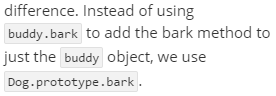
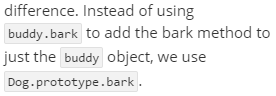
****

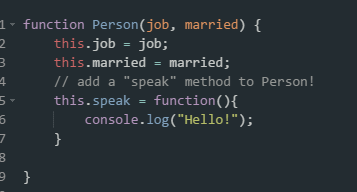
把方法加给父类，而不是子类。

****

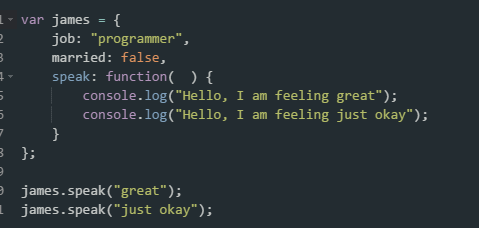
把方法加给父类，而不是子类。

Add method to Object:

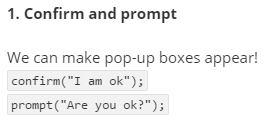
第一种方法



第二种方法





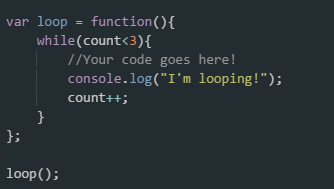
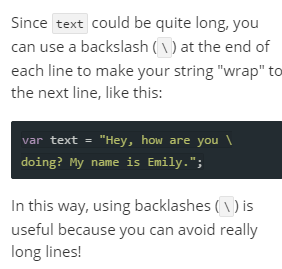
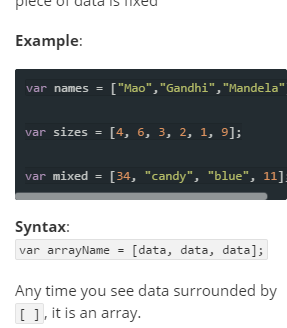
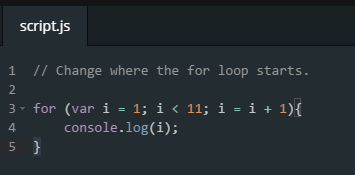


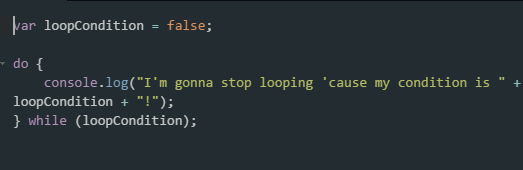
console.log("The answer makes sense!");

"wonderful day".substring(3,6);

var myColor = "red";

console.log(myColor.length);

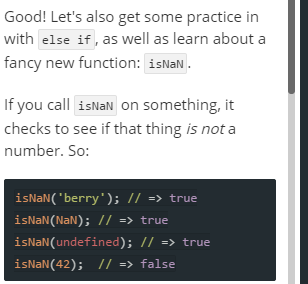




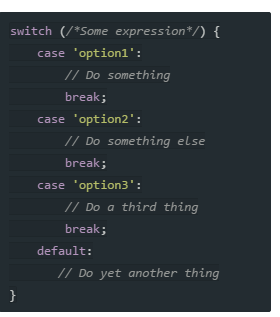
either 0 (which JavaScript reads as false) or 1 (which JavaScript reads as true).

Math.floor() 只取整数位。

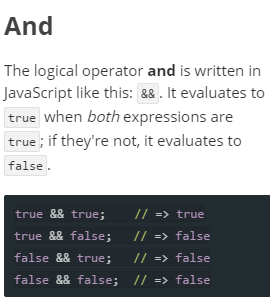
所以 **Math.Floor(Math.random()) 要么是0 要么是1. Random取得0到1的一个随机小数。**



