MỤC LỤC

[Bài Biến (variable) 1](#_Toc70450042)

[Bài Kiểu dữ liệu (data type) dynamic type 2](#_Toc70450043)

[2.1 Primitive types: 2](#_Toc70450044)

[2.2 Reference types 4](#_Toc70450045)

[Bài Arithmetic Operators (phép toán số học) 4](#_Toc70450046)

[Bài Comparison operators (phép so sánh) 5](#_Toc70450047)

[Bài Vòng lặp for (for loop) 6](#_Toc70450048)

[Bài Array Methods 7](#_Toc70450049)

[Bài Math object 11](#_Toc70450050)

[Bài ‘new’ keyword 12](#_Toc70450051)

[Bài Prototypes (Khuôn mẫu) 12](#_Toc70450052)

[Bài NPM node package manager 13](#_Toc70450053)

[Bài if … else và else if 13](#_Toc70450054)

[Bài Ternary operator 14](#_Toc70450055)

[Bài Sync. vs Async. (Đồng bộ và bất đồng bộ) 14](#_Toc70450056)

[Bài Callback Hell (nhiều tác vụ phụ thuộc vào nhau ) 14](#_Toc70450057)

[Bài Promise 14](#_Toc70450058)

[Bài timing event 15](#_Toc70450059)

[Bài scope 16](#_Toc70450060)

[Difference between NodeJS and Javascript 16](#_Toc70450061)

# **Bài Biến (variable)**

Biến trong javascript được lưu trữ trong bộ nhớ browser process (tiến trình trình duyệt) có nghĩa là được lưu trong phần RAM mà tình duyệt đang sử dụng

Khai báo biến

Example: var a = 1; //declare a variable a, assign value 1 to variable a

Để khai báo sử dụng **const, let, var**

\* Scrope là nơi biến có thể được truy cập

\* Hoisting là một cơ chế trong JS, nó giúp cho các biến và hàm khi được khai báo đều được chuyển lên đầu trước khi câu lệnh được thực hiện

- const: khai báo hằng số, giá trị không đổi.

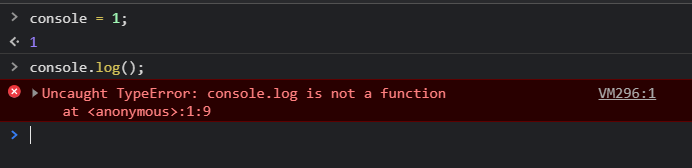
- let: biến chỉ truy cập trong block bao quanh nó được xác định bằng cặp {}

- var: biến truy cập ở scope function hoặc bên ngoài function, global.

\* Note: có thể không sử dụng var để khai báo nhưng không nên dùng bởi vì: nếu sử dụng var, biến được khai báo trong scope. Nếu không sử dụng từ khóa var, biến sẽ nổi lên qua các lớp của scope cho đến khi nó gặp một biến có tên đã cho hoặc đối tượng toàn cục.

Ex:

Không sử dụng var



# **Bài Kiểu dữ liệu (data type) dynamic type**

Dynamic typing: JS is a dynamic typing language. When you declare a variable, you do not need to specify what type this variable is. JS có thể suy ra kiểu dữ liệu dựa trên giá trị gán vào, có nghĩa là có thể gán bất kì kiểu dữ liệu nào cho biến đó.

## 2.1 Primitive types:

học thuộc (lưu giá trị trong stack)

- Number

Ex: Var a = 100;

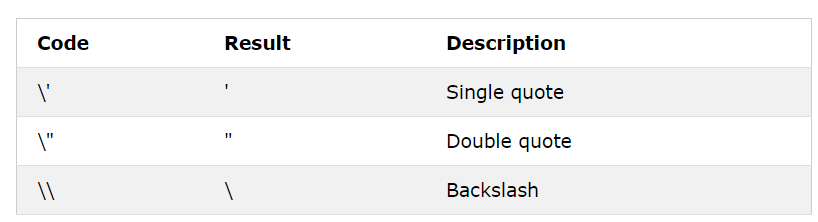
-String (immutable)

Ex: var str = ‘Hello’;

var str1 = “Hello”;

