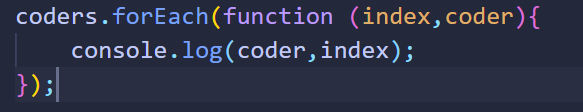
1.Đệ quy ;

A, loop 🡪 tốn CPU

Đệ quy 🡪 tốn ram

2. làm việc với mảng

a.forEach() : duyệt mảng



Coder🡪 lần lượt là các phần tử trong mảng

Index 🡪 lần lượt là số thứ tự trong mảng

b. lưu ý:

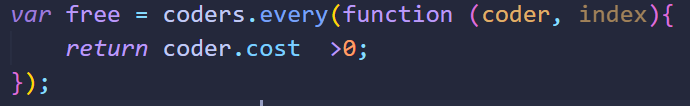
b1. Index có thể có hoặc không

b2. Nếu có index thì đầu vào bắt buộc theo thứ tự vậy

🡪 duyệt qua phần tử thật của mảng, không trả về cái gì

3. .every(): kiểm tra điều kiện toàn phần tử trong mảng

a. return về boolen

b. 

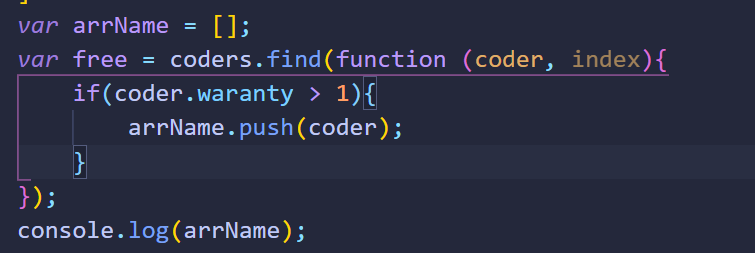
c. nhận hai đối số theo thứ tự, index( không bắt buộc có)

4. some(): y chang cấu trúc every() , khác là tìm phần tử nào thoả điều kiện đúng thì mới dừng , return về hàm boolen

5.find() : tìm thằng nào thoả điều kiện thì nhận thằng đó trong lượt duyệt đó

a. nếu tạo fucntion của hàm này thì nó chỉ return phần tử đúng đầu tiên ,

b. mỗi phần tử đúng thì được return trong lượt duyệt đó ( nếu muốn nhận được đủ phần tử , này để biết thêm cách hoạt động hàm này , 🡪 duyệt toàn bộ thì dùng hàm filter ở dưới)

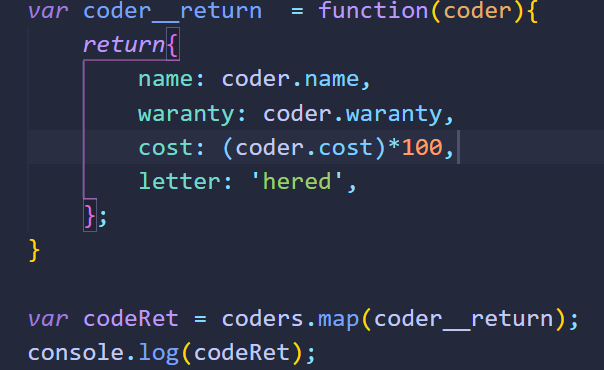


6. filter(): giống cách hoạt động hàm find(), khác -🡪 toàn bộ quá trình chạy của nó cuối cùng return về một array chứ các phần tử thoả điều kiện

a. Nó giống cách hoạt động của mục 4.b

7. map() return về mảng

a. khi truyền function trong map, return về cái gì thì tại lượt duyệt đó, phần tử đó bị thành đúng cái return đó



