

闭包可以用在许多地方。它的最大用处有两个，一个是前面提到的可以读取函数内部的变量，另一个就是让这些变量的值始终保持在内存中。

**使用闭包的注意点**

1）由于闭包会使得函数中的变量都被保存在内存中，内存消耗很大，所以不能滥用闭包，否则会造成网页的性能问题，在IE中可能导致内存泄露。解决方法是，在退出函数之前，将不使用的局部变量全部删除。

2）闭包会在父函数外部，改变父函数内部变量的值。所以，如果你把父函数当作对象（object）使用，把闭包当作它的公用方法（Public Method），把内部变量当作它的私有属性（private value），这时一定要小心，不要随便改变父函数内部变量的值。



函数可以用来作为构造函数来使用。另外只有函数才有prototype属性并且可以访问到，但是对象实例不具有该属性，只有一个内部的不可访问的\_\_proto\_\_属性。\_\_proto\_\_是对象中一个指向相关原型的神秘链接。按照标准，\_\_proto\_\_是不对外公开的，也就是说是个私有属性，但是Firefox的引擎将他暴露了出来成为了一个共有的属性，我们可以对外访问和设置。

实例就是通过构造函数创建的。实例一创造出来就具有constructor属性（指向构造函数）和\_\_proto\_\_属性（指向原型对象），

构造函数中有一个prototype属性，这个属性是一个指针，指向它的原型对象。

原型对象内部也有一个指针（constructor属性）指向构造函数:Person.prototype.constructor = Person;

实例可以访问原型对象上定义的属性和方法。

**var** obj = **new** Base();

new操作符具体干了什么呢?其实很简单，就干了三件事情。

**var** obj = {};
obj.\_\_proto\_\_ = Base.prototype;
Base.call(obj);

在检查变量是否存在时，最好选择是使用typeof

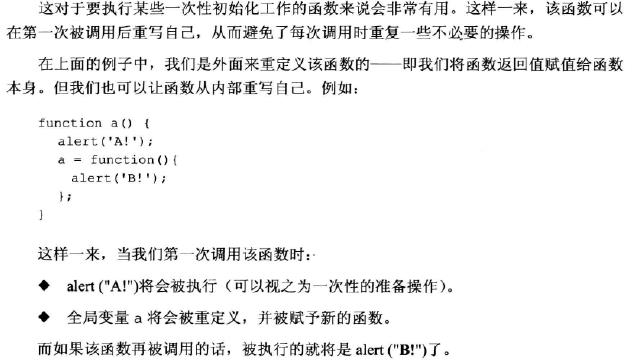
>>if(typeof somevar !== 'undefined'){result = 'yes';}

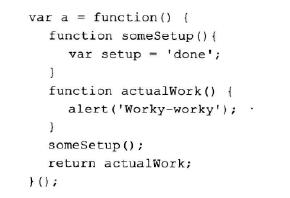
>>result;

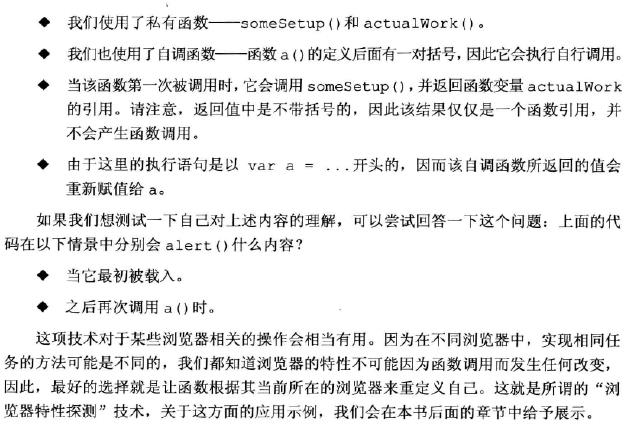
如果直接使用if(somevar){...}如果somevar不存在就会在控制台上打印出somevar is not defined.作为js高手肯定不会多此一举，其次，就算if(somevar)返回的是false,也并不意味着somevar就一定没有意义,它也可以是任何一种被初始化为falsy值(如false或0)的已声明变量.

(function(name){....})();我们只需要将匿名函数的定义放进一对括号中,然后外面再紧跟一对括号即可，第二对括号起到的是“立即调用”的作用，同时它也是我们向匿名函数传递参数的地方。使用自调匿名函数的好处在于这样不会产生任何全局变量。缺点是无法重复执行（除非将它放在某个循环或其他函数中）。最适合执行一些一次性的或初始化的任务。

