JS BaSic

# 1. javascrip là gì?

Javascrip là ngôn ngữ lập trình được sử dùng nhiều trên nền web.

# 2. Nodejs?

Được việt bằng js. Là môi trường

https://topdev.vn/blog/node-js-la-gi/

Tài liệu:

https://codelearn.io/learning/javascript-co-ban

# Đường dẫn js trong html

<script src="https://www.w3schools.com/js/myScript1.js"></script>

<script src="/js/myScript1.js"></script>

# Output print:

console.log(); => in trên cmd console

doccoment.write(); => in trên web

document.getElementById("demo").innerHTML = “”; document.getElementById("demo").style.color = "red";

=> thay đổi dữ liệu trong thẻ có id=”demo”.

alert(); => in ra thông báo

confirm(); => in ra thông báo

prompt(); => in ra thông báo

## Console :

console.log() => in ra nội dung.

console.warn() => in ra cảnh báo.

console.erro() => in ra cảnh báo lỗi.

# setTime:

## setTimeout

setTimeout(function() {

Lệnh;

},1000)

## setInterval

setInterval(function() {

Lệnh;

}, 1000)

setInterval(

() => {

test2();

test();

test5();

test6();

},

300

);

# Biến - dữ liệu

## Kiểu khai báo biến:

Const: biến không thể thay đổi

Let: từ khóa để xác định một biến có phạm vi hạn chế

Var: dù đặt ở đâu thì tên biến sẽ được khai báo và mang lên trên đầu code.

Không khai báo: dữ liệu và biến sẽ được tính từ lúc bạn khai báo.

## Kiểu dữ liệu của biến:

Kiểu đơn giản:

- kiểu số - number

- kiểu string

- kiểu true flase

- undefined

- null

- symbol

Kiểu phức tạp:

- Function

- Object: Object và array

## Kiểm tra dữ liệu:

Typeof biến;

isNaN(biến);

Chuyển kiểu:

Biến = Biến.toString()

Biến = string(biến)

Biến = number(biến)

## Kiểu true false:

Nhận giá trị false:

False

‘’

Undefined

NaN

Null

## Kiểu string

<https://www.w3schools.com/js/js_string_methods.asp>

https://www.youtube.com/watch?v=b4YivuRmcEw&list=PL\_-VfJajZj0VgpFpEVFzS5Z-lkXtBe-x5&index=23

chuoi = `them bien ${biến} tesst `;

tìm kiếm sửa xóa chuyển đổi kiếm đếm.

Loại bỏ các kí tự đặc biệt.

Chuyển sang dạng mảng

Biến.length()

**Các lệnh thao tác:**

Length chuỗi.length

Find index:

+ chuỗi.indexOf(“chuỗi cần tìm”, ví trí bắt đầu) trả về vị trí đầu tiên tìm được

+ chuỗi.lastIndexOf(“chuỗi cần tìm”) trả về vị trí cuối cần tìm.

Cut string chuỗi.slice(vị trí đầu, vị trí cuối)

Replace chuỗi.replace.(“chuỗi cần thay”, “chuỗi thay”)

Thay tất cả sử dụng biểu thức chính quy chuỗi.replace.(/chuỗi cần thay/g, chuỗi thay”)

Convert to upper case chuỗi.toUpperCase() => in thường thành in hoa

Convert to lower case chuỗi.toLowerperCase() => in thường thành in hoa

Trim chuỗi .trim() loại bỏ khoảng trắng ở đầu cuối.

Split chuỗi.split(“kí tự phân tách thành mảng”) chuyển chuỗi thành array

Get a character by index chuỗi.charAt(vị trí) lấy ra kí tự

## Kiểu number

- làm tròn.

Biến.toFixed(số đằng sau dấu phẩy)

## Kiểu array

<https://www.w3schools.com/js/js_arrays.asp>

https://www.youtube.com/watch?v=KrYacXScNQk&list=PL\_-VfJajZj0VgpFpEVFzS5Z-lkXtBe-x5&index=26

Array.isArray(biến) kiểm tra xem có phải kiểu array không?

