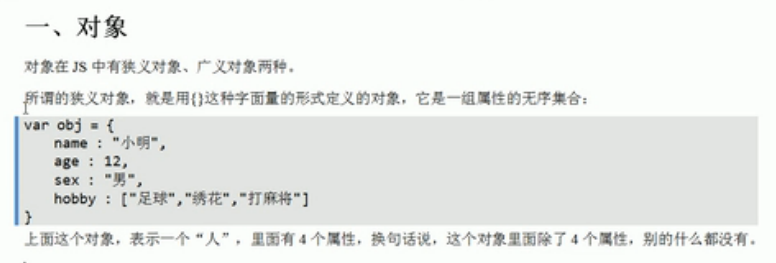
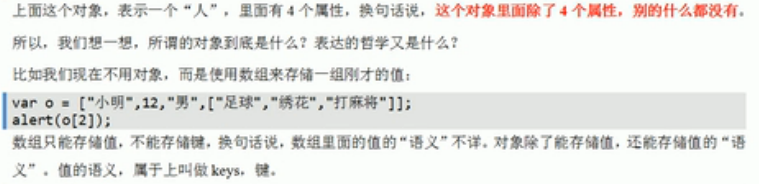
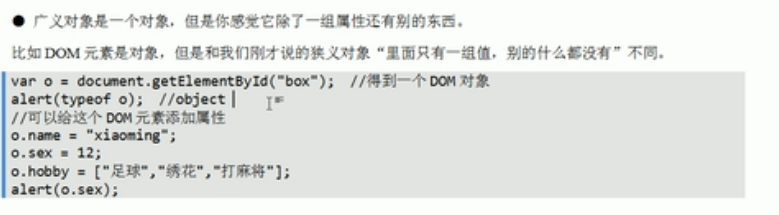
01认识对象



字面量创建的对象与new Object定义一样



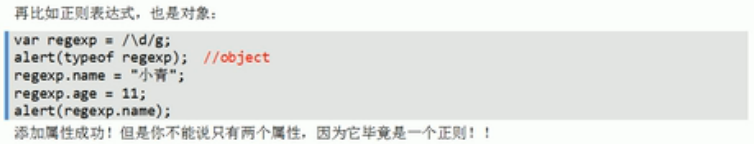


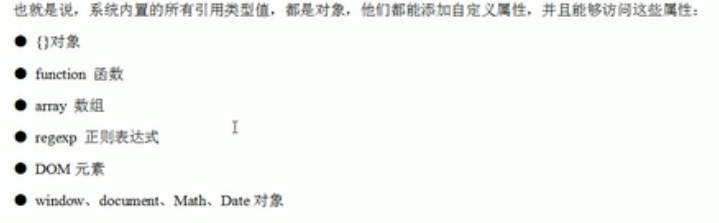




以下代码测试在node

function test(){}  
undefined  
typeof test  
'function'  
test.name='function'  
'function'  
test  
[Function: test]   
test.name  
'test'





对象都能添加自定义属性，除了基本数据类型（number string boolean undefined null）外所有的都是对象。

> reg=/box/ig

/box/gi

> reg

/box/gi

> reg.name='regexp'

'regexp'

> reg

{ /box/gi name: 'regexp' }

> reg.test('this is a box')

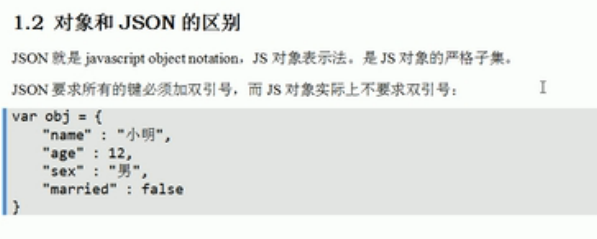
true

还有基本类型的包装类



字符串不能添加属性，但也不会报错。

02 json和对象



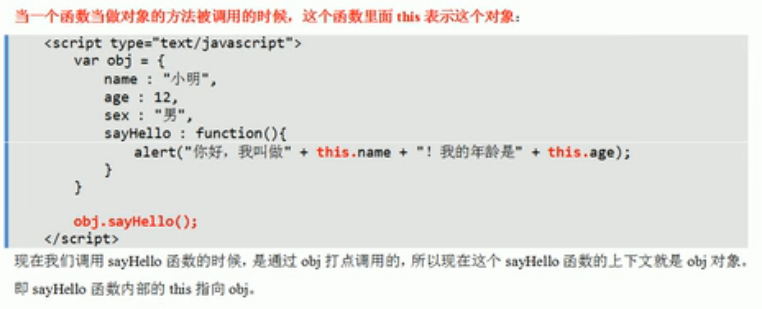
在js中的键名不需要加双引号，但是不符合js标识符命名规范的建需要加双引号（不能是单引号）



