**Bài 1: Thuật ngữ term :** dùng nhiều trong toán học.

* Các thuật ngữ trong lập trình: Biến, kiểu dữ liệu, mảng, Hàm…

**Bài 2: Biến:**

Chương trình chạy lưu trữ tạm thời trên Ram. Kết thức chương trình thì mất đi.

Đặt tên biến chữ đầu biến viết thường.

Var a = 1 // lưu biến a vào thanh ram.

//var là key word

**Bài 3: kiểu dữ liệu:**

* Prinitive type (kiểu nguyên thủy): number, string,boolean.
* Special types ( kiểu đặc biệt ): underfined, null.
* Reference type ( loại tham chiêu ): Array, object.

Number: var a= 100;

String: var str = “xin chao”;

Boolean ( true, fasle ) ẩn hiện : var isHidden = fasle;

Kiểu undefined: var a; khai báo không gán kết quả trả về thì giá trị trả về undefined.

Null: a = null // a là giá trị rỗng;

**Bài 4: kiểu Object:** dùng để mô tả một đối tượng nào đấy.(con người: màu tóc, màu da)

Var a = {

Key = value;

}

GET: truy xuất. a.key || A.[‘key’];

SET: gán giá trị lại. a.key = “teo”;

**Delete a.key:** xóa môt thuộc tính của đối tượng.

**Bai 5: Mảng Array**: element : các phần tử trong mảng.

Array thường dùng để lưu trữ các phần từ giống nhau.

* Index : chỉ số của các phần tử trong mảng;
* Length: độ dài của mảng;

Var evenNumber = [2,4,6];

Chỉ số index bắt đầu từ 0;

Chiều dài của mảng từ 1;

GET: evenNumber[1];//4

SET: đổi vị trí evenNumber[1] = <tên đối tượng>

**Bài 6 phép toán số học:**

Var a =1;

* 1) ++ ,--: ++ a // thay đổi giá trị của a ngay lúc đấy a =2 ; a++ giá trị trước khi tăng 1
* 2) \* / %
* 3) + -

**Bài 7 : phép gán:** bình thường thì thực hiện từ trái qua phải nhưng phép gán thì thực hiện từ phải sang trái.

* = : a = 1;
* +=

**Bài 8: function()** thực hiện một tác vụ, chức năng nào đấy nào đấy

Function tenHam(input1,input2,...)

{

//code

Return something ; trả về một cái gì đấy

}

Những gì những gĩ trong function dùng xong nó sẽ hủy đi.

Call a function(input1, input2,...); // gọi function(); truyền các tham số vào phải là giá trị thực.

**Return:** biến hàm thành một giá trị. Khai báo ra, muốn sử dụng thì phải console.log() mới hiện ra. Thực hiện một quy trình và nó trả về giá trị.

* Dừng luôn tại đấy. đánh dấu sự kết thúc của hàm đây.

**Anonymous == không tên.:** Khai báo biến cho hàm đấy. sử dụng một 2 lần thì nen sử dụng nonymous function.

**Clousure function:** không gian đóng cho cái function đấy. sử dụng biến bên ngoài function đấy.

* Có thể thay sử dụng, thay đổi biến ở 3 phạm vi, thứ nhất toàn cục, thứ 2 biến cục bộ của cha, thứ 3 biến cục bộ của chính nó.

**Bài 9: object method:** phương thức đối tượng.

Bark : Function(){ // function bây giờ được gọi là method.

Console.log(‘Meo meo’);

}// **function không có tên được gọi là anonymous function**

This: là keyword dùng trong ngữ cảnh của method đại diện chính Object trong trường hợp này.

Get function trong object: myDog.bark();

**Bài 10 phép so sánh**

a = [1, 2]; b = [1, 2]; là bạn đang gán a và b đến 2 đối tượng khác nhau (được đặt ở các vùng nhớ khác nhau trong bộ nhớ, mặc dù hai đối tượng này có cùng giá trị).

Phép === xảy ra nếu a = b; và ngược lại hoặc a = c; b = c;…

var a = {

foo: 'bar'

};

var b = {

foo: 'bar'

};

console.log(a === b);

// vì 2 object a và b không tham chiếu đến cùng một vị trí trong bộ nhớ.

