// Post man

1. 4 hành động thêm, sữa , xoá ,dữ liệu gọi là ( CRUD)

a. C: Create(tạo) 🡪 post

b. Read (đọc website, lấy dữ liệu) -> get

c. U: Update ( cập nhật ) 🡪 put/patch

d. D : Deletet( xoá) 🡪 delete

-post , get, put/patch, delete : là các phương thức HTTP( HTTP HTTP protocol)

2. trong postman :

Sửa ( put / patch ) và xoá (delete) phải truyền vào link host / chỉ mục của id

3. fetch

a. mặc định trong fetch tự định nghĩa phương thức get

a1. Thằng get lấy dữ liệu về nên không cần truyền dữ liệu gì đi, còn lại phải có dữ liệu gữi đi

//////////////////////////////////////////////////////////////////////////////

ECMAscript6

4. let , var, const

a. let , const ngoài block mẹ thì không còn ảnh hưởng \

b. trong lớp block , một let( hoặc const) được đặt nhiều lần thì ưu tiên cập nhật giá trị tại block bên trong nhất

c.chỉ có mỗi var ( được hỗ trợ hosting)

d.let & const

d1> ex:

const a =1; a= 4; console.log(a) 🡪 a = 1;

ex2:

*const* course = {

    name : 'javascript'

}

course.name = 'PHP'

kết quả trả ra là PHP 🡪 khi nào a = cái gì đó mới không đỗi thôi , còn thuộc tính vẫn có thể đỗi được

f. khi nào dùng biến có gán lại thì dùng let

5. expresstion function 🡪 arrow function

a.(tham só truyền vào)=> {return ….}

b.nếu không có {} thì đăng sau => là gì thì nó return về cái đó

b1. Const sum = (a,b) => a+b;

b2. Nếu return về mọt object thì phải nhóm nó lại

const sum (a,b) => ( {a: a, b : b } );

d. nếu hàm có một tham số truyền vào thì không cần ngoặc tròn

d1. Const log => console.log(log);

log('hello word');

e. trong object , một phương thức ( function) không dược biểu diễn bằng arrow function

f. arrow function không dùng làm object constructor đc

6. class

a.cung cấp cách viết object tách riêng thuộc tính và phương thức bóc tách ra chứ không để ngang cấp nhau như cách viết cũ

b.ex:

class Person  {

    constructor(name,age){

        this.name = name;

        this.age = age;

    }

    getName(){

            return this.name;

    }

}

7. element.name = element['name']

8. đặt gía trị mặc định ( default parameter values)

9. đặt thuộc tính mà tên biến = giá trị của biến && đồng thời biến đã được khai báo

+ đặt tên hàm bỏ function()

*var* name = 'javascript';

*var* price = '1000';

*var* course = {

    name,

    price,

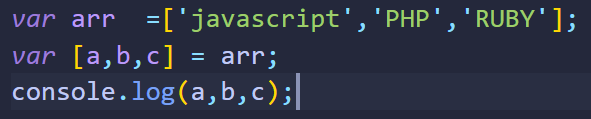
    getName(){

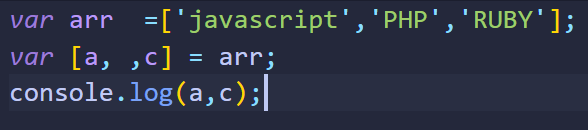
*return* name;

    }

}

10. Destructuring

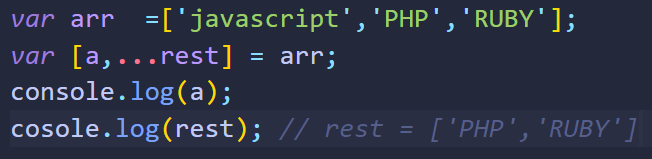
a.

b. 