Làm việc với mảng:

- Chuyển sang kiểu chuỗi. array.toString()

- array.length => đếm số phần tử trong mảng

- join array.join(“ kí tự phân tách”) => chuyển sang chuỗi

- pop array.pop() => xóa phần tử cuối mảng. và trả về phần tử đã xóa

- push array.push(phần tử 1, phần tử 2, …) => thêm phần tử cuối mảng và trả về độ dài của mảng

- shift array.pop() xóa phần tử đầu mảng. và chả về phần tử đã xóa.

- unshift array.unshif(phần tử 1, phần tử 2, …) => thêm phần tử vào đầu mảng và trả về độ dài mảng.

- splicing array.splice(truyền vị chí, số phần từ muốn xóa, phần tử 1 muốn thêm, phần tử 2 muốn thêm, …)

- concat array.concat(array2). Hợp nhất 2 mảng.

- slicing array.slice(truyền vị trí, xóa đến ví trí nào).

## Array methords:

https://www.youtube.com/watch?v=AT-yhX26\_Ao&list=PL\_-VfJajZj0VgpFpEVFzS5Z-lkXtBe-x5&index=51

Array.forEach(function(phần tử, chỉ số của mảng){}) duyệt từng phần tử của mảng

Aray.every(function(function(phần tử, chỉ số của mảng){ return điều kiện;})) kiểm tra hàng loạt điều kiện của mảng. trả về true false (điều kiện and: đảm bảo tất cả thỏa mãn return true)

Aray.some(function(function(phần tử, chỉ số của mảng){return điều kiện;})) kiểm tra hàng loạt điều kiện của mảng. trả về true false (điều kiện or: đảm bảo một điều kiện thỏa mãn return true)

Aray.find(function(phần tử, chỉ số của mảng){ return điều kiện;})) kiểm tra hàng loạt điều kiện của mảng. Trả về phần tử đầu tiên thỏa mãn.

Array.filter(function(phần tử, chỉ số của mảng){ return điều kiện;})) kiểm tra hàng loạt điều kiện của mảng. Trả về tất cả phần tử thỏa mãn.

Array.map()

Function thay\_doi(data){

Return data;

}

Newarray = Array.map(thay\_doi);

reduce()

## Kiểu function

Có 3 loại:

1. Declaration khai báo bình thường: function hello(){}
2. Expression khai báo :

+ var hello = function hello() {}

+ hello(function hello(){})

+ var myObject = { myFunction(){}}

https://www.youtube.com/watch?v=jE6UPl17Nvo&list=PL\_-VfJajZj0VgpFpEVFzS5Z-lkXtBe-x5&index=28

function test() {

console.log(arguments)

}

## Kiểu object

Thêm cặp key: value=> object.key\_new = value; hoặc object[key\_new] = value;

Thêm biến vào object:

var thongtin = "email";

var myObject = {

    name: "buido",

    [thongtin]: "buivando130898@gmail.com",

    getname: function(){

    return this.name;

    }

}

Object.keys(name\_object); Trả về mảng chứa key

Object.values(name\_object); Trả về mảng chứa value

## Object constructor

function user(fname, lname, avatar) {

    this.fname = fname;

    this.lname = lname;

    this.avatar = avatar;

    this.getname = function(){

        return `${this.fname} ${this.lname}`

    }

}

var author = new user("do", "bui", "av");

var user = new user("do", "bui", "av");

console.log(user.getname());

## Object prototype

Thêm thuộc tính, phương thức vào hàm tạo

// object contructtor

function user(fname, lname, avatar) {

    this.fname = fname;

    this.lname = lname;

    this.avatar = avatar;

    this.getname = function(){

        return `${this.fname} ${this.lname}`

    }

}

user.prototype.khoa = "K20"   //thêm thuộc tính khoa = "K20"

user.prototype.truong = function() {      //thêm phương thức

    return "Vien dai hoc mo Ha Noi";

}

var author = new user("do", "bui", "av");

var use = new user("do", "bui", "av");

console.log(use.getname());

# Date

https://developer.mozilla.org/vi/docs/Web/JavaScript/Reference/Global\_Objects/Date

function date\_gas()

