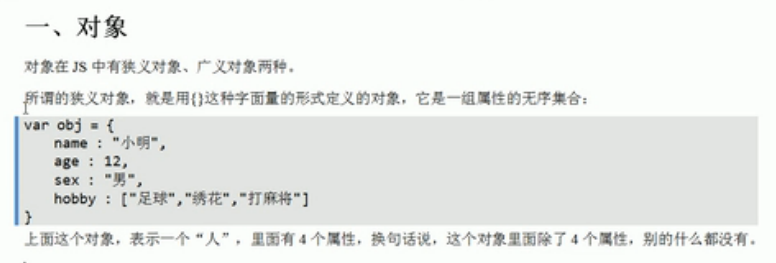
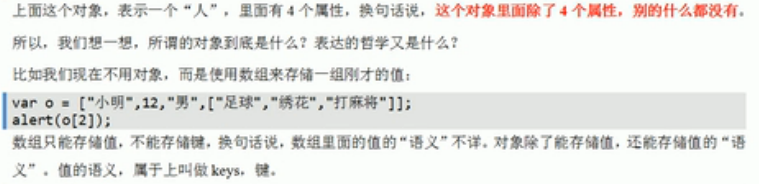
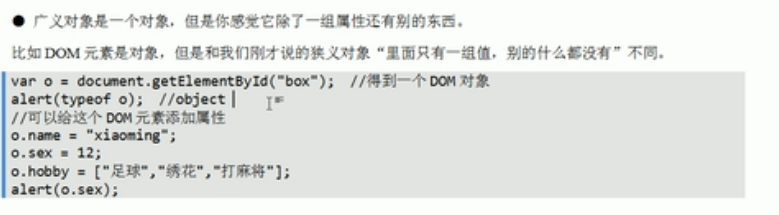
01认识对象