**Bài 11: Vòng lặp for:**

Dùng để lặp qua từng phần tử.

**Bài 13: Vòng lặp for .. of; for .. in:**

  var toneTeo = {

            fullName: 'Thích Tên Tèo',

            age: 22,

            gender: true

        };

        var employees = [

            { name: 'teo', age:22 },

            { name: 'chuoi', age:24 },

            { name: 'hue', age:21 }

        ];

        for (var employee of employees) {

            console.log(employee.name);

        }//lặp các phần từ object trong mảng

        for (var key in toneTeo) {

            console.log(toneTeo[key]);

        }//lặp các các thuộc tính theo key trong 1 object

**Bài 13: Array methods: cũng là một objbect:**

-Phương thức thêm 1 phần tử vào phía sau của mảng push: arr.push( giá trị ). Có thể thêm nhiều phần tử.

-Phương thức thêm 1 phần tử vòa phía đầu của mảng arr.unshift( giá trị ). Có thể thêm nhiều phần tử.

-Phương thức remove xóa đi phần tử cuối cùng của mảng arr.pop().

-Phương thức array.splice(vitri bắt đầu, số phần tử muốn cắt): methods: xóa bao nhiêu phần tử làm thay đổi mảng gốc. Phương thức này vừa xóa lại vừa thêm:

Arr.splice(vị trí ,số phần tử muốn xóa, ...): ... thêm phần tư thêm vào vị trí xóa.

-Phương thức array.slice (vitri bắt đầu, số phần tử muốn cắt): methods: tạo ra mảng mới không làm thay đổi mảng gốc

-Phương thức remove xóa đi phần tử đầu tiên của mảng arr.shift().

-Phương thức indexOf kiểm tra xem phần tử mình cần tìm kiếm có trong mảng hay không, return giá trị đầu tiên xuất hiện tron mảng arr.indexOf(giá trị) // trả về vị trí của phần tử xuất hiện đầu tiên. Nếu nó trả về -1 thì phần tử đấy không tồn tại trong mảng.

-Phương thức lastIndexOf return về phần tử xuất hiện cuối cùng trong mảng. Ngược lại với indexOf.

🡺Nếu như indexOf và lastIndexOf trả về cùng 1 giá trị giống nhau => Giá trị đó tồn tại trong mảng.

Truy xuất mảng tới vị trí không hợp lệ => undefined.

Gán giá trị cho vị trí không hợp lệ cũng sinh ra phần tử trong mảng vào cuối mảng, mình có thể thêm trực tiếp bằng cách 2 này: arr[ arr.length ]=’test’;

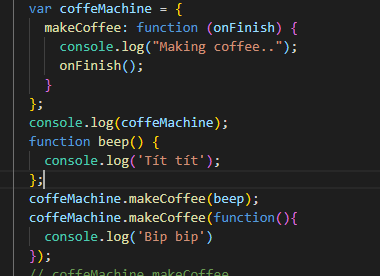
- Dùng Array.from(collection): chuyển đổi một collection thành một mảng.

- Dùng Array.join(‘ ’): chuyển đổi một mảng thành 1 chuôi.

**Bài 13: Dùng fuction như tham số( Callblack )**

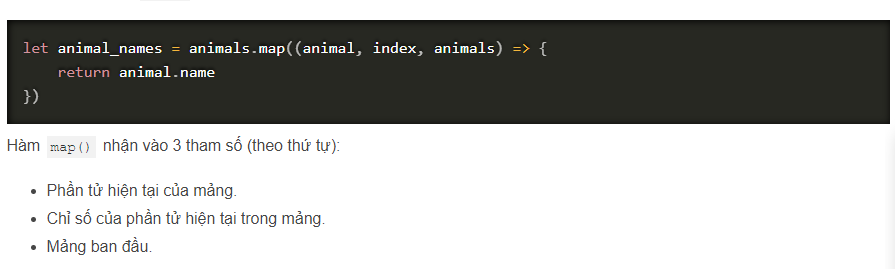
Tryền vào hàm tên biến đại diện cho một hàm.