{

    let today\_time = new Date();

    let gas\_date\_month = today\_time.getUTCMonth() + 1;

    let gas\_date = today\_time.getUTCFullYear() + "-" + gas\_date\_month  + "-" + today\_time.getUTCDate();

    return gas\_date;

}

function time\_gas()

{

    let today\_time = new Date();

    let gas\_time = today\_time.getUTCHours() + ":" + today\_time.getUTCMinutes() + ":" + today\_time.getUTCSeconds();

    return gas\_time;

}

# Lệnh điều kiện

## Switch

var date = 3

switch(date) {

    case 2: {

        console.log("thu 2");

    };break;

    case 3: {

        console.log("thu 3");

    };break;

    default:

        {

            console.log("deo biet");

        }

}

Toán tử 3 ngôi:

var coin = 10;

var danh\_gia = coin >10 ? "nhieu" : coin < 10 ? "it" : "lol"

console.log(danh\_gia);

# **Polyfill**

Để được hẫu trỡ các tính năng mới bởi trình duyệt cũ.

# Một số từ khóa:

|  |  |
| --- | --- |
| **Từ khóa** | **Sự miêu tả** |
| break | Kết thúc một công tắc hoặc một vòng lặp |
| continue | Nhảy ra khỏi vòng lặp và bắt đầu ở trên cùng |
| debugger | Ngừng thực thi JavaScript và gọi (nếu có) hàm gỡ lỗi |
| try ... catch | Triển khai xử lý lỗi cho một khối câu lệnh |

# Events

|  |  |
| --- | --- |
| onchange | Một phần tử HTML đã được thay đổi |
| onclick | Người dùng nhấp vào một phần tử HTML |
| onmouseover | Người dùng di chuyển chuột qua một phần tử HTML |
| onmouseout | Người dùng di chuyển chuột khỏi phần tử HTML |
| onkeydown | Người dùng nhấn một phím bàn phím |
| onload | Trình duyệt đã tải xong trang |

# Html DOM

Gồm 3 thành phần:

1. Element: ID, class, tag,..

Css selector,

document.getElementById("demo").innerHTML = “”;

document.getElementBydocument.getElementTagName("demo")

document.getElementsByClassName("demo").style.color = "red";

var node = document.getElementById("demo"); => dùng để chỉnh sửa id.

var node = document.getElementsByClassName("demo"); => dùng để chỉnh sửa class

var node = document.querySelector(‘.tên\_thẻ’)

var node = document.querySelectorAll(‘.tên\_thẻ li’)

1. Attribute
2. Text

## Thay đổi các thuộc tính của html:

<https://www.youtube.com/watch?v=ZS2v9B9DRm8&list=PL_-VfJajZj0VgpFpEVFzS5Z-lkXtBe-x5&index=67>

Them id class thuộc tính bên trong thẻ.

# **Higher order**

function cha(){

    let x = 10;

    return function con(p){

        ++x;

        return x + p;

    }

}

const bien = cha();

console.log(bien(5));

console.log(bien(5));

# OPP

function sinhVien(name, age, diem){

    this.name = name;

    this.age = age;

    this.diem = diem;

    this.kq = function(){

        if(this.diem > 5)

            console.log("hs kha");

        else console.log("hs kem");

    }

}

const stt1 = new sinhVien("do", 23, 9);

stt1.kq();

//Viết thêm hàm:

sinhVien.prototype.test = 30;

