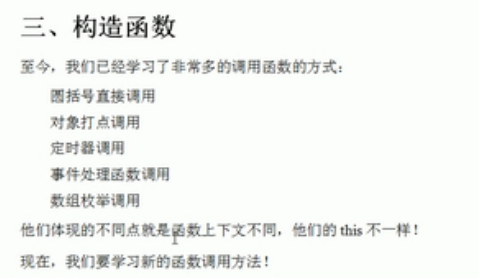
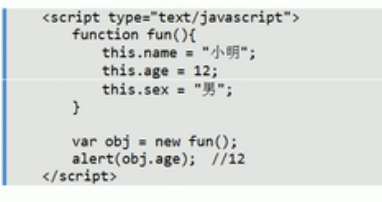
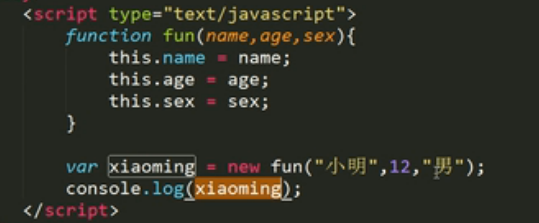
11 构造函数

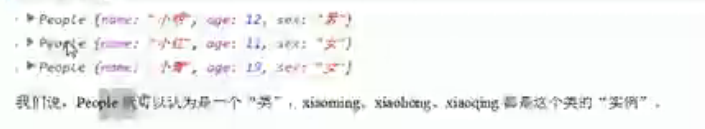




使用new运算符调用一个函数时，会经历四步走：

1. 函数内部悄悄的创建一个局部变量，是一个空对象{}
2. 函数将自己的上下文设置为整个对象，即所有语句中的this就表示这个对象
3. 函数执行所有语句
4. 所有语句执行完毕后，函数将return这个对象，函数将把自己的上下文返回





JS中没有类的概念，我们这里只是和JAVA、c++、c#做一个类比，js只有构造函数，当一个函数被new操作符调用的时候，这个函数是一个构造函数，它总能返回一个类的具有相同属性群的对象，感觉在构造东西，所以这个函数很神奇，像一个模子，在制作类似的对象。

为了提醒其他的程序员，这是个必须用new调用的函数，换句话说提醒别人这是一个构造函数，这类函数的名字必须首字母大写。



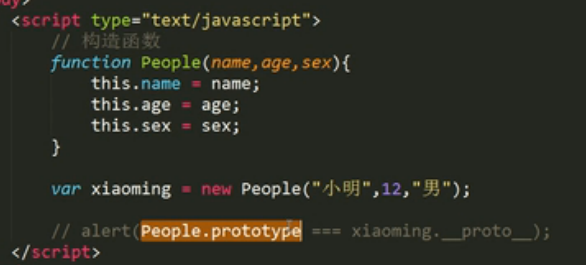
一个函数是否是构造函数取决于怎么调用，如果使用new操作符调用则是构造函数，否则不是。

12 原型链

每一函数都有一个属性叫做prototype，指向一个对象，当这个构造函数被new的时候，它的每一个实例的\_\_prototype\_\_属性也指向这个实例。

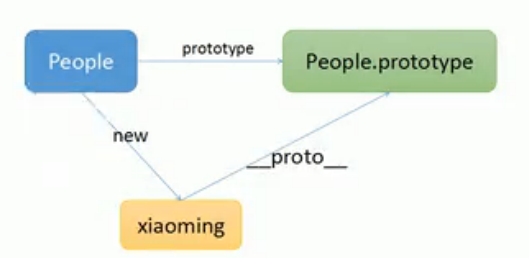
函数prototype🡺实例\_\_proto\_\_

console.log('Person.prototype===a.\_\_proto\_\_? '+(Person.prototype===a.\_\_proto\_\_));//true



