**Default parameters.**

* Khi ta khai báo mắc định thì lúc hàm mà không truyền tham số thì giá trị mặc định sẽ được lấy.

*function* showInfo(*name*, *price*=20){

return `Course name:<b>${name}</b> , price:<b>${price}</b> `;

}

*var* dom = document.getElementById("mContent");

dom.style.color = "#01744DF";

dom.innerHTML = showInfo("ES6",5);

**Rest params (Tham số còn lại).**

//...values là khai báo một mảng các tham số truyền vào

*function* caculate(*action*,...*values*){

*let* result = 0;

switch(action){

case '+':

//Duyệt tất cả các giá trị của tham số truyền vào

//và thực hiện cộng chúng lại.

for(*let* value of values) result +=value;

break;

case '-':

for(*let* value of values) result -= value;

result+=values[0];

break;

}

return result;

}

*console*.log(caculate("-",2,3,3));

**Arrow functions**

//Cách viết và gọi hàm bình thường.

*function* func01(*name*, *age*){

return `My name is ${name}, ${age} years old`;

}

*console*.log(func01("John",69));

//

*var* func02 = *function*(*name*, *age*){

return `My name is ${name}, ${age} years old`;

}

*console*.log(func01("John",69));

//

*var* func03 = (*name*, *age*) *=>* {

return `func03 My name is ${name}, ${age} years old.`;

}

*console*.log(func03("John",69));

//Khi chỉ trả về có một khối lệnh thì không cần cặp ngoặc {} và từ return.

*var* func04 = (*name*, *age*) *=>* `func04 My name is ${name}, ${age} years old.`;

*console*.log(func04("John",69));

//Nếu chỉ có một tham số truyền vào ta có thể bỏ đi cập ().

*var* func05 = *name* *=>* { return `func05 My name is ${name}`;}

*console*.log(func05("John"));

//Khi mà không đó đối số truyền vào.

*var* func06 = ()*=>*{

return `This is a func06`;

}

**Demo sử dụng Arrow function.**

//Chuyển tất cả các chữ thường về chữ hoa.

//nếu viết đầy đủ thì course sẽ (course),

//ta có thể lược bỏ dấu () để thành courses.map((course)=>{...}

*let* courses = ["Android","Java","php","angular"];

*console*.log(

courses.map((*course*)*=>*{

return course.toUpperCase();

})

);

//Sắp xếp mảng tăng dần

//từ khóa function được thay bằng dấu =>

*let* scores = [3,5,7,8,9];

scores.sort(

//Trong trường hợp chỉ có một lệnh thì ta có thể bỏ đi {}

(*x*,*y*)*=>*x>y ?true: false

);

*console*.log(scores);

**Con trỏ this.**

//Đây trường hợp có lỗi, không có giá trị cho this.name.

*let* student = {

name: "John",

courses: ["Android","PHP","Java"],

showInfo:*function*(){

this.courses.forEach(*function*(*course*){

*console*.log(`${this.name} study ${course}`);

});

}

}

//Đây là cách viết ta không thường dùng, chỉ để tham khảo mà thôi.

*let* student = {

name: "John",

courses: ["Android","PHP","Java"],

showInfo:*function*(){

*var* \_this = this;

this.courses.forEach(*function*(*course*){

*console*.log(`${\_this.name} study ${course}`);

});

}

}

//Đây là cách viết ta không thường dùng, chỉ để tham khảo mà thôi.

*let* student = {

name: "John",

courses: ["Android","PHP","Java"],

showInfo:*function*(){

this.courses.forEach(*function*(*course*){

*console*.log(`${this.name} study ${course}`);

}.bind(this))

}

}

//Cách thường dùng

*let* student = {

name: "John",

courses: ["Android","PHP","Java"],

showInfo:*function*(){

this.courses.forEach((*course*)*=>*{

*console*.log(`${this.name} study ${course}`);

})

}

}

//Ta có thể đơn giản hóa nó lại như sau

*let* student = {

name: "John",

courses: ["Android","PHP","Java"],

showInfo:*function*(){

this.courses.forEach(*course=>console*.log(`${this.name} study ${course}`))

}

}

student.showInfo();

**Enhanced.**

//Property value shorthand

*function* getCourse(*name*, *price*,*free*){

return {

name: name,

price: price,

free: free

}

}

*console*.log("ES6",20,false);

//Method properties

*function* getCourse(*name*, *price*,*free*){

return {

name,

price,

free,

showInfo1: *function*(){

*console*.log(`${name}`+'-'+`${price}`+'-'+`${free}`);

},

showInfo2(){

*console*.log(`${name}`+'-'+`${price}`+'-'+`${free}`);

},

//Đây là method property.

//delimiter mang gias trị mặc định là dấu "-"

showInfo3(*delimiter*="-"){

*console*.log(`${name+delimiter+price+delimiter+free}`);

}

}

}

*var* myCourse = getCourse("ES6",20,false);

myCourse.showInfo3("---");

*console*.log(myCourse);

//computed property names

*let* propPrefix = "support\_";

*let* bootstrapSupport = {

[propPrefix + "chrome"]: true,

[propPrefix + "ie"]: true,

[propPrefix + "firefox"]: false

}

*console*.log(bootstrapSupport);

**Hướng đối tượng class**

*class* Student{

//Hàm khởi tạo cho lớp Sinhvien

*constructor*(*code*, *name*, *age*){

this.code = code;

this.name = name;

this.age = age;

}

showInfo(){

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

}

getAge(){

*let* today = new Date();

*let* year = today.getFullYear();

return year - this.age;

}

}

*let* studentObj = new Student("sv1","John",1992);

*console*.log(studentObj);

*console*.log(studentObj.showInfo());

**Kế thừa**

*class* Person{

*constructor*(*name*, *code*, *age*){

this.name = name;

this.code = code;

this.age = age;

}

showInfo(){

*console*.log(this.name + "-" + this.code + "-" + this.getAge());

}

getAge(){

*let* today = new Date();

*let* year = today.getFullYear();

return year - this.age;

}

}

//Student kế thừa từ lớp Person

*class* Student extends *Person*{

*constructor*(*name*, *code*, *age*, *score*){

//Gọi constructor của lớp cha (Person)

super(name, code, age);

this.score = score;

}

showInfo(){

//Gọi lại showInfor() của lớp cha (Person)

super.showInfo();

*console*.log("-"+this.score);

}

}

*let* personObj = new Person("john","121",1990);

personObj.showInfo();

*let* studentObj = new Student("john","33333",1990,80);

studentObj.showInfo();