Khái báo hàm truyền nó vô một phương thức của làm tham số.

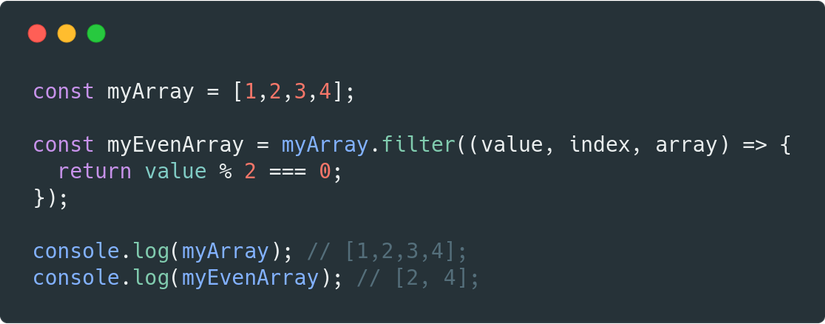
**Bài 14: Array.map.**

Phương thức của Array biến đôi từng thành phần trong array cũ thành các phần tử mới theo một quy tắc được định nghĩa của function().



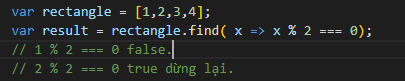
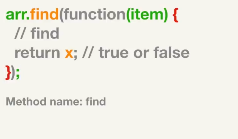
**Bài 14: Array.filter:**

Phần tử mảng đi qua function() lọc trả ra một mảng phù hợp với yêu cầu của function đưa ra**.**

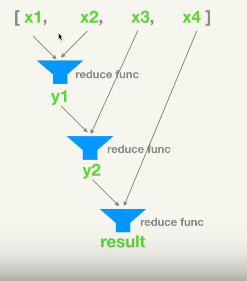
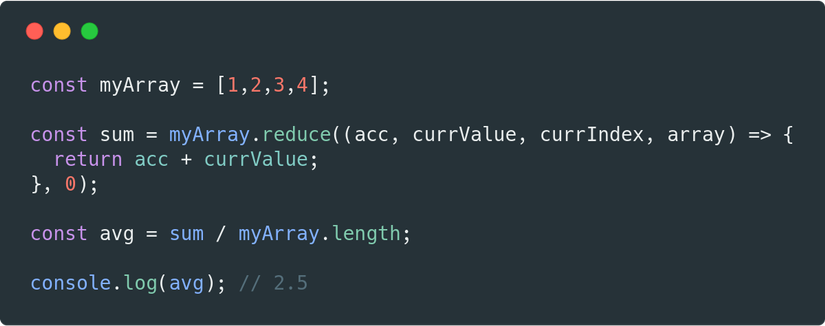
****

**Bài 15: Array.find:**

Lọc phần tử mảng đi qua function() trả về giá trị true hoặc fasle; lọc được kết quả phù hợp thì sẽ dừng lọc luôn.

**Bài 16: Array.reduce:**

( nó tích luỹ tất cả các giá trị trả về . Giá trị của nó là sự tích luỹ của các tích luỹ trả về trước đó ) giá trị hiện tại , index hiện tại và cuối cùng là toàn bộ mảng .

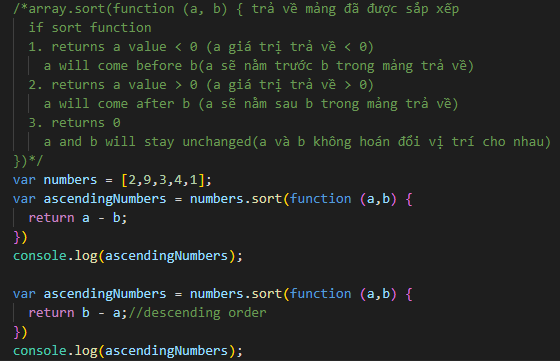
 Reduce có nhận vào 2 tham số 1 function(), tham số 2 là init : Nếu như cung cấp cho nó tham số thứ 2 thì trong vòng lặp đầu tiên nó sẽ dùng ở vị trí item1 nếu không có init thì nó sẽ lấy item1, item2.