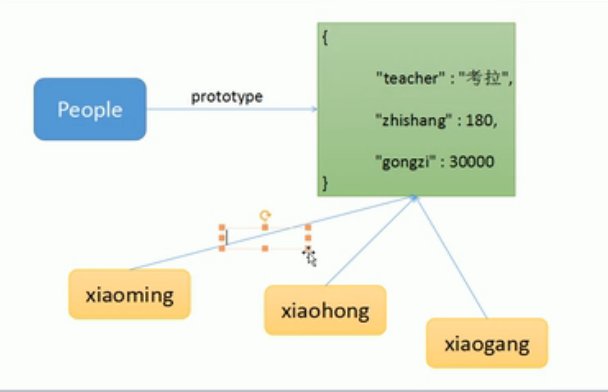
每一个函数天生都有一个prototype属性，指向一个空对象，也就是说，我们不需要定义这个prototype属性，任何一个函数只要写出来就拥有了prototype属性，这个属性对它自己而言没有任何意义。它的唯一意义是儿子们的指明灯。构造函数的prototype指向谁，它new出来的实例的\_\_proto\_\_属性就指向谁。有上面我们也知道了

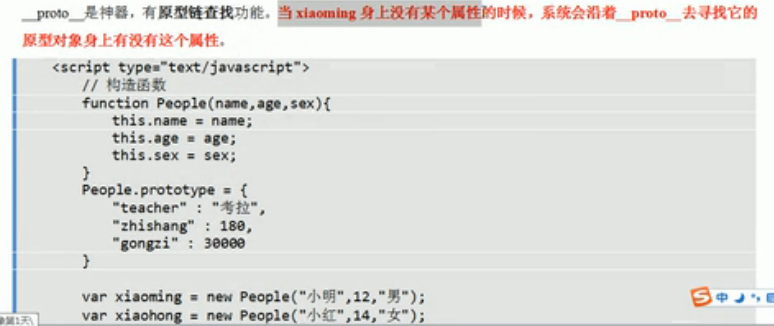
构造函数.prototype===其实例.\_\_proto\_\_成立。

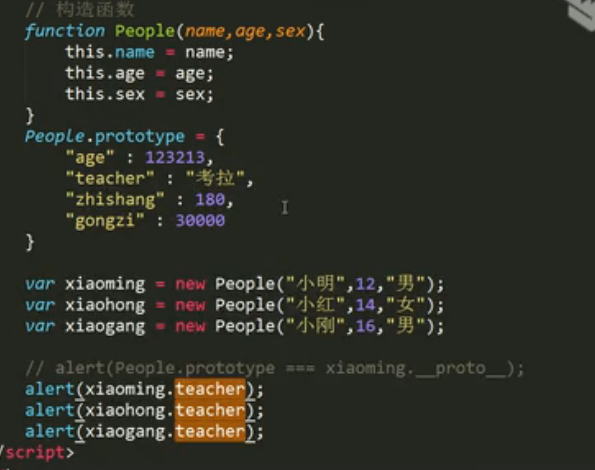


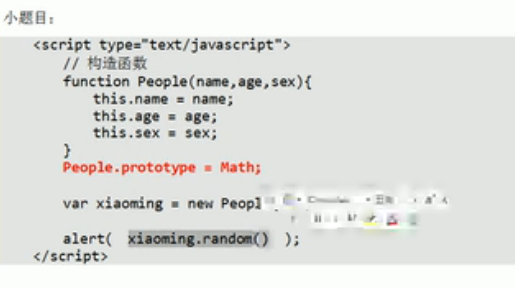
我们称People.prototype是People构造函数的原型，People.prototype是小明的原型对象

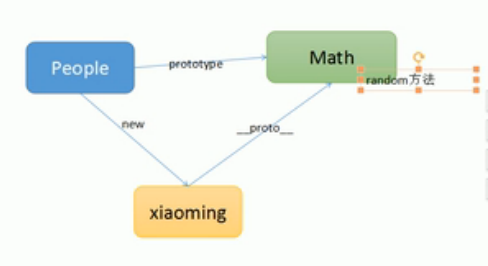
\_\_proto\_\_是神器，有原型链查找功能，当xiaoming身上没有某个属性的时候系统会沿着\_\_proto\_\_去寻找它的原型对象上有没有这个属性。