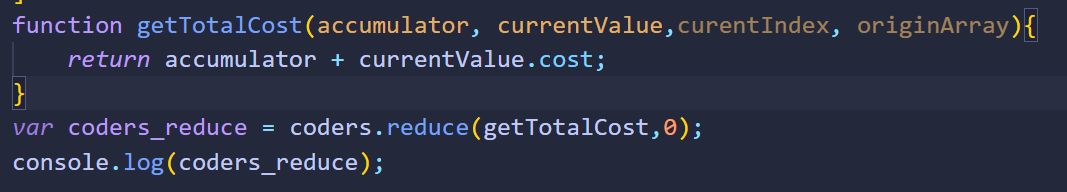
Lưu ý , trong hàm coder\_\_return : không dùng this để trỏ vào

b. có thể truyền tối đa ba đối số , ( course, index, originArray) cái thứ 3 là nó truyền ở dưới phần tử trong console phần tử đó lúc ban đầu

8. accumulator: lưu trữ

Initial value : giá trị khởi tạo

Reduce : dùng để return về một giá trị tính toán được nhờ việt duyệt qua tất cả các phần tử trong array

🡪

Truỳen vào hàm reduce function + giá trị tích luỹ ban đầu ( accumulator)

1. Trong function return về cái gì thì lượt duyệt ngay sau đó sẽ dùng nó để là accumulator,
2. Accumulator mặc định lấy phần tử đầu tiên trong mảng
3. CurrentValue:

C1; trường hợp không có giá trị khởi tạo:

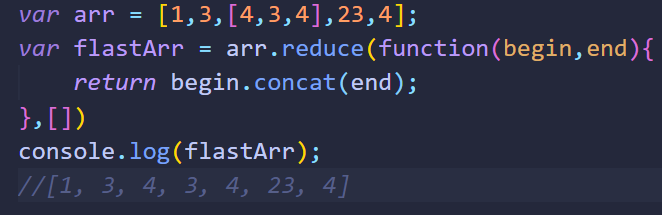
* Là phần tử đầu tiên trong mảng

C2. Trường hợp có giá trị khởi tạo:

🡪là phần tử kế tiếp phần tử đầu tiên

1. Có thể truyền ngay từ đầu accumulator là một mảng( ứng dụng làm kiểu cộng chuõi string thay vì cộng giá trị)
2. Tham số phía sau function trong reduce chỉ cần thiết khi mà phần tử đầu tiên không phải là cùng kiểu giá trị và cũng giá trị muốn tính toán 🡪 nếu mà cùng thì khỏi dùng tham số này : vd , nếu array là [1,2,3,4,4], tính sum mảng thì dùng reduce không cần truyền 0 vào vì mặc định phần tử đầu tiên cũng chính là cái mà cần xét rồi ,
3. Dù accumulator có là gì đi nữa thì nó chỉ đóng vai trò là currentValue ở vị trí bắt đầu ở đâu mà thôi.

9. kỹ thuật dùng reduce làm phẳng deep array

a. 

10.Number và Number object

a. covert : chuyển đỗi

🡪convert string 🡪 number : dùng Number(’99.99’) === 99;

Đây là chuyển đỗi thông thường , còn khi tạo var x = new Number(’99.99’)

* Đây lại là một object chứ ko phải là number , với rất nhiều thứ kèm tho
* <https://kipalog.com/posts/Lam-sao-de-convert-String-sang-Number-trong-JavaScript>

11.thuộc tính includes

A.dùng được cho cả String && Array

b/ return về boolen

c/ truyền vào hai đối số (‘từ cầnt ìm kiếm’ , vị trí bắt đầu tìm)

d. nếu vị trí bắt đầu tìm là số âm thì nó sẽ tính là = string.length ( hoặc array.length) +(số âm đó)

d1. Nếu mà abs(số âm truyền vào ) lớn hơn length của string ( array) dó thì nó sẽ tính là 0

12. typeof return về string

13. callback()

a. truyền một hàm vào một hàm

b. hàm con được gọi ra để thực hiện ngay trong hàm mẹ

14. lưu ý xương máu về array

Nếu giá trị muốn add vào array không thuần đơn giá trị ,thì phải dùng thuộc tính push để đẩy nó vào array , không được dùng gán(= )

15. empty elements of array:

a. dùng for in khử được tình trạng này