div className='input-group'>
        <label>Username</label>
        <input
          type={'text'}
          value={person.username}
          onChange={(event) => handleOnChangeUsername(event)}
        // onChange={handleOnChangeUsername}
        />
      </div>
      <div className='input-group'>
        <label>Email</label>
        <input type={'email'} />
      </div>
      <div className='input-group'>
        <label>Password</label>
        <input type={'password'} />
      </div>
    </div>
  )
}

//lưu ý về merge object với React Hook
```

#33.2. Copying objects with the spread syntax

Using a single event handler for multiple fields
=> cần truyền thêm name cho event

Lưu ý khi sử dụng spread syntax: shallow copy

#34. Write concise update logic with Immer

Khi state React 'deeply nested - gồm nhiều lớp object lồng nhau', dễ nhất để giải quyết vấn đề là tạo ra
bản sao của state và sửa đổi nó (tương tự _.cloneDeep hoặc JSON.parse/JSON.stringify)
- Dùng Immer để mutate object (deep copy)
- Khác với các các truyền thống (spread syntax..., object.assign, lodash.clone), Immer không 'overwrite - ghi đè' lại state cũ, mà tạo ra 1 state copy hoàn toàn mới.

Cài đặt immer hook: **npm install --save-exact use-immer@0.7.0 immer@9.0.15**

Cách sử dụng immer hook:

import { useImmer } from "use-immer";
=> thay vì useState của React

const [state, setState] = useImmer(initState);

với setter, có 2 cú pháp:
- truyền vào giá trị bình thường giống setter của useState (nếu không mutate state).
vd: setState(20)

- truyền vào 1 function có thể muate:
setState(draft => {
//mutate here
});

#35.1 Mutate simple object
//todo
#35.2 bài tập mutate question object
//todo
#35.3 mutate question object with immer
//todo

#36.1 Updating Arrays in State

1.Với JS, chúng ta có thể 'mutate' array, nhưng khi dùng state để lưu trữ array (với React), chúng ta nên đối xử như là 'immutable' data (dữ liệu không thể sửa đổi)

=> Tương tự như objects, khi chúng ta muốn update 1 array thì:
+ clone array thành 1 array mới
+ cập nhật state theo array mới (immutable)

2. Update arrays mà không cần sửa đổi (mutate):
Trong thế giới JS (datatype), array là object (kiểu dữ liệu object) -> dùng với React, nên sử dụng như là read-only.
+ không nên :
re-assign items trong 1 array.
vd: array = [1, 2, 3] => modify: array[0] = 'bla bla';

cũng như sử dụng push() (thêm item) và pop() (xóa item)

+ nên sử dụng các phương pháp tạo ra clone của array ban đầu
ví dụ: sử dụng filter() và map()

3. Adding to an array (thêm phần tử vào array)
- Không nên:
array.push(item) (thêm phần tử vào cuối mảng)
hoặc
array.unshift(item) (thêm phần tử vào đầu mảng)

- Nên: (có thể sử dụng spread syntax)
newArray = [...array, item] // thêm phần tử mới vào cuối mảng
hoặc
newArray = [item, ...array] // thêm phần tử mới vào đầu mảng

4. Removing from an array (xóa phần tử khỏi array)
Cách dễ nhất là sử dụng hàm filter()
https://developer.mozilla.org/en-US/docs/Web/JavaScript/Reference/Global_Objects/Array/filter

5. Transforming from an array (thay đổi cấu trúc của array)
sử dụng hàm map()

https://developer.mozilla.org/en-US/docs/Web/JavaScript/Reference/Global_Objects/Array/map
6. Update objects inside arrays
- ví dụ bugs khi modify

#36.2 Arrays with Objects
- Broken code with mutate: (example)

=> Fix with immer (mutate state directly)

Chapter 13: State Structure (NEW)

Thiết kế State sao cho nó chuẩn performance, chứ đừng dừng lại ở mức 'code cho chạy được'...