Tham số init: phụ thuộc vào giá trị mà mình muốn ví dụ:

* Muốn là số thì init truyền vào 0;

**Bài 17: Sort an array:**

Sắp xếp các phần tử trong mảng.



String.sort((a, b) = a > b ) // sort A -> Z;

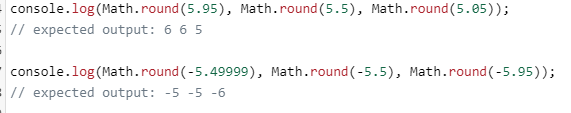
Cách đơn giản để sắp xếp sort tăng lên:

Numbers.sort().

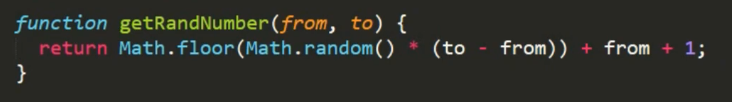
**Bài 18 : Math Object : toán học.**

**Thuộc Tính:**

* Properties: Math.PI : 3.14;
* Math.ceil(number): làm tròn số thập phân lên// 9,5 lên 10;
* Math.floor(number): làm tròn số thập phân xuống// 9.9 lên 9;
* Math.round(numder): làm tròn số nguyên gần nhất



* Math.max(x1,x2,….): trả về giá trị lớn nhất.
* Math.min(x1,x2,…): trả về giá trị nhỏ nhất.
* Math.random(): trả về 0 đến 1 số ngẫu nhiên.

Ví dụ: 

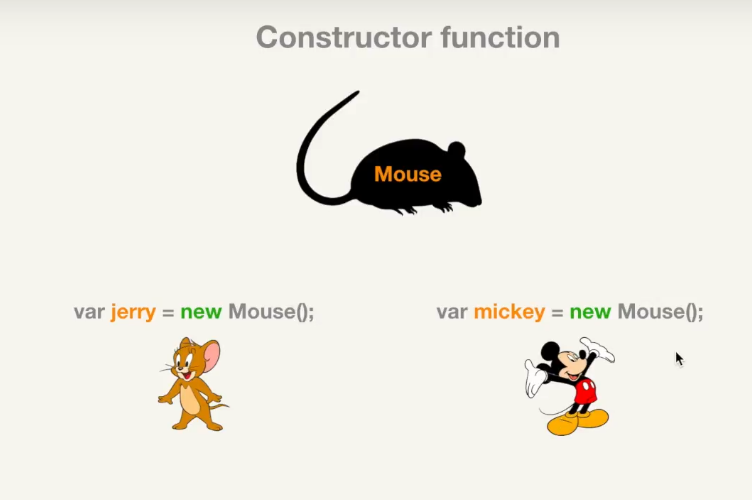
* Math.pow(x,y): trả về x mũ y.

**Bài 19 : the ‘new’ keyword**

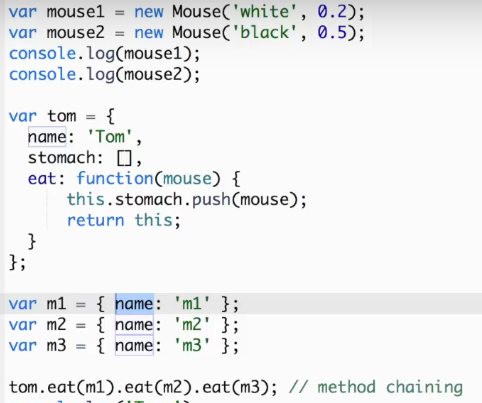
New: cách tạo Object bằng keyword new:

Constructor function: hàm xây dựng một cái gì đấy. như là một function nó có thể nhận vào tham số. Tất cả functiont trong javascript đều có thể gọi new được.

* Contructor: Dùng như nguyên mẫu , đối tượng được copy từ function này.