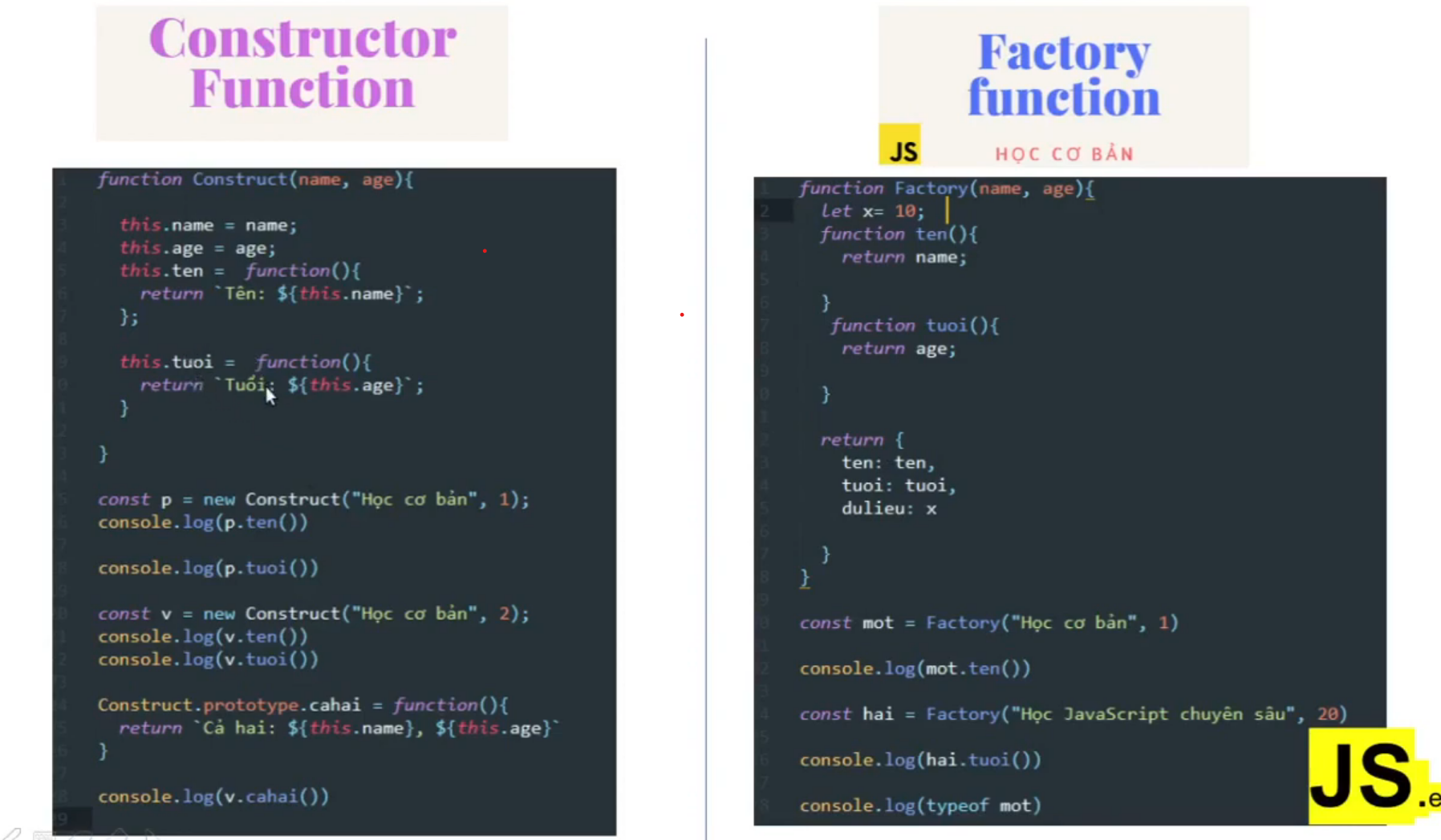
sinhVien.prototype.test1 = function() {

    console.log("dup");

};

console.log(stt1.test);

stt1.test1();



# Promise:

## Hàm promise:

- Sync Đồng bộ

- Async Bất đồng bộ : các hàm setTimeout, setInterval, fetch,

// XMLHttpRequest, file reading,

// request animation frame

## Bất đồng bộ:

## Thời gian chạy:

setInterval(function, time); Lặp sau time.

setTimeout(function, time); Chạy sau time.

setInterval(

() => {

test2();

test();

test5();

test6();

},

300

);

## Callback

- Truyền hàm vào hàm:

VD:

Function myFC(fun) {

Fun(“hello”);

}

Function call\_type(value) {

Console.log(“type:” typeof(value));

}

# Json

<https://www.youtube.com/watch?v=Uph14HYkgEQ&list=PL_-VfJajZj0VgpFpEVFzS5Z-lkXtBe-x5&index=88>

json sang js: JSON.parse(biến)

js sang json: JSON.stringify(biến)