#37.1 Lựa chọn 'State Structure'

Một cấu trúc 'state' tốt, sẽ tạo ra điểm khác biệt giữa các component.
- Cấu trúc tốt: kiểm soát tốt state (modify) và tracking/debugging.
- Ngược lại: nguồn gốc sinh ra bugs :D

Nguyên tắc khi lựa chọn cấu trúc cho State:

1.Nhóm các state liên quan tới nhau thành group (Group related state) : nếu các biến states liên quan tới nhau,
và mỗi khi update, chúng ta update chúng => merge thành single state.

```
 const [A, setA] = useState(...)
 const [B, setB] = useState(...)
...update
setA(...)
setB(...)

... merge:  const [C, setC] = useState({A, B});
//update
setC({A, B})
```

2. Không lưu các biến mâu thuẫn với nhau trong cùng state (Avoid contradictions in state): dễ dẫn tới bugs => avoid

3. Không lưu 'các state không cần thiết' (Avoid redundant state): Nếu 1 biến (biến X) có thể tính toán được từ các state hiện có,
thì không cần thiết phải lưu biến (X) vào trong state.

4. Không lưu 'trùng state' (avoid duplication instate): khi data của các biến state giống nhau :D

5. Không dùng 'deeply nested state' (state lồng nhiều lớp với Object): sẽ 'không thuận lợi' khi update 1 biến nằm sâu trong object

==========

#37.2 Nhóm các state liên quan tới nhau

Group related state :

Đôi khi chúng ta phân vân giữa việc tạo state như thế nào cho nó hợp lý, ví du:
const [name, setName] = useState('');
const [age, setAge] = useState('');

hoặc
const [person, setPerson] = useState({ name: '', age: '' });

- Khi gom (nhóm) các state thành 1 object/array thì:
+ ưu điểm: sử dụng ít biến
các biến nào liên quan tới nhau thì nhóm lại => tường minh
+ nhược điểm: khi update 1 fields, sẽ cần copy lại fields trước (vì react Hook không tự động merge)

ví dụ:
setPerson({ ...person, name: 'newName' })

ví dụ về sync state khi nhóm lại:
https://codesandbox.io/s/sj26ig?file=/App.js&from-sandpack=true

#37.3 Không lưu các biến mâu thuẫn với nhau

Avoid contradictions in state :
bad: https://codesandbox.io/s/vjyj5v?file=%2FApp.js&from-sandpack=true

good: https://codesandbox.io/s/xcr4gi?file=%2FApp.js&from-sandpack=true

#37.4 Giản lược các biến không cần thiết

Avoid redundant state
bad: https://codesandbox.io/s/jcpgso?file=%2FApp.js&from-sandpack=true

good: https://codesandbox.io/s/8ryt1w?file=%2FApp.js&from-sandpack=true

#37.5 Hạn chế việc lưu trùng data trong state

Avoid duplication in state:
https://codesandbox.io/s/jcdoji?file=%2FApp.js&from-sandpack=true

https://codesandbox.io/s/ppywq7?file=%2FApp.js&from-sandpack=true

https://codesandbox.io/s/by1h3c?file=%2FApp.js&from-sandpack=true

#37.6 Hạn chế sử dụng các object/array lồng nhau nhiều lớp

Avoid deeply nested state

#38.1 Extracting State Logic into a Reducer

- Component với nhiều state:
 + Code nhiều (khó maintainer)
 + Event handler (code state logic + update state)

- Giảm độ phức tạp của component
- Tất cả logic (liên quan tới state) sẽ để tại một nơi
- Move outside component

=> call 'reducer'

=> move from useState to useReducer

==========
#38.2 Reducers (assume understanding Redux)

Chuyển từ useState sang useReducer trong 3 bước:
B1: Thay vì setState trực tiếp, => dispatch actions

B2: viết reducer function

B3: Sử dụng reducer (hook) trong component (lấy data từ reducer để sử dụng)

Các keywords:
- Reducer (giản lược hóa): nơi tách code logic khỏi component (xử lý tại đây)
- Dispatch (send)/action: sử dụng trong component, báo hiệu cho reducer biết, nó 'cần làm gì' với 'action'
được gửi đi
- Action (hành động): id của 'event handler'. mỗi 1 tương tác của người dùng, chính là 1 hành động.
