Các phương thức vòng lặp nâng cao MAP, Reduce

+ forEach(): duyệt qua từng phần tư của mảng

EX: let courses1 = [

    {id: 1,name: 'JS',coin: 250},

{id: 2,name: 'HTML, CSS',coin: 350},

{id: 3,name: 'Python',coin: 0},

{id: 4,name: 'Ruby',coin: 150},

{id: 5,name: 'Dart',coin: 400}

];

courses1.forEach(function(course, index) {

    console.log(index, course)

})

+ every(): dùng để kiểm tra tất cả các phần tử thuộc một mảng phải thỏa mãn 1 điều kiện gì đó

EX: let courses1 = [

    {id: 1,name: 'JS',coin: 250},

    {id: 2,name: 'HTML, CSS',coin: 350},

    {id: 3,name: 'Python',coin: 0},

    {id: 4,name: 'Ruby',coin: 150},

    {id: 5,name: 'Dart',coin: 400}

];

let isfree1 = courses1.every(function(course, index) {

    return course.coin === 0;

})

console.log(isfree1); 🡺 false

+ some(): dùng để kiểm tra chỉ cần một phần tử phải thỏa mãn 1 điều kiện gì đó

EX: let courses1 = [

    {id: 1,name: 'JS',coin: 250},

    {id: 2,name: 'HTML, CSS',coin: 350},

    {id: 3,name: 'Python',coin: 0},

    {id: 4,name: 'Ruby',coin: 150},

    {id: 5,name: 'Dart',coin: 400}

];

let isfree1 = courses1.some(function(course, index) {

    return course.coin === 0;

})

console.log(isfree1);

+ find(): tìm kiếm phần tử nếu có thì trả về chính phần tử đó và chỉ trả về 1 phần tử duy nhất nếu không có thì trả về undefined

EX: let courses1 = [

    {id: 1,name: 'JS',coin: 250},

    {id: 2,name: 'HTML, CSS',coin: 350},

    {id: 3,name: 'Python',coin: 0},

    {id: 4,name: 'JS',coin: 150},

    {id: 5,name: 'Dart',coin: 400}

];

let isfree1 = courses1.find(function(course, index) {

    return course.name === 'JS';

})

+ fillter(): tìm kiếm phần tử nếu có thì trả về 1 mảng gồm tất cả các phần tử tìm được nếu không thì trả về undefined

EX: let courses1 = [

    {id: 1,name: 'JS',coin: 250},

    {id: 2,name: 'HTML, CSS',coin: 350},

    {id: 3,name: 'Python',coin: 0},

    {id: 4,name: 'JS',coin: 150},

    {id: 5,name: 'Dart',coin: 400}

];

let isfree1 = courses1.filter(function(course, index) {

    return course.name === 'JS';

}) 🡺 [ { id: 1, name: 'JS', coin: 250 }, { id: 4, name: 'JS', coin: 150 } ]

+ map(): Thường dùng để render giao diện

lặp hết phần tử mảng trả về phần tử bạn muốn trả

Gồm 3 tham số: value, index, originArray(mảng ban đầu)

EX: let courses1 = [

    {id: 1,name: 'JS',coin: 250},

    {id: 2,name: 'HTML, CSS',coin: 350},

    {id: 3,name: 'Python',coin: 0},

    {id: 4,name: 'Ruby',coin: 150},

    {id: 5,name: 'Dart',coin: 400}

];

let newcourse = courses1.map(function(value, index, originArray) {

    return { index: index + 1, id: value.id, name: `Khoa hoc ${value.name}`,

coin: value.coin, coinText: `Gia coin: ${value.coin}`,

originArray: originArray} });

console.log(newcourse)

+ reduce() : muốn nhận về 1 giá trị duy nhất sau khi tính toán trong 1 array

Gồm 4 tham số: + accumulator: giá trị khởi tạo

+ currentValue: phần tử hiện tại đang đứn

+ index: chỉ mục hiện tại

+ originArray: Mảng ban đầu

EX: let courses1 = [

    {id: 1,name: 'JS',coin: 250},

    {id: 2,name: 'HTML, CSS',coin: 350},

    {id: 3,name: 'Python',coin: 0},

    {id: 4,name: 'Ruby',coin: 150},

    {id: 5,name: 'Dart',coin: 400}

];

let totolCoin = courses1.reduce(function (accumulator, currentValue, currentIndex, originArray) {

    return currentValue.id + accumulator;

}, 0); console.log(totolCoin)

EX: let topicss = [

    {topic: "Front-end",courses: [{id: 1,title: "HTML, CSS"},{id: 2,title: "Javascript"}]},

    {topic: "Back-end",courses: [{id: 1,title: "Java"},{id: 2,title: "PHP"}]}

];

let newArray = topicss.reduce(function(a, b) {

    return a.concat(b.courses);

}, []);

EX: function arrToObj(array1) {

    let results = array1.reduce(function(obj, value) {

        obj[value[0]] = value[1];

        return obj;

    }, {});

    return results;

}

var array= [

    ['name', 'Sơn Đặng'],

    ['age', 18],

]; console.log(arrToObj(array)); // { name: 'Sơn Đặng', age: 18 }

+ Tự tạo phương thực Reduce của riêng

EX: Array.prototype.reduce2 = function(callback, result) {

    let i=0;

    if(*arguments*.length < 2) {

        i = 1;

        result = *this*[0]

    }

    for(;i<*this*.length;i++) {

        result = callback(result, *this*[i], i, *this*)

    }

    return result;

}

const numbers = [1,2,3,4,5];

const Result = numbers.reduce2((total, number) => {

    return total + number

}, 10)

console.log(Result);