Nên dùng ‘’

**Case sử dụng backslash escape character (\)**

****

- Boolean: True, False

ex: var isHidden = false;

- Undefined

ex: var a;

- Null

ex var a = null;

- Symbol

**So sánh null vs undefined**

|  |  |
| --- | --- |
| **Undefined** | **Null** |
| Bất kì biến nào chưa được gán giá trị thì sẽ có giá trị **undefined**  Means a variable has been declared but not defined | Null is an empty or non-existent value.  Null must be assigned |

## 2.2 Reference types

lưu các giá trị trong heap và trỏ địa chỉ qua stack

- Object

ex: var myDog = {

name: ‘Mina’,

weight: 20, // key: value,

isAlive: false

}

console.log(myDog);

- Array

ex: var bunchOfDogs = [dog1, dog2, dog3];

# **Bài Arithmetic Operators (phép toán số học)**

1. ++ --

++a thay đổi giá trị của a trả về giá trị sau khi tăng

a++ trả về giá trị trước khi tăng

1. \* / %
2. + -
3. assignment operators

= += -= \*= /=

Function:

ex: function calculateDiscSquare(r) {

return r \* r \* 3.14;

}

Object Method

var myDog = {

name: ‘Mina’,

weight: 20, // key: value,

isAlive: false

bark: function () { // anonymous function

console.log(‘Gau Gau’)

}

}

myDog.bark();

# **Bài Comparison operators (phép so sánh)**

> >= < <= === !== == !=

**So sánh == vs ===**

**=== khi dùng == thì chỉ so sánh value còn === so sánh cả value và data type**

**!== tương tự ===**

# **Bài Vòng lặp for (for loop)**

**for (init, condition, final-expression)** {

statements;

}

init : biểu thức khởi tạo,

condition: biểu thức điều kiện condition true thì statement thực hiện

final-expression: biểu thức thực hiện sau cùng

**for (let employee of employees){ //lọc qua các value trong mảng**

console.log(employee.name, employee.age)

} // không modify được giá trị array bởi vì for of sao chép employees vào employee nên khi modify chỉ thay đổi được employee không có tác dụng trên employees

// for each giống với for of điều trên nhưng for each có thể làm việc với index

// Chỉ làm việc với for of khi chỉ cần làm việc với các value trong mảng, không cần tới index

**for (var key in myDog) { //lọc qua các key trong object**

console.log(key, );

}

// for in với object: value trong object có thể modify

// for in với array: có thể dùng khi làm việc với index

**// KHÔNG DÙNG CÁI NÀY** (bởi vì những thứ có thể iterable được thì mới for of và for each được)

ex:

for (**const** item of person) { // person là một object không iterable được

  console.log(item);

}

# **Bài Array Methods**

**a.concat(b);** // return mảng mới có các phần tử là phần tử a nối phần tử b nhưng phần tử trong a vẫn không đổi

**a.push(b);** // return về độ dài mới, thay đổi phần tử trong a

**a.pop();** // removes the last element of an array and returns that element. This method changes the length of an array

**a.shift();** // removes the first element of an array and returns that element. This method changes the length of an array.

**a.unshift()** // thêm vào đầu mảng, return length of new array. làm thay đổi mảng a

**a.slice(start,end)** //returns một mảng mới bao gồm các phần tử được trích xuất từ vị trí bắt đầu đến kết thúc, và sẽ KHÔNG bao gồm phần tử end. Không thay đổi mảng a. phần tử end có thể không cần truyền vào, khi đó sẽ hiểu là lấy hết các phần tử còn lại. khi end lớn hơn mảng vẫn không lỗi.