**Method chaning:** thêm return this; vào cuối phương thước thì có chuỗi phương thức



**Default parameters :**

function multiply(a, b = 1) {

return a \* b;

}

multiply(5, 2); // 10

multiply(5); // 5

multiply(5, undefined); // 5

**Kiểm tra key có trong object không:**



**Bài 9.1 Các phương thức cho string Object**

**string.toString**(). Ép kiểu sang chuỗi.

* .toUpperCase() : Viết hoa toàn bộ string.
* .toLowerCase() : Viết thường toàn bộ string.
* .indexOf(‘string start’, vi tri) :Tìm vị trí không tìm thấy trả về -1
* .lastIndexOf(‘string start’) : Tìm từ cuối lên.
* .split(‘)

**Kĩ thuật spread operator: coppy tạo thành đối tượng mới.**

|  |  |
| --- | --- |
| **Array** | **Object** |
|  |  |

**Làm việc với một ‘’chiếc hộp’’ Closure:**

Là một môi trường trong đó function có thể gọi được những biến ngoài function đó.

**Đệ quy:**

Là khi một function gọi đến chính nó đến khi không còn gọi được nữa.

**Object.key(obj):** trả về mảng các key object.





**Object.assign():**

Khi làm việc với nhiều object:

let o1 = { a: 21 };

let o2 = { b: 22 };

let o3 = { c: 24 };

const obj = Object.assign({},o1, o2, o3);

console.log(obj); //{a: 21, b: 22, c: 24}

Gộp các object với cùng properties:

let o1 = { a: 21, b: 22, c: 24 };

let o2 = { b: 25, c: 26 };

let o3 = { c: 27 };

let finalObj = Object.assign({}, o1, o2, o3);

console.log(finalObj);//{a: 21, b: 25, c: 27};

**Bài 20: Prototypes in JavaScript: ( KHUÔN MẪU )**

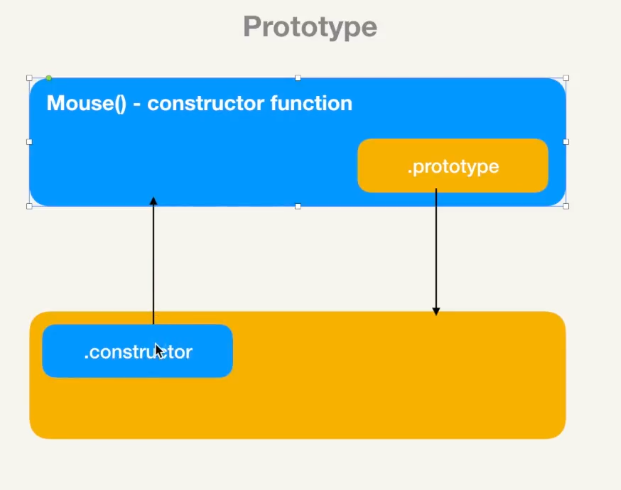
Object được tạo ra ví dụ là những cái bánh.

|  |  |
| --- | --- |
| **Contructor function** | **Normal function** |
| Được đi cùng với từ khóa NEW  Tạo ra các object mới.  Tên Hàm thường là một cái dánh từ.  Không thực hiện chức năng gì , không return gì luôn : thường khai báo một số thuộc tính. | Làm một công việc cụ thể nào đấy  Tên hàm thường là một động từ.  Chữ cái đầu viết thường. Vidu: nameClass |

Tất cả các function(Object) trong Javascript khi được khai báo đều có một **Prototype:**

**Trong Prototype:** Lại có một thuộc tính là .contructor. Hai thằng trỏ vào nhau.

console.log(Mouse.prototype.constructor === Mouse); // true



Prototypes là một Object được chia sẻ giữa tất cả cho những Object được tạo bằng từ khóa New của constructor.

Tại sao dùng Prototypes: Tiết kiệm bộ nhớ hơn, chỉ khai báo 1 lần, lưu một lần thôi.

* Nếu so sánh prototypes của 2 đối tượng sẽ trả vè một thằng thì sẽ trả về true, còn chúng ta tạo method trong constructor thì đối tượng sẽ khởi tạo nhiều lần, lưu nhiều lần.