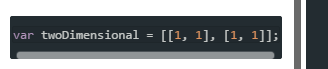
检查一个变量 是否是数字

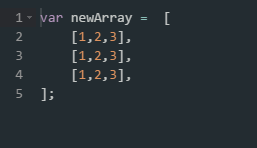


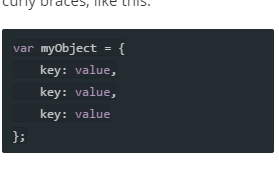
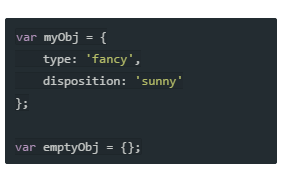
单引号，别忘了break。还有default：

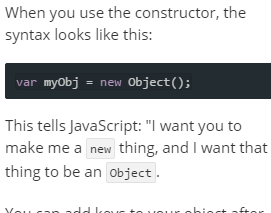


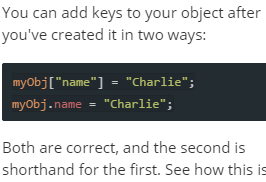
变大小写 



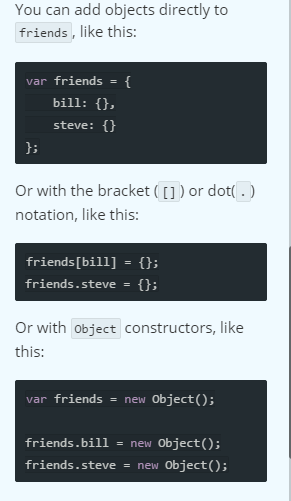


注意逗号 连接





Add object

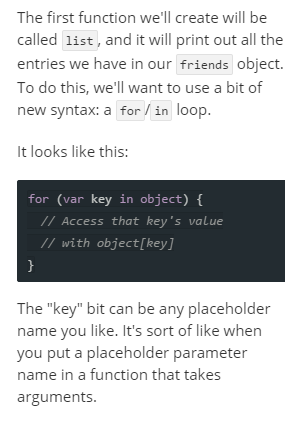


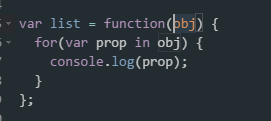
在 object中添加object的两种方法：

方法一 

方法二

For循环的第二种写法。利用 for（var key in object）





**var friends = {};**

**friends.bill = {**

firstName: "Bill",

lastName: "Gates",

number: "(206) 555-5555",

address: ['One Microsoft Way','Redmond','WA','98052']

**};**

**friends.steve = {**

firstName: "Steve",

lastName: "Jobs",

number: "(408) 555-5555",

address: ['1 Infinite Loop','Cupertino','CA','95014']

**};**

**var list = function(obj) {**

**for(var prop in obj) {**

**console.log(prop);**

**}**

**};**

**var search = function(name) {**

**for(var prop in friends) {**

**if(friends[prop].firstName === name) {**

**console.log(friends[prop]);**

**return friends[prop];**

**}**

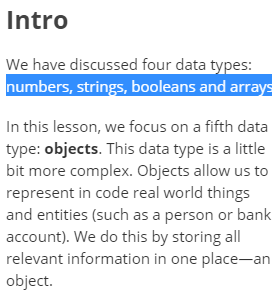
**}**

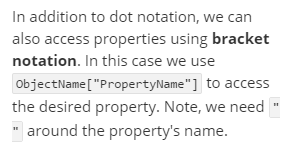
**};**

list(friends);

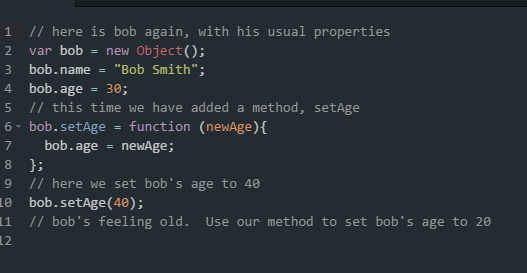
search("Steve");