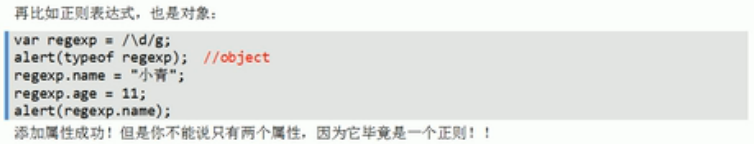
字面量创建的对象，除了显示定义的不包含任何其他的东西

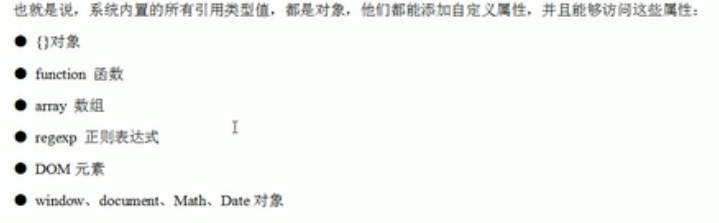








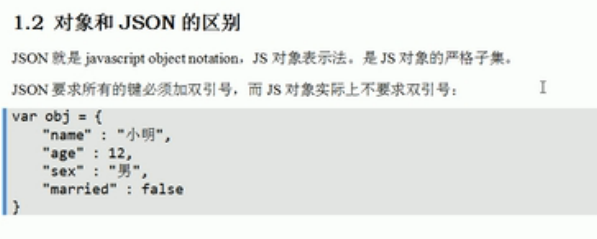


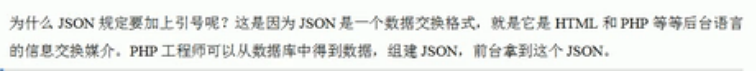


还有基本类型的包装类



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）

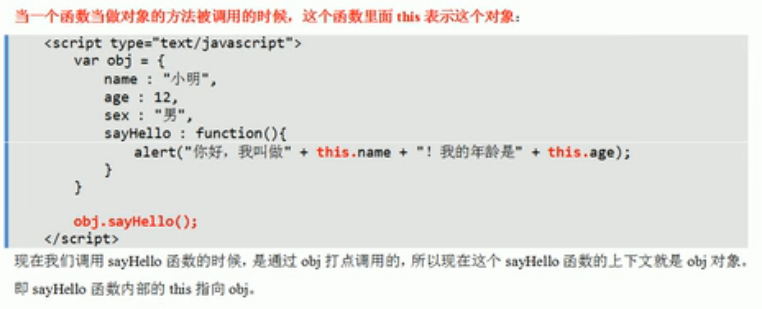


03 对象的方法



上下文在这里指的是函数的this指代的对象



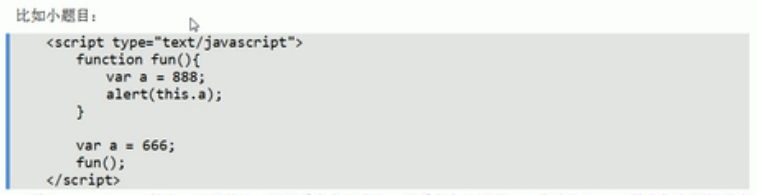




函数的上下文取决于函数的调用而非函数的定义。

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

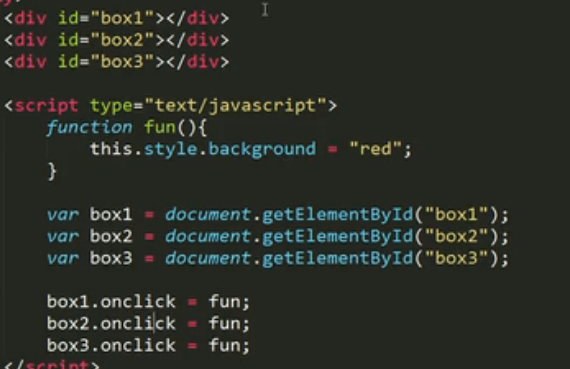






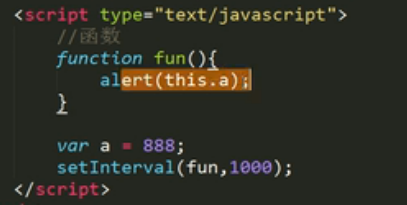
06 函数的上下文-规律3





04 函数的上下文-规律4



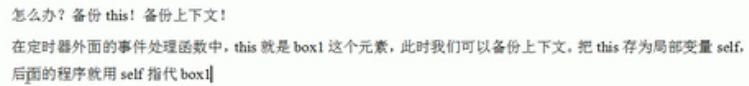






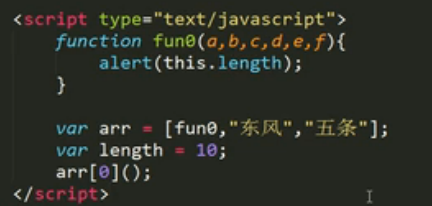
直接调用指定的dom对象或者备份上下文（使用其他非关键词）





08 函数上下文-规律5

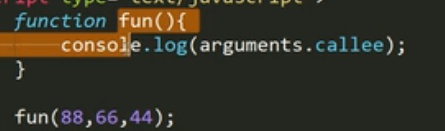




因为数组是object,索引访问调用类似对象.方法



09 函数的上下文小考题





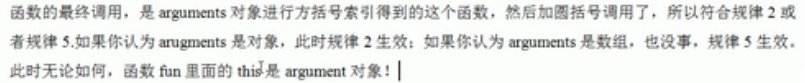
function cc(a,b,c){

}

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





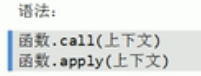
10 call()和apply()



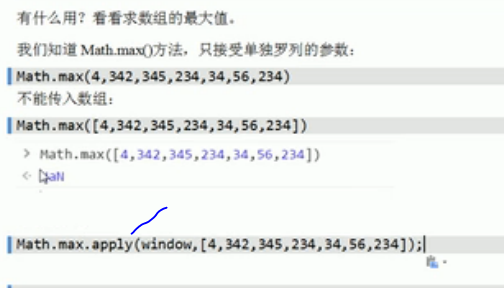
对象冒充，只有函数才能调用call和apply方法（函数的方法）

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

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

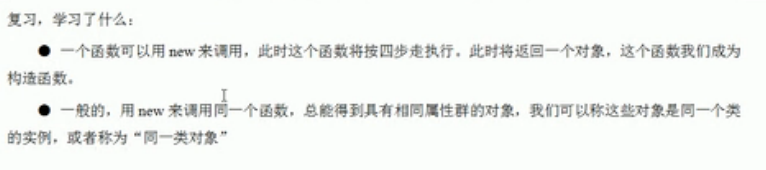






Apply接受数组作为参数并将数组参数的元素作为各个单独的参数解包





New 就是运算符，不要太有类的概念。

Javascript不是一个面向对象（OO）的语言，而是基于OO；