**Bài 21 : Cài đặt NodeJS + Sublime Text trên windows.**

**Lệnh CMD:**

**- cd:** chuyển thư mục.

**+ cd ..** :trở về thư mục trước nó , cha của nó.

**+ cd users** :đi vô thư mục users.

**- cd C:\:** Để trở về ổ C.

**- dir:** Hiển thị tất cả các thư mục.

**- mkdir test:** tạo thư mục test.

**- echo > test.html :** tạo file test.html trong thư mục test.

**- del test.html :** xóa file test.html trong thư mục test.

**- rmdir test** : xóa thư mục test đi.

**- cls:** Xóa trắng.

**Bài 22: Node module system**

Node module : chia nhỏ chương trình ra.

Có nhiều constructor khó quản lý cần module.

Ví dụ: Có một file index,js.

* Tách file mới tên là mouse.js: tách hết constructor của và prototype sang, cuối có dong code : **module.exports = Mouse** // mô hình xuất khẩu. có nghĩa là file Mouse.js là một module nó exports ra contructor function này.
* File index,js ở đầu khai báo một biến :
* Var Mouse = require(‘./mouse’) // ./ nghĩa là trong cùng một thư mục với file index.js.

**Bài 23:Node built-in module:**

Node được xấy dựng bên trong module:

Những module được built sẵn khi bạn cài nodeJs:

* File system (hệ thống tập tin): Đọc và ghi file trong máy mình.

Module: **fs.readFileSync(path[,options]):** exports ra thực chất chỉ là một Oject thôi. Trong đó nó có một method gọi là readFileSync : truyền vào đường dẫn cho nó.

1). Tạo mới một thư mục , tao file index.js:

- var fs = require(‘fs’); // tên giống tên biến. không cần ./ nó tự tìm trong hệ thống.

- var text = fs.readFileSync(‘./song.txt‘,{encoding: ‘utf8’}); //truyền method để đọc file,

- console.log(text);// in kết quả

- fs.writeFileSync(‘./song-2.txt’,’I am trung kien’);// dùng method write file để viết file.

**Bài 24: npm (Node Package Manager)**

Ví dụ: Khi viết một app nào đấy thì nó có nhiều module nhỏ khác nhau:

* Có thể module các bạn tự viết.
* Có thể module của NodeJS.
* Có nhà các nhà phát triển khác developer khác.

**NPM :** Khi chúng ta cài NodeJs thì nó đi cùng nhau.

Module ở ngoài không được tích hợp trong nodejs , thì phải cài nó.

Ví dụ: readline-sync : khi các bạn chạy cmd nó hỏi một số câu hỏi rồi nhập vào.

* **1)npm init:** để khởi tạo Project này có dùng các module bên ngoài.

Tạo được file package.json. mô tả project.

* **2)npm install <tên module> --save:** cài module. –save để nó lưu trong file package.json này.

# Module: password-generator : tạo password;

**Bài 24: Json methods:**

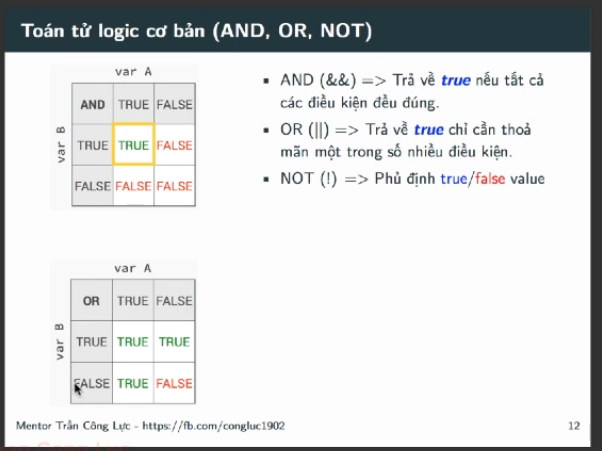
Object JSON:

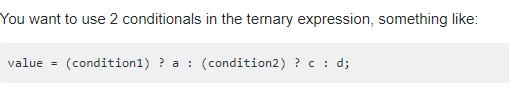
* Method 1: stringfy - Convert an Object to a JSON string.

JSON.stringfy(objName): sẽ in ra một string khác object là key có dấu “”; chuyển thành một string rồi thì có thể lưu nó vào file được để lưu nó lại.

* Method 2: parse – Convert a JSON string to an Object.