OBJECTS

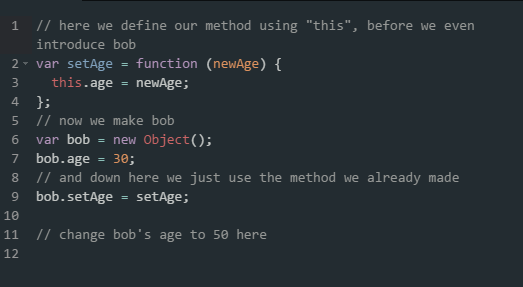


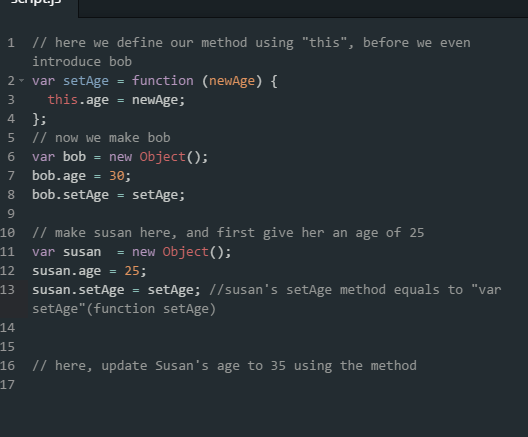


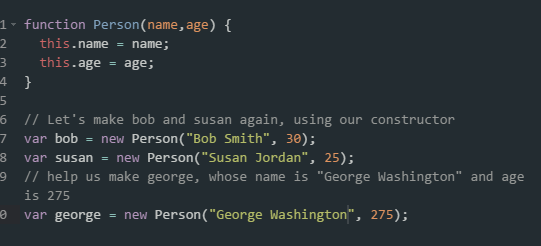
**Object内部的方法**



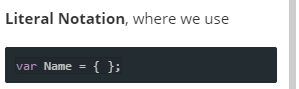
方法括号内的 参数 不需要提前声明，只是内部变量。



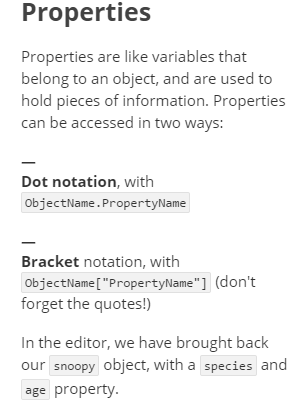
利用已经创建的变量，new一个新的子变量。



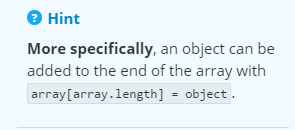
**两种构造方法：**

1. ****
2. **using constructor notation**
3. **Var newObject = new Object（）；**

**两种调用方法**

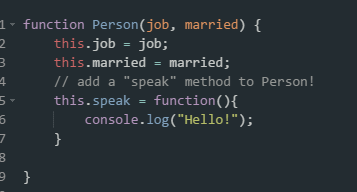


如何往array里面加 东西

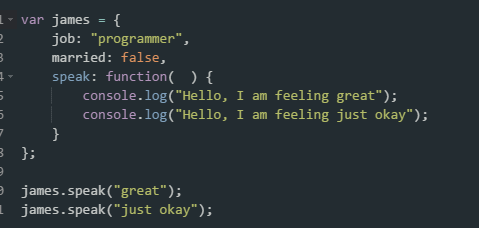


Add method to Object:

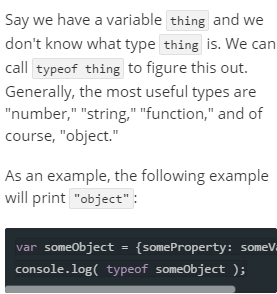
第一种方法



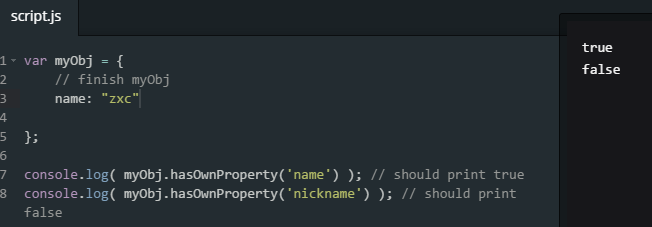
第二种方法



如果不知道 打印的东西的 种类 使用 typeof

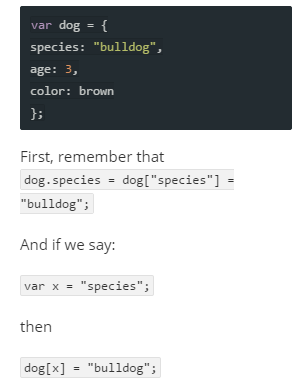


**判断object里面有没有这个属性 hasOwnProperty**

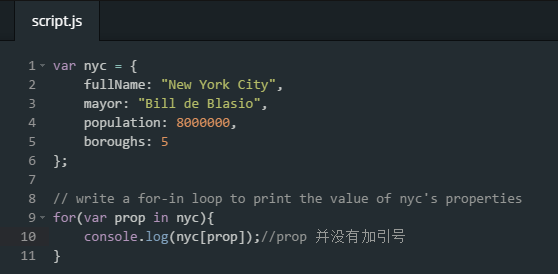


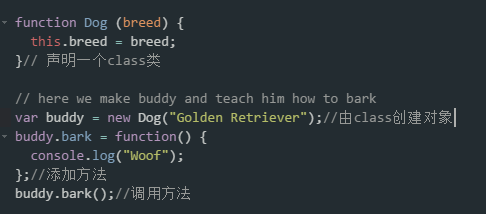
1. **打印出obj里面所有的 元素名字**





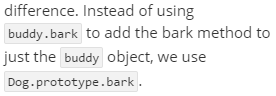
1. **打印出obj 中 元素的 value**



****

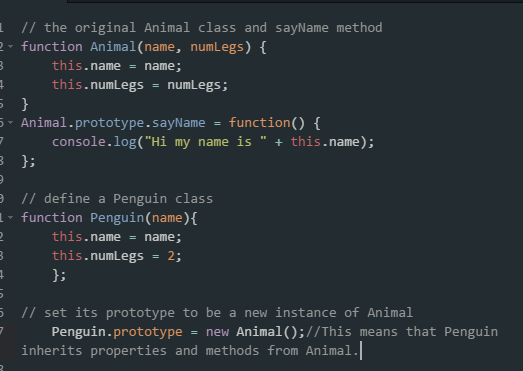
**声明class 创建对象 添加方法**

**给类添加方法，那么由类生成的对象 就都可以调用这个方法了。**Because we have now changed the prototype for the classDog. **This immediately teaches allDogs the new method.**

****

把方法加给父类，而不是子类。

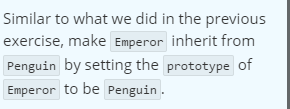
改变 类的 prototype 变成其他类的 prototype 相当于继承了其他类，然后可以调用该类的对象和方法。

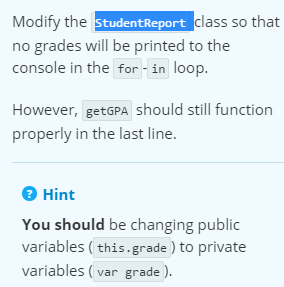
****

**接上图。 var penguin = new Penguin("aa");**

**penguin.sayName();**

**产生一个新的Penguin对象 就可以调用 Animal类里面的方法了。继承的过程**

****

**如何公有 变私有。在类里面不是this。 而是 var**

**var cashRegister = {**

**total: 0,**

**//insert the add method here**

**add : function(itemCost){**

**this.total += itemCost;**

**}, //在一个obj里面定义 东西的时候 要用 逗号 隔开**

**scan: function (item) {**

**switch (item) {**

**case "eggs":**

**this.add(0.98);**

**break;**

**case "milk":**

**this.add(1.23);**

**break;**

**//Add other 2 items here**

**case "chocolate":**

**this.add(0.45);**

**break;**

**case "magazine":**

**this.add(4.99);**

**break;**

**}**

**return true;**

**}**

**};**

**//Scan 2 eggs and 3 magazines**

**cashRegister.scan("eggs");//引用的时候要加 引号**

**cashRegister.scan("eggs");**

**cashRegister.scan("magazine");**

**cashRegister.scan("magazine");**

**cashRegister.scan("magazine");**

**//Show the total bill**

**console.log('Your bill is '+cashRegister.total);**