03 对象的方法



上下文在这里指的是函数的this指代的对象

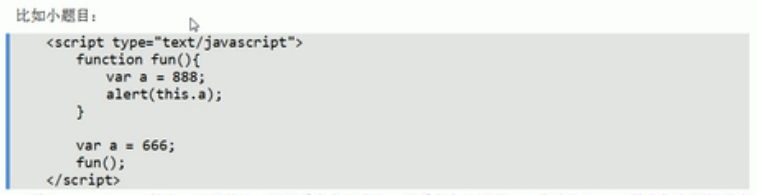




函数的上下文取决于函数的调用而非函数的定义，上下文指调用函数的对象。

04 函数的上下文-规律1





所有的全局变量都是window对象的属性，而函数里面的局部变量，不是任何对象的属性只是局部变量



function outter(){

var a='outer';

console.log(this.a);

function inner(){

var a='inner';

console.log(this.a);

}

inner();

}

console.log('window.a:',this.a);

outter();//内层和外层的this在这里都是window

05 函数的上下文-规律2

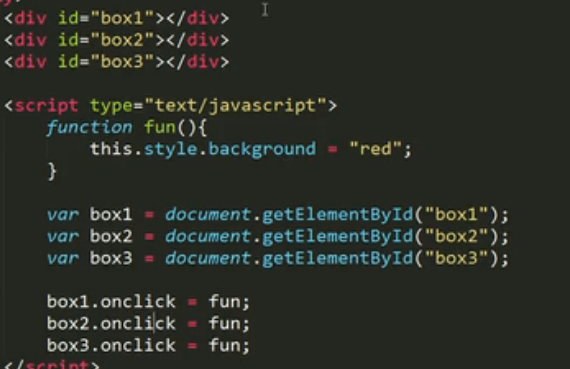






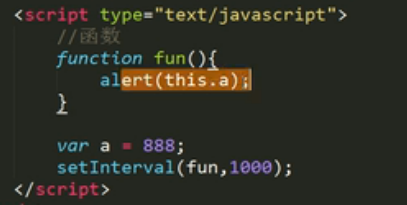
06 函数的上下文-规律3





04 函数的上下文-规律4



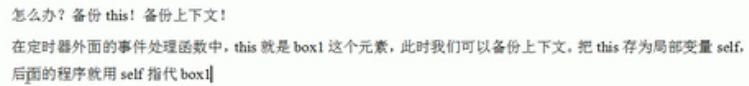






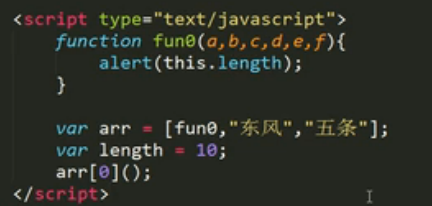
直接调用指定的dom对象或者备份上下文（使用其他非关键词，that=this）





08 函数上下文-规律5

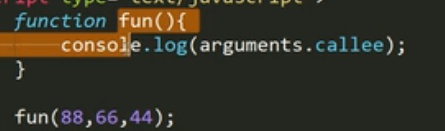




因为数组是object,索引访问调用类似对象.方法



09 函数的上下文小考题





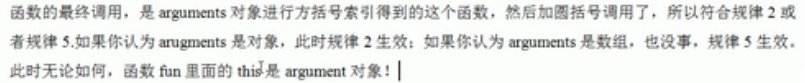
function cc(a,b,c){

}

console.log(cc.length);//函数形参的个数等价于arguments.callee.length

console.log(arguments.length);//指的是实参的个数





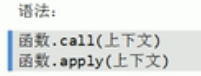
10 call()和apply()