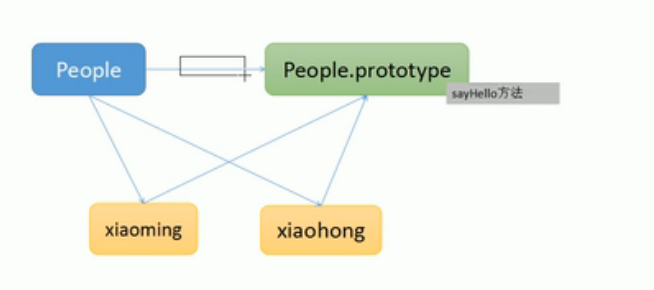




4.3 方法的定义在原型上

我们刚才知道如果把函数写在构造函数里面，就相当于定义在了对象上，此时函数会产生多个同的副本，相应的对象删的方法也会产生多个。影响程序的效率。此时可以把函数定义在函数的原型上，这样所有new出来的东西，\_\_proto\_\_就会指向这个对象，从而能够使用这个方法。

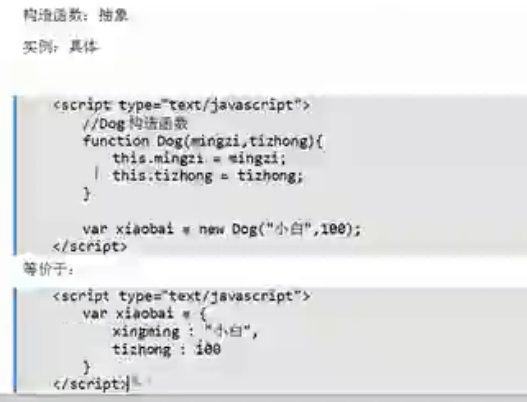




13 构造函数和实例的关系

构造函数：抽象，相当于面向对象语言中的类class。

实例：具体

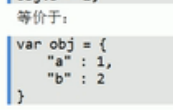


14 内置构造函数1



系统内置了很多的构造函数，它们也叫做我们的基本类型值，引用类型值的包装类。



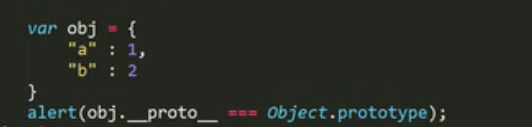


也就是说字面量语法底层也是调用了new Object

console.log(new Object);

console.log({});

所以，任何一个json的\_\_proto\_\_都是Object.prototype。因为json就是object的实例。



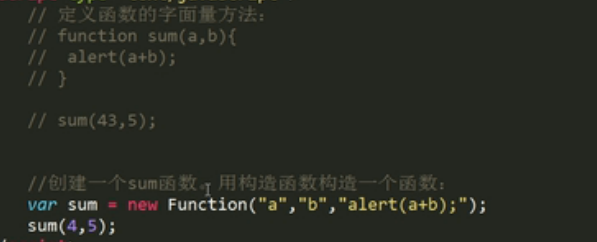
Object.prototype是所有对象的原型链终点。

3.2 Funchion函数

系统内置了一个Function的构造函数 ，用于构造函数，事实上所有的function字面量都是他的实例。







window.onload=function(){

// 所有的的字面量function都是Function的实例

function func(){

}

console.log('func instanceof Function? '+(func instanceof Function));//true

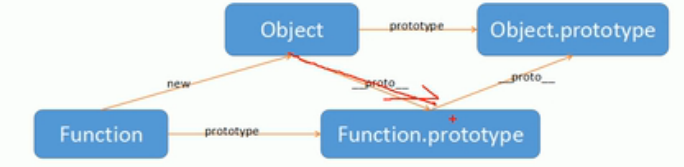
var func2=new Function('a','b','console.log(a+b);');

console.log(func2);