**a.splice(start, deleteCount, item1, item2, ….)** // changes the contents of an array by removing or replacing existing elements and/or adding new elements

ex:

const months = ['Jan', 'March', 'April', 'June'];

months.splice(1, 0, 'Feb');

console.log(months); // Array ["Jan", "Feb", "March", "April", "June"]

months.splice(4, 1, 'May');

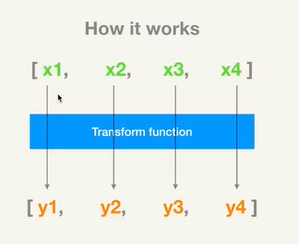
console.log(months); // Array ["Jan", "Feb", "March", "April", "May"]

*\* function as a parameter (callback)dùng function như một tham số*

Callback tức là truyền một đoạn code (Hàm A) này vào một đoạn code khác (Hàm B). Tới một thời điểm, Hàm A sẽ được Hàm B gọi lại (callback)

**array.map**

Cách hoạt động: Biến đổi tập hợp phần tử thành phần tử mới theo một quy tắc nào đó, số lượng phần tử không thay đổi



a.map(function(item){

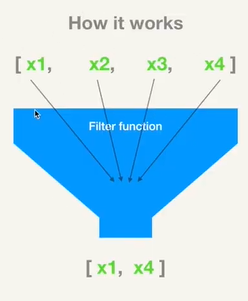
//transform

return newValue;

});

**array.filter**

Lọc ra những phần tử theo filter function



Syntax:

a.filter(function(item){

//filter

return newValue; //true or false

});

**array.find**

trả về giá trị đầu tiên ở trong mảng thỏa mãn điều kiện hoặc có thể trả về undefined

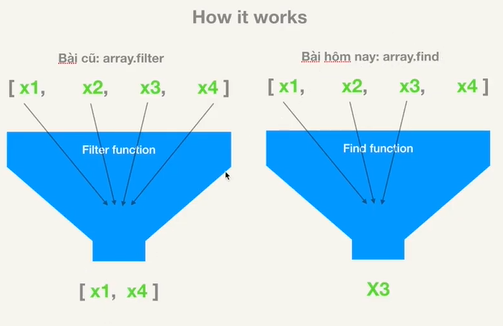
Syntax:

a.find(function(item){

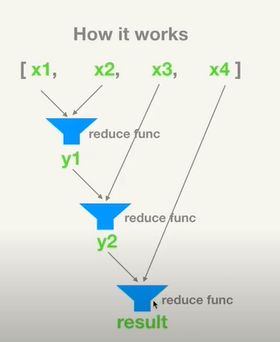
//find

return x; //true or false

});



**array.reduce**



a.reduce(function(item1, item2){

//calculation

return x;

});

a.reduce(function(item1, item2){

//calculation

return x;

}, init); // giá trị khởi tạo ban đầu

*Nếu không có init thì giá trị ban đầu là item 1 và item 2, nếu có init thì init làm tham số đầu tiên, tham số thử 2 là phần tử đầu tiên*

**array.sort(function(a,b){})**

if sort function

1. returns a value <0

a will come before b

1. return a value > 0

a will come after b

1. return 0

a and b stay unchanged

# **Bài Math object**

Math.PI

Math.ceil(number) làm tròn số thập phân lên

Math.floor(number) làm tròn số thập phân xuống

Math.round(number) làm tròn số trả về giá trị của 1 số integer gần nhất

Math.max( x1, x2, x3,…)

Math.min(x1, x2, x3,…)

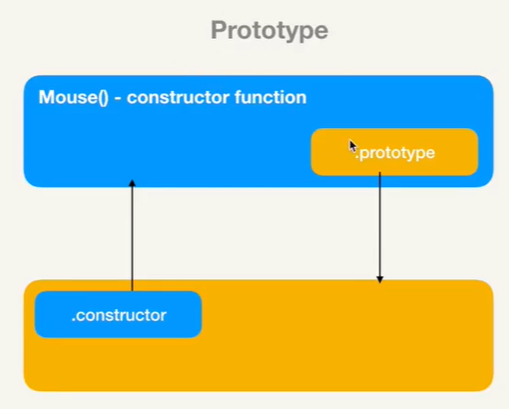
Math.random() trả về số ngẫu nhiên khoảng từ 0 -1

# **Bài ‘new’ keyword**

The ‘new’ keyword is used in javascript to create a object from a constructor function.

