TUTORIAL ES6

Table of Contents

[1 Block 1](#_Toc60777609)

[2 Phân biệt VAR và LET 1](#_Toc60777610)

[2.1 VAR 1](#_Toc60777611)

[2.2 LET 1](#_Toc60777612)

[3 Const 2](#_Toc60777613)

[4 Template string 2](#_Toc60777614)

[5 FOR 2](#_Toc60777615)

[5.1 ARRAY 2](#_Toc60777616)

[5.1.1 for( index = 1; index < lengthArr; index++ ) 2](#_Toc60777617)

[5.1.2 for( index in arr ) 3](#_Toc60777618)

[5.1.3 for( value of arr ) 3](#_Toc60777619)

[5.2 OBJECT 3](#_Toc60777620)

[5.2.1 for( index in obj ) 3](#_Toc60777621)

[5.2.2 for: value of obj 4](#_Toc60777622)

[6 Spread operator 4](#_Toc60777623)

[7 Destructuring 4](#_Toc60777624)

[7.1 ARRAY 4](#_Toc60777625)

[7.2 OBJECT 5](#_Toc60777626)

[8 Default params in functions 5](#_Toc60777627)

[9 Rest params 5](#_Toc60777628)

[10 Arrow functions 6](#_Toc60777629)

[10.1 Không tham số 6](#_Toc60777630)

[10.2 Có tham số 6](#_Toc60777631)

[11 Từ khóa this 6](#_Toc60777632)

[12 Enhanced 7](#_Toc60777633)

[12.1 Properties value shorthand 7](#_Toc60777634)

[12.2 Method properties 8](#_Toc60777635)

[12.3 8](#_Toc60777636)

# Block

for(){

    if(){

        while(){

        }

    }

}

Hàm trên có 3 khối:

BLOCK FOR > BLOCK IF > BLOCK WHILE

# Phân biệt VAR và LET

## VAR

// Var khai báo trong BLOCK, có thể tái sử dụng ngoài BLOCK

var a = 100

if(a > 50){

    var msg = "WIN"

    console.log(msg);

}

console.log(msg);

## LET

// LET chỉ sử dụng trong khối BLOCK, phạm vi nhỏ hơn VAR

let a = 100

if(a > 50){

    let msg = "WIN"

    console.log(msg);

}

console.log(msg);

# Const

Hằng số, không thể thay đổi giá trị biến số, nên viết biến bằng chữ hoa.

const COURSE\_NAME = "ES6";

console.log(COURSE\_NAME);

# Template string

Xuất ra chuỗi có lồng ghép các biến.

var course\_name = "ES6";

<div> Hello ${course\_name} </div>

# FOR

## ARRAY

Array

let arr = ["Java", "NodeJS", "HTML"];

### for( index = 1; index < lengthArr; index++ )

let lengthArr = arr.length;

for(index = 0; index < lengthArr; index++){

    console.log(index  + ': ' + arr[index]);

}

0: Java

1: NodeJS

2: HTML

### for( index in arr )

for(index in arr){

    console.log(index);

}

0

1

2

### for( value of arr )

for(value of arr){

    console.log(value);

}

Java

NodeJS

HTML

## OBJECT

Objects

let obj = {

            course1 : "Java",

            course2 : "NodeJS",

            course3 : "HTML"

};

### for( index in obj )

for(index in obj){

    console.log(index);

}

course1

course2

course3

### for: value of obj

for(index in obj){

    console.log(obj[index]);

}

Java

NodeJS

HTML

Lưu ý: với Object phải sử dụng index để lấy giá trị của Object.

# Spread operator

Sử dụng để lồng ghép các giá trị của mảng này vào mảng khác.

…<name\_array>

let arr1 = ["a", "b", "c"]

let arr2 = ["g", "h", "f"]

let arr = ["d", "e", "f"]

// Sử dụng spread (…) để tạo array có full các phần tử của ba mảng

arr = ["d", ...arr1, "e", ...arr2, "f"]

console.log(arr)

# Destructuring

Sử dụng để gán giá trị của mảng (array) hoặc đối tượng (Object) cho một mảng hoặc đối tượng khác.

## ARRAY

Gán giá trị mảng arr cho mảng **[id, name, free]**

let arr = ["id", "name", "free"];

let [id, name, free] = arr;

console.log(id + ' - ' + name + ' - ' + free);

## OBJECT

Gán giá trị Object obj cho đối tượng **{id, name, free}**

let obj\_value = {

            id: "id obj",

            name: "name obj",

            free: "free obj"

          };

let {name, id, free, admin = "admin"} = obj\_value;

console.log(id + ' - ' + name + ' - ' + free + ' - ' + admin);

**Lưu ý**:

* Khi gán giá trị thì OBJ cần lấy cần trùng id trong obj\_value.
* Thứ tự id trong mảng cần lấy không cần trùng với thứ tự id trong mảng obj.
* Mảng cần lấy có thể bổ sung giá trị mặc định (VD: admin) mà mảng giá trị obj\_value không có.