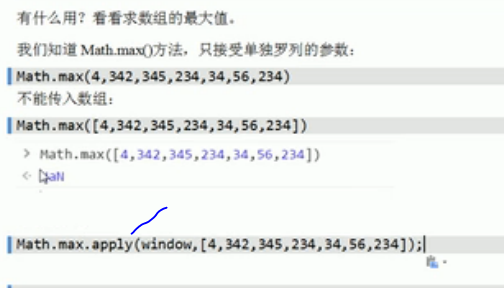
对象冒充，改变函数执行时的上下文。只有函数才能调用call和apply方法（函数的方法）

console.log('call' in cc);//true

console.log('apply' in cc);//true

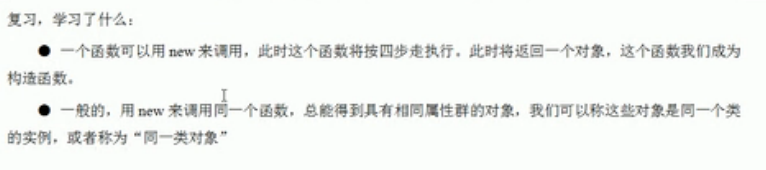






Apply接受数组作为参数并将数组参数的元素作为各个单独的参数解包

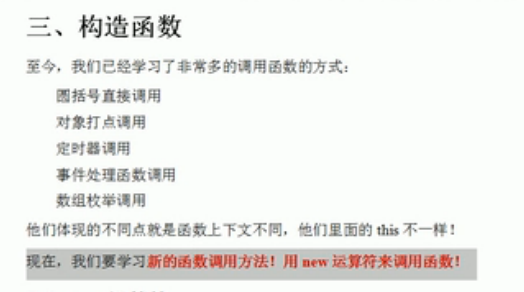


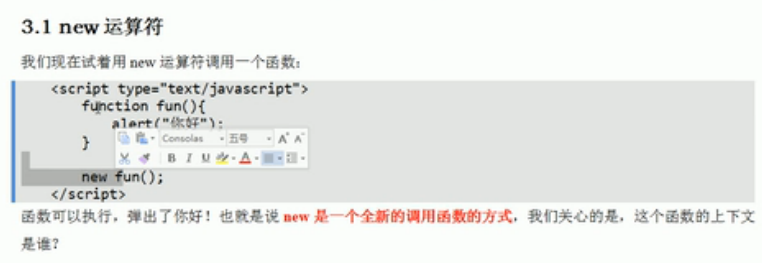


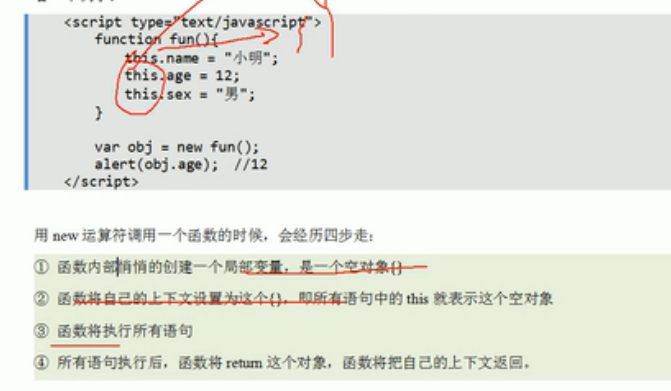
New关键字调用函数，首先创建一个新的对象，然后将当前函数执行的上下文设置为这个对象，并在最后返回这个对象。

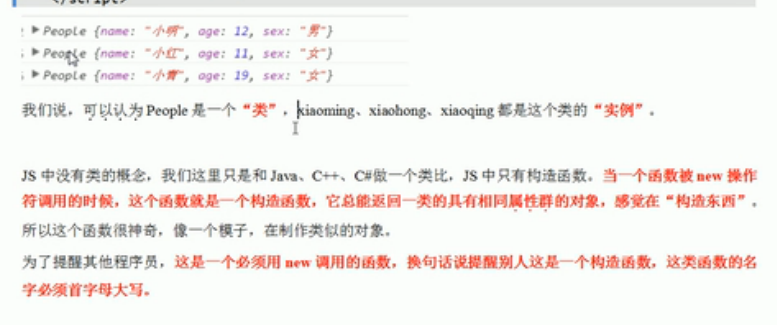
New 就是运算符，不要太有类的概念。













//===========================20170606