**Bài 25: if… else:**



Toán tử 3 ngôi: 

Toán tử 3 ngôi: <https://repl.it/@hoanthptvn/DaringSaltyText>

**Bài 26: Sync. Vs .Async**

**Sync:** Đợi nhau ( đồng bộ ) làm từng bước.

**Async:** Không đồng bộ khởi động một công việc trong thời gian trờ thì chúng ta làm việc khác.

Trong lập trình trong quá trình Read / Write file to fs:

Network: khi tải 1 trang web về nó tốn thời gián trao đổi giữa sever. Sẽ dùng Async.

Module: của nodeJs readFile : hàm bất đồng bộ

Module: <https://www.npmjs.com/package/axios>

Module : request.

Việc sử lý các tác vụ trong JS là bất đồng bộ các tác vụ sẽ không được thực hiện theo đúng trình tự như chúng ta nhìn code . Từ đó chúng ta sử dụng callback có thể điều khiển việc sử dụng chúng theo đúng ý mình muốn.

Bản chất js là ngôn ngữ bất đồng bộ.

**Javascript tổng quát** :

Có heap và stack:

Heap: là nơi cấp phát và giải phóng bộ nhớ.

**Bài 27: Callback hell**

Callback hell gọi file lồng nhau file được load , rồi đến hàm thực thi , giống hình kim tự tháp.



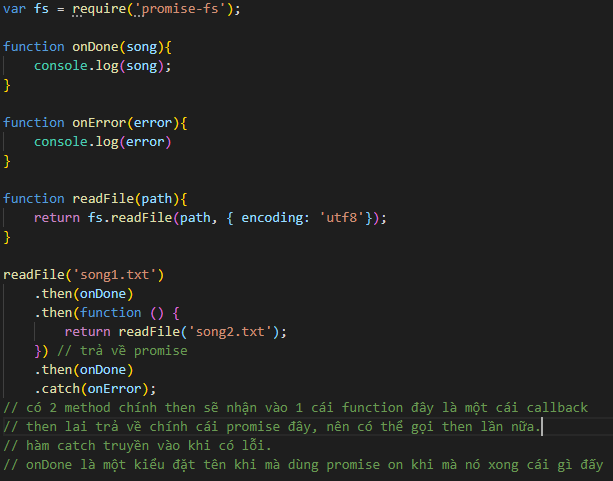
**Bài 28: Promise.**

Trong js có một Object nữa gọi là Promise là một constructor function:



**Promise:** là một cơ chế trong JS giúp bạn thực thi với các tác vụ bất động bộ mà không rơi vào callback hell hay pyramid of doom, tình trạng là hàm callback lồng nhau quá nhiều tâng

**Module hỗ trợ:** promise-fs



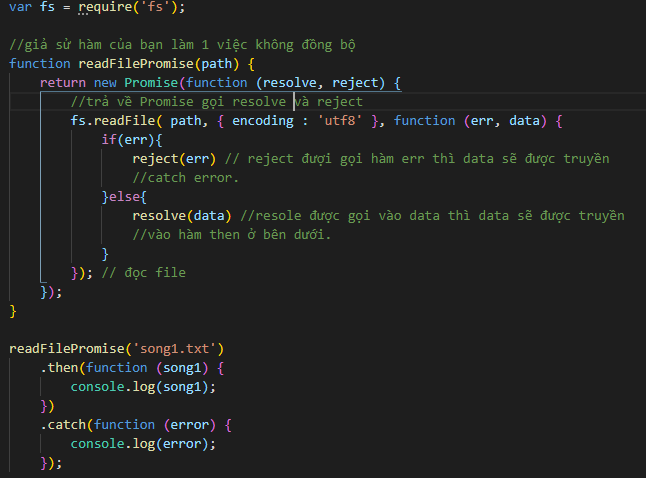
Then: sẽ trả về một cái callback().

Catch: sẽ trả về khi gặp lỗi.

