1、检测对象属性是否存在（对象.属性名）。

①使用undefined判断

console.log(obj.id === undefined) //true===>表示不存在，false===>表示存在。

②使用对象的hasownProperty()方法

console.log(obj.hasOwnProperty('id')) //对象下方法hasOwnProperty('属性名') true=>存在，false=>不存在

③使用关键字in

console.log('id' in obj) true==>存在，false==>不存在

④在条件语句中直接判断

if(obj.id){ } //undefined、null、空字符串会自动转换为false。

对象的方法。

方法对应的是一个函数。

var person = {

name: '张三',

sex: 1,

play: function (a,b,c#形参) {//方法对应的是函数

console.log(this.name+"--play--")

return X;#返回值

},

play2:play2,

}

function play2(){

console.log(this.name+"--play2--")

}

// 调用对象的方法

person.play();

person.play2();

console.log(person)

数据存储方式：

原始数据类型直接将数据保存在栈内存中。

引用类型将数据保存在堆内存中，同时生成一个地址(指针)，将地址保存在栈内存中（变量）。

引用类型数据的销毁：一旦引用类型数据销毁后，就会释放内存空间，加快运行速度。只要引用类型数据不被任何地址所指向就会自动销毁。

null：空地址，不指向任何引用类型数据类型。

销毁引用类型数据

person = null;

person2 = null;

var a = {a:1,b=2}

var b = {a:1,b=2}

console.log(a===b) //false,比较在栈内存中的地址

// 原始类型数据存储方式

var a = 1;//在栈内存中开辟了一片空间为a，值为1。

var b = a;

a = 2;

// 引用类型数据存储，創建對象的同時，在堆内存中开辟空间保存数据，会生成一个地址(指针)（16进制），把地址保存到栈内存的变量中。

// 把对象的变量赋值给另一个变量，把地址拷贝一份再赋值，两者指向同一个对象。

var person = {

name: "ran",

size: 'xl'

}

// 把一个对象的变量赋给另一个变量，两个变量指向同一个对象

var person2 = person;

person2.size = "xxxxl"

console.log(person2)