console.log('func2 instanceof Function? '+(func2 instanceof Function));//true

}

任何函数都是Function的实例,Object也是Function的实例，Function自己也是自己的实例。



window.onload=function(){

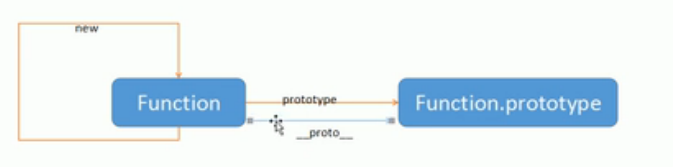
console.log('Function.prototype===Object.\_\_proto\_\_? '+(Function.prototype===Object.\_\_proto\_\_));//true

console.log('Function.prototype.\_\_proto\_\_===Object.prototype? '+(Function.prototype.\_\_proto\_\_===Object.prototype));//true

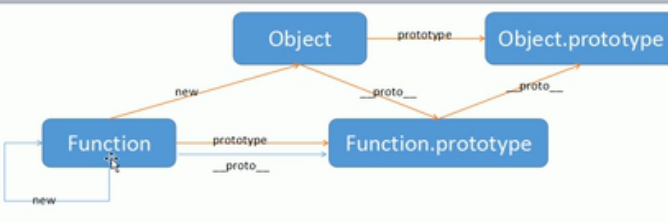
console.log('new Funtion instanceof Object? '+(new Function instanceof Object));//true

}

Object是Function的实例，所以Object.\_\_proto\_\_指向Function.prototype，而Function.prototype是Object的实例，所以Function.prototype.\_\_proto\_\_指向Object.prototype;Function是Function的实例。



console.log(Function.prototype===(new Function).\_\_proto\_\_);//true



3.3 Array 函数

Array是系统内置的数组构造函数，用于构造数组

window.onload=function(){

var arr=new Array();

arr[0]='a';

arr[1]=1;

console.log(arr);

}

这等价于：

var arr2=['a',1];

有一个坑，就是数组的构造函数里面可以写参数，表示数组的长度，但是数组可以自动扩展，不受长度的限制。

var arr3=new Array(2);

console.log(arr3.length);//2

console.log(arr3[0]);//undefined

console.log(arr3[1]);//undefined

console.log(arr3[2]);//undefined,能越界

console.log(arr3);//[undefined × 2]

arr3[0]=0;

console.log(arr3.length);//2

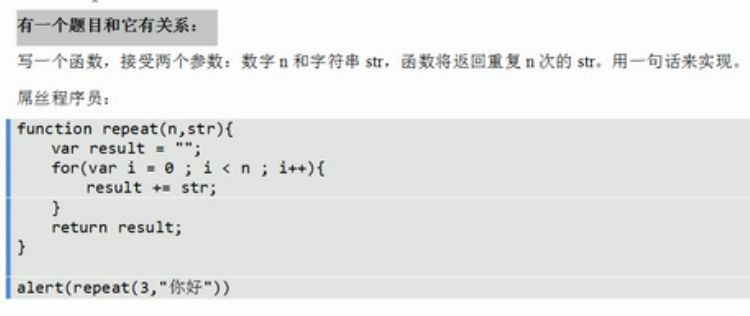
arr3[1]='a';

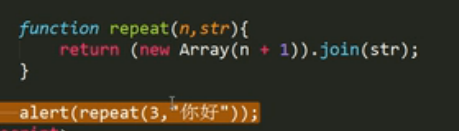
console.log(arr3.length);//2

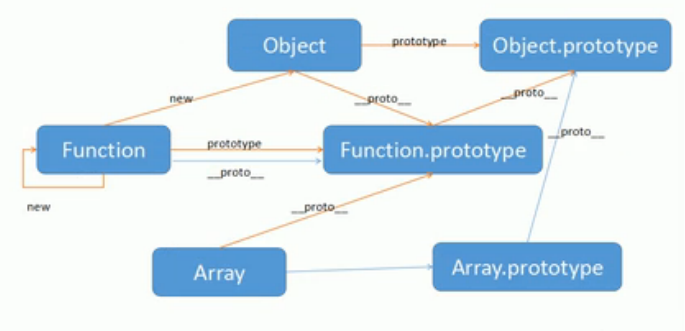
arr3[2]=2;