# **Bài Prototypes (Khuôn mẫu)**

|  |  |
| --- | --- |
| Constructor function | Normal function |
| Được sử dụng với từ khóa ‘new’, để tạo ra object mới, tên hàm thường là danh từ | Dùng để làm 1 task cụ thể, têm hàm thường đặt theo: sayHello |



prototype object được chia sẻ giữa tất cả object được tạo bởi keyword ‘new’

\* Nếu là 1 constructor thì sẽ có prototype

\* Nếu là 1thằng được tạo từ new, là một constructor function với điều kiện là kế thừa prototype của thằng khác thì có \_\_proto\_\_

# **Bài NPM node package manager**

readline-sync

\* JSON methods

JSON.stringfy convert an object to a JSON string

JSON.parse convert a JSON string to an object

# **Bài if … else và else if**

if … else

else if

# **Bài Ternary operator**

Syntax: condition ? expression when true : expression when false

# **Bài Sync. vs Async. (Đồng bộ và bất đồng bộ)**

Đồng bộ: làm xong việc này mới tới việc khác, chạy theo luồng (tuần tự)

Bất đồng bộ: không cần chờ việc khác xong

# **Bài Callback Hell (nhiều tác vụ phụ thuộc vào nhau )**

# **Bài Promise**

giải quyết vấn đề trong bất đồng bộ

var promise = new Promise (

function (resolve, reject) {

//Thành công: resolve()

//Thất bại: reject()

}

)

Cách hoạt động:

promise

.then( function() {

// được gọi khi resolve() được gọi

})

.catch ( function() {

// được gọi khi reject() được gọi

})

.finally( function() {

// khi 1 trong resolve() và reject() được gọi

})

promise.all([

readFilePromise(‘./song.txt’),

readFilePromise(‘./song1.txt’),

…..

]).then()

.catch()

Bài Node co

Bài async await

# **Bài timing event**

* setTimeout(fn, ms) //miliseconds: sau khi khoảng ms kết thức mới execute func
* setInterval(fn,ms): giống setTimeout(), nhưng lặp lại func sau mỗi khoảng ms

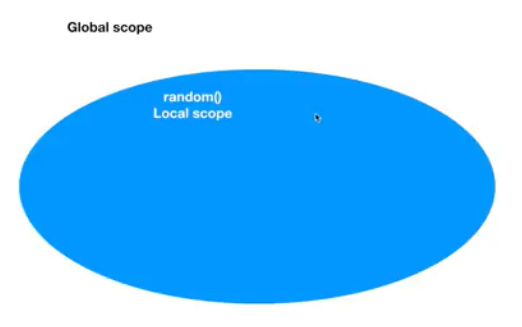
Bài date

Node moment.fromNow

# **Bài scope**

Global scope: toàn cục

Local scope: cục bộ



# Difference between NodeJS and Javascript

- NodeJS

|  |  |  |
| --- | --- | --- |
|  | Javascript | NodeJS |
| 1 | Javascript is a programming language that is used for writing scripts on the website.  Là ngôn ngữ lập trình được sử dụng để viết đoạn mã trên website | NodeJS is a Javascript runtime environment  Là môi trường thực thi JS |
| 2 | Javascript can only be run in the browsers.  Chỉ run trên browsers | NodeJS code can be run outside the browser.  Có thể run ngoài browser |
| 3 | It is basically used on the client-side.  Sử dụng bên phía client | It is mostly used on the server-side.  Phần lớn sử dụng bên phía server |
| 4 | Javascript is capable enough to add HTML and play with the DOM.  Đủ khả năng để thêm HTML và tương tác với DOM | Nodejs does not have capability to add HTML tags.  Không có khả năng thêm HTML |
| 5 | Javascript can run in any browser engine | Nodejs can only run in V8 engine of google chrome. |
| 6 | Javascript is used in frontend development. | Nodejs is used in server-side development. |

\* DOM: (Document Object Model) cơ bản, DOM kết nối các trang web tới các đoạn mã hay các ngon ngữ lập trình

toán tử điều kiện || và &&

tới even loop giải thích khai báo không từ khóa

Khác nhau giữa node co và promise all

<https://developer.mozilla.org/vi/docs/Web/JavaScript/Reference/Global_Objects/Promise/all>