# Default params in functions

Gán giá trị mặc định cho một variable trong function.

function caculate\_M(d, h, f = 0.5){

    var result = (3.14/4)\*((d^2)/10000)\*h\*f

    return result

}

// Không truyền tham số f - giá trị default param

console.log(caculate\_M(10,8));

# Rest params

Biến đại diện cho tập hợp các giá trị của **mảng**.

function sumValues(...values){

    var result = 0;

    for(let value of values){result += value};

    return result;

}

// Tính tổng các giá trị truyền vào

console.log(sumValues(10, 12, 15));

# Arrow functions

## Không tham số

Không có tham số truyền vào ().

var hello = ()=>{

    console.log("Hello ES6!");

}

hello();

## Có tham số

Một tham số truyền vào, 3 cách viết hàm khác nhau.

Cách viết 1:

var hello = (name)=>{

    console.log(`Hello ${name}` );

}

Cách viết 2:

function hello(name){

    console.log(`Hello ${name}` );

}

Cách viết 3:

var hello = name => console.log(`Hello ${name}` );

hello("ES6!");

# Từ khóa this

Sử dụng để truy suất thông tin thuộc tính của đối tượng.

// John study PHP

// John study Java

// John study HTML

// Cach 1:

let stutdent1 = {

    name: "John",

    courses: ["PHP", "Java", "HTML"],

    showInfo: function(){

        for(let value of this.courses){

            console.log(`${this.name} study ${value}`)

        }

    }

}

stutdent1.showInfo()

// Cach 2:

let stutdent2 = {

    name: "John",

    courses: ["PHP", "Java", "HTML"],

    showInfo: function(){

        this.courses.forEach((course\_value)=>{ // Neu dung function(course\_value) se khong in duoc bien this.name

            console.log(`${this.name} study ${course\_value}`)

        })

    }

}

stutdent2.showInfo()

# Enhanced

## Properties value shorthand

Sử dụng để thay cho cách viết “key : value” trong JavaScript trước đây:

var course\_PHP = "PHP"

var course\_JAVA = "JAVA"

// Trước đây

var all\_course1 = {

    course\_PHP: course\_PHP,

    course\_JAVA: course\_JAVA

}

// Sử dụng Properties value shorthand

var all\_course2 = {

    course\_PHP,

    course\_JAVA

}

## Method properties

Loại bỏ chữ function(), viết method trực tiếp vào key.

// Trước đây

var object1 = {

    name: "Java 1",

    showInfo: function(){

        console.log(this.name)

    }

}

// Method properties

var object2 = {

    name: "Java 2",

    showInfo(){ // Bỏ chữ function, viết trực tiếp vào function-name

        console.log(this.name)

    }

}

object2.showInfo();

## Computed property names

Sử dụng để tùy biến tên proterty-name (key) của đối tượng. Property-name phải đặt trong dấu [property-name].

// Sử dụng để tùy biến [property name] (giá trị key) trong Object

let prefix = "prefix\_"

let Courses = {

    [prefix + "php"]: "PHP",

    [prefix + "java"]: "JAVA"

}

console.log(Courses)

# OOP

## Class

Sử dụng để tạo đối tượng, khai báo thuộc tính, viết phương thức cho đối tượng.

class Stutdent {

    constructor(code, name, age){

        this.name = name,

        this.code = code,

        this.age  = age

    }

    showInfo(){

        return this.code + " - " + this.name + " - " + this.getAge()

    }

    getAge(){

        return this.age;

    }

}

let studentObj = new Stutdent("SV001", "John", 28);

console.log(studentObj.showInfo());

## Inheritance

Kế thừa thuộc tính của lớp cha và phương thức của lớp cha.

class Person {

    constructor(code, name, age){

        this.name = name,

        this.code = code,

        this.age  = age

    }

    showInfo(){

        return this.code + " - " + this.name + " - " + this.getAge()

    }

    getAge(){

        return this.age;

    }

}

class Student extends Person {   // Kế thừa lớp Person

    constructor(code, name, age, score){

        super(code, name, age)   // Kế thừa thuộc tính code, name, age của class cha

        this.score = score       // Bổ sung thêm thuộc tính score

    }

    showInfo(){

       return super.showInfo() + " - " + this.score   // Kế thừa phương thức showInfo từ lớp cha và bổ sung thêm thuộc tính score

    }

}

let studentObj = new Student("SV001", "John extend", 28, 99);

console.log(studentObj.showInfo());