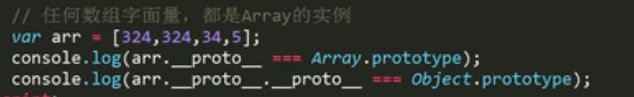
console.log(arr3.length);//3

console.log(arr3);









3.4 RegExp函数

任何一个正则表示，都是RegExp函数的实例

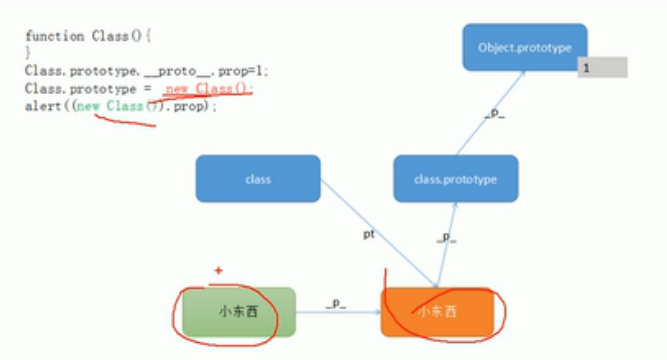
var reg1=/\w/ig;

var reg2=new RegExp('\\w','ig');

console.log(reg1);// /\w/gi

console.log(reg2);// /\w/gi

console.log('reg1==reg2? '+(reg1==reg2));//false



3.6 Number函数

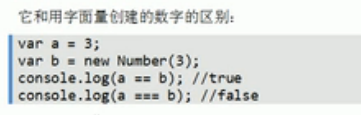
Number是系统的内置构造函数，用于创建数字对象

var num1=new Number(10);

console.log(num1);//Number {[[PrimitiveValue]]: 10}

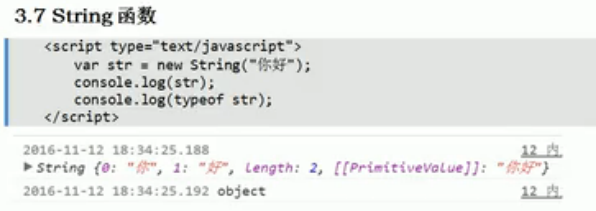
console.log(num1.\_\_proto\_\_.\_\_proto\_\_===Object.prototype);//Object.prototype

也就是说，用内置的构造函数创建数字的时候，得到的是一个对象，这个对象的原始值属性PrimitiveValue是3，此时这个属性不能被枚举。

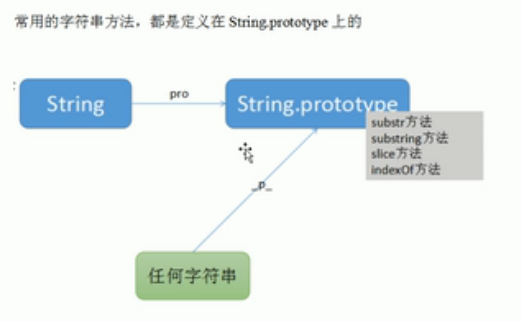




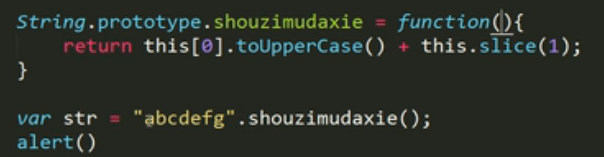
任何需要转为数字的隐式转换，实际上就是在调用Number函数。



在js上，字面量的字符串和new String()建立的字符串是完全一样的，这不同于java，所以在js中基本不使用new String来建立字符串。



在原型上增加方法，拓展内置原型或者对象的方法





//===================20171009