用新函数覆盖旧函数







如果一个函数需要在其父级函数返回之后留住对父级作用域的链接的话，就必须要为此建立一个闭包。

var n;

function f(){

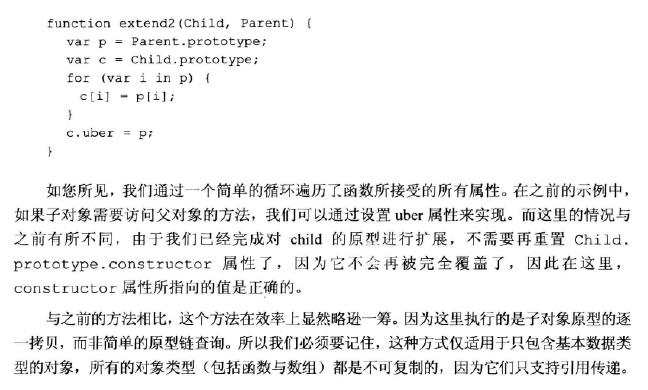
var b = "b";

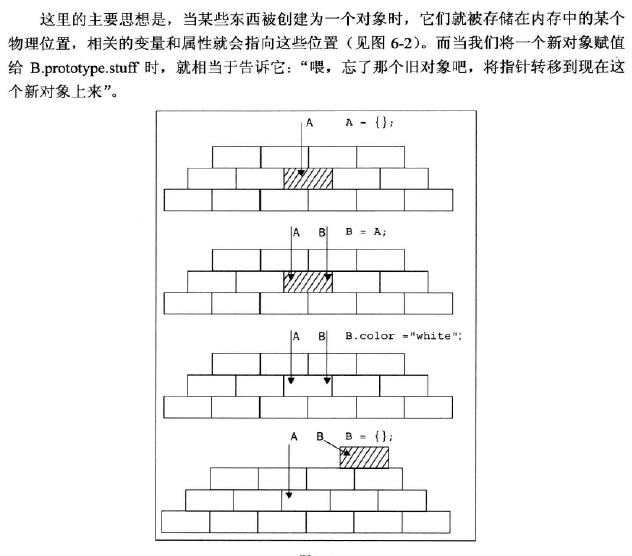
n = function(){return b;}

}

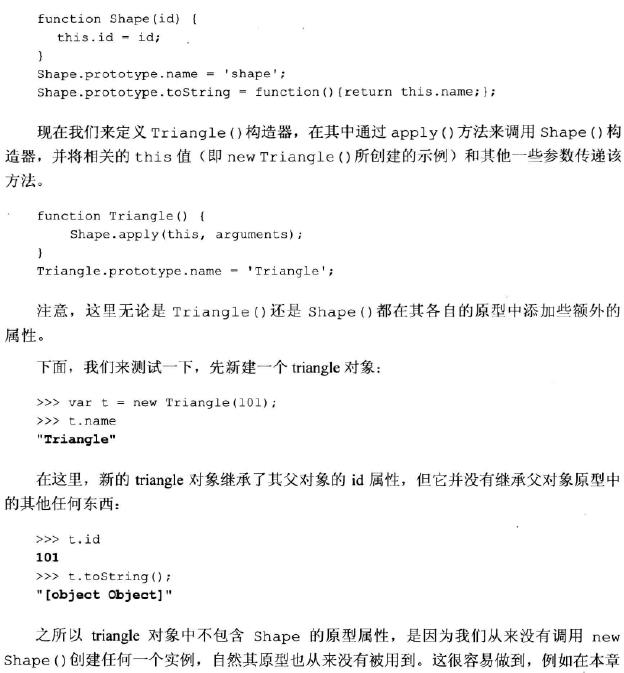
如上例子所示：f是n的父级函数，在f返回之后，n依然可以访问f中的局部变量b.

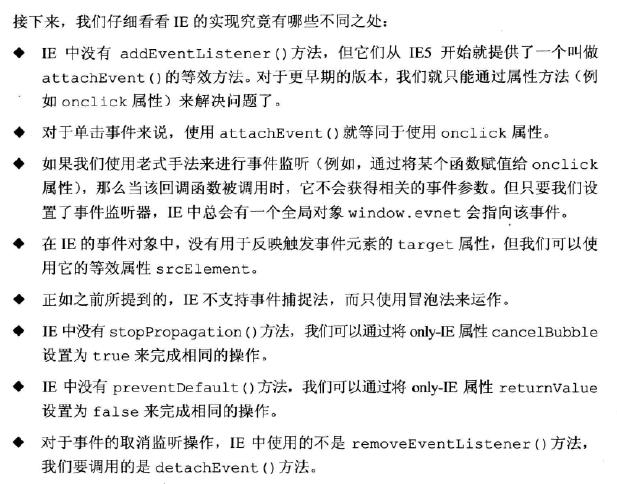
如果所有属性都指向了一个相同的对象，父对象就会受子对象属性的影响，就必须利用某种中介来打破这种连锁关系。我们可以用一个临时构造函数来充当中介。即我们创建一个空函数f(),并将其原型设置为父级构造器。然后我们既可以用new F()来创建一些不包含父对象属性的对象，同时又可以从父对象prototype属性中继承一切。