c….REST

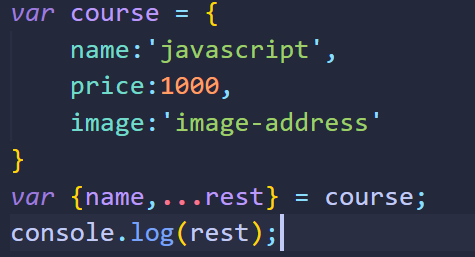
a. đối với arr

🡪 return về array



b.đối với object

🡪 Return object



11. newObject



A tuy đỗi tên biến nhưng nó là rest

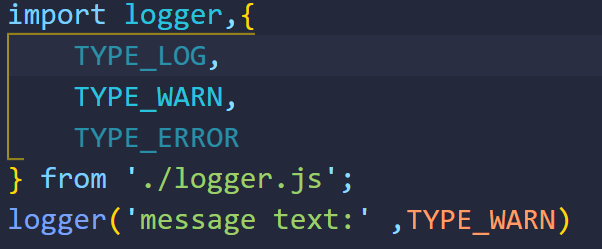
12. import \* as constants from "./constants.js"

a.cái file sử dụng import phải có attribute bên file html type = "module"

b. Lấy ra tất cả import ( không phải default ) trong file dẫn tới

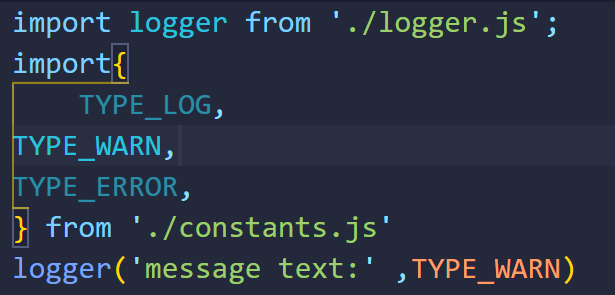
c. Một module chỉ export được một cái thôi

13. dùng destructuring để import ké theo default,

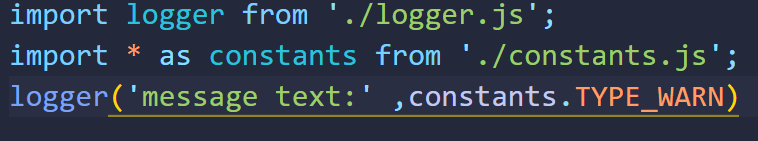


14. tách các export( không phải là default ra một file riêng)

a.

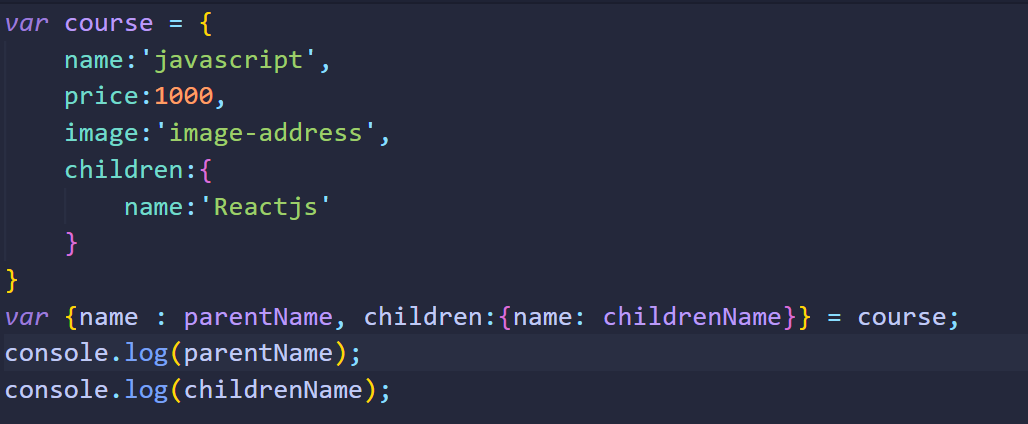


b.cách import thằng constants thành một object chứa các thuộc thuộc tính:



12. trường hợp đặt biệt : khi có name , và có object lại có name

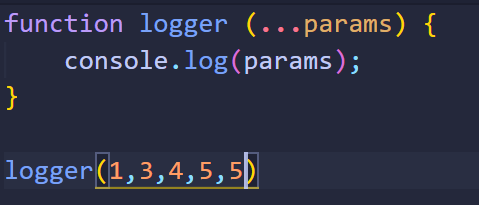
🡪 khai báo ten phụ cho biến name



13. dấu ba chấm : …

a. khi sử dụng với distructuring thì nó là toán tử rest( sử dụng để định nghĩa ra tham số )

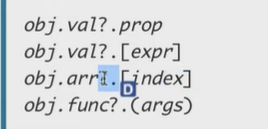
còn lại thì là spread



🡪 params = [1,3,4,5,5]

Cho dùng tên khác ( params ) nhưng vẫn là rest

14.Option chaning (?)



- Thêm vào thành phần không chắc chắn tồn tại