với 1 hành động, state sẽ thay đổi => dispatch (actions) === send (actions) from component to reducer

#39.1 Normalizing State Shape (Giản Lược Hóa State - Boost Performance)

Inspired by: https://redux.js.org/usage/structuring-reducers/normalizing-state-shape

1. Vấn đề gặp phải với nested objects
- Mutate data (kể cả khi sử dụng immer) sẽ cần cẩn thận, nếu không sẽ dẫn tới mutate sai object => bugs
- Khi update 1 fields, dẫn tới update cả object ban đầu
(update state => render toàn bộ DOM chỉ với update 1 field duy nhất)

- Object nested nhiều lớp, và chứa nhiều data => khi cần update/delete 1 filed/child
=> hiệu năng không cao (slow)

=> tìm cách giản lược hóa object, chia nhỏ data =>
=> flattening object
=> normalized data

2. Designing a Normalized State
State Before:
//todo

State After:
//todo

3. Nguyên tắc khi Normalize Data
- Mỗi 'loại data' là 1 đối tượng, ứng với 1 'table'
- Mỗi 'data table' sẽ lưu trữ 1 loại object, với key là id

Tác dụng: giải quyết bài toán về nested objects hiệu quả hơn, đặc biệt khi data lớn
ví dụ: object từ hàng trăm và hàng ngàn phần tử (size)

#39.2 Tích hợp thư viện Normalize

npm install --save-exact normalizr@3.6.2

github: https://github.com/paularmstrong/normalizr

docs: https://github.com/paularmstrong/normalizr/tree/master/docs

#39.3 Bài tập normalize

Todo...

#39.4 Tối ưu hóa với immer + normalize question

Todo...

#40. What's next

- Thực hành chuyên sâu với bài test Fresher: https://hoidanit.com.vn/khoa-hoc/giai-ma-trinh-do-bai-test-fresher-react-640beb7df7099c369b3bc695.html

- Học react với Typescript: https://hoidanit.com.vn/khoa-hoc/react-pro-typescript-thuc-hanh-du-an-portfolio-6458f8f188a9e90d1174e47d.html

- Học react với Framework Next.js: https://hoidanit.com.vn/khoa-hoc/react-pro-max-lam-chu-toan-dien-reactjs-hien-dai-65198100e6bafa8caad417a6.html

- Học kiến thức Backend:
https://hoidanit.com.vn/khoa-hoc/backend-restful-server-voi-nodejs-va-express-sql-mongodb-640b539cfe283eefef939870.html

https://hoidanit.com.vn/khoa-hoc/nestjs-voi-typescript-mongodb-sieu-de-64686ec6fb456bbb90663dd6.html

Lời Kết

Như vậy là chúng ta đã cùng nhau trải qua hơn 190+ videos về cách luyện mindset (cách tư duy) với React.

Tất cả các kiến thức mình chia sẻ, đều **được lấy từ kinh nghiệm đi làm của mình và…**

trang tài liệu của React: https://react.dev/

Dĩ nhiên rằng, trong quá trình quá trình thực hiện khóa học này, mình sẽ không thể tránh khỏi những sai sót (vì nếu không có sai số thì mình làm member của team React rồi :v).

Vì vậy, nếu thấy sai sót, các bạn cứ thoải mái đóng góp qua Fanpage Hỏi Dân IT nhé.
https://www.facebook.com/askITwithERIC

Sau khóa học này, sẽ là update và thêm mới các khóa học tại https://hoidanit.com.vn/

Hẹn gặp lại các bạn ở các khóa học tiếp theo ….
Hỏi Dân IT