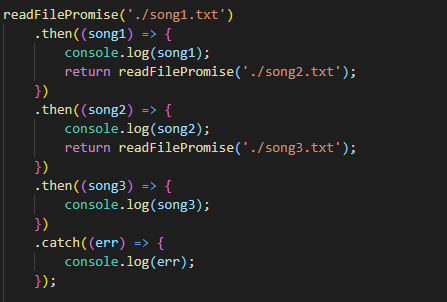
**Module Request:**

****

**Cách 2: using module nodejs.**

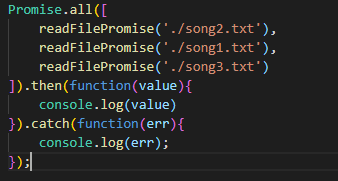


**Gọi nhiều file**



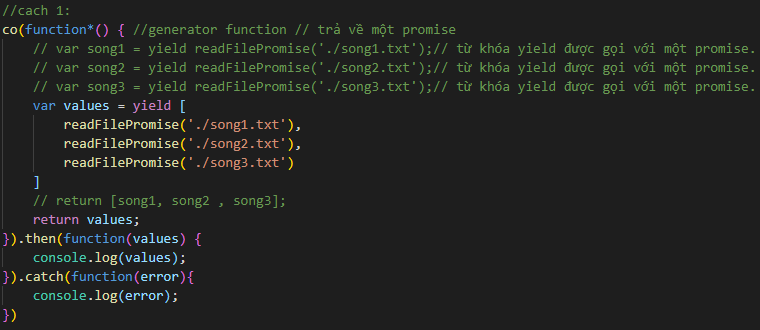
**Bai 29 : Promise.all**

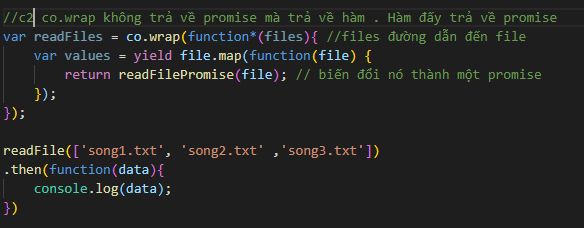
**Method promise.all:** Là một method của promise.

* Dùng để thu gon code của bạn.thành một array
* 

**Bài 30: Module co node**

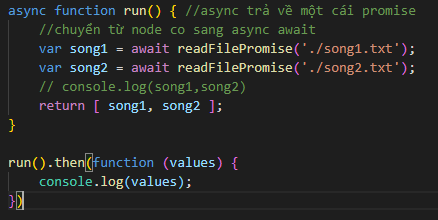
Module được phát triển bởi cộng đồng.





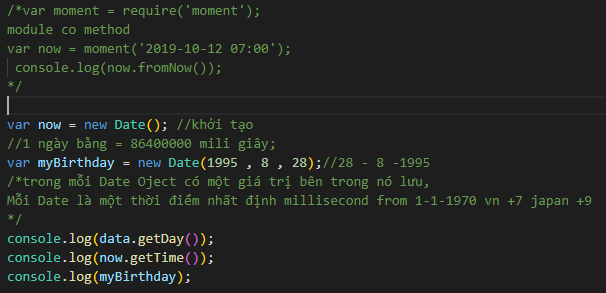
**Bai 30 : Async await:**

Keyword này tồn tại từ bản node 7.6 trở lên: Tương tự như node co



**Bai 31: Date object:**

Module: **moment** : Method fromNow() // khoảng cách giữa thời gian thực.



**Bai 31: Scope**

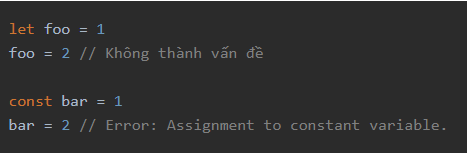
**Phạm vi hoạt động của một biến.**

Giữ phạm vi biến rộng quá:

1. **Global scope:** Không nằm trong hàm nào cả thuộc đối tượng Object window

* Có 2 cách hiển thị là console.log(x) || console.log(window.x); Phải khởi tạo giá trị ban đầu cho biến không thì biến đó có giá trị undifile.

1. **Local scope:** Biến khai báo trong hàm; chỉ sinh ra khi khai báo hàm và bị hủy đi khi hàm thực hiên xong.



**Bai 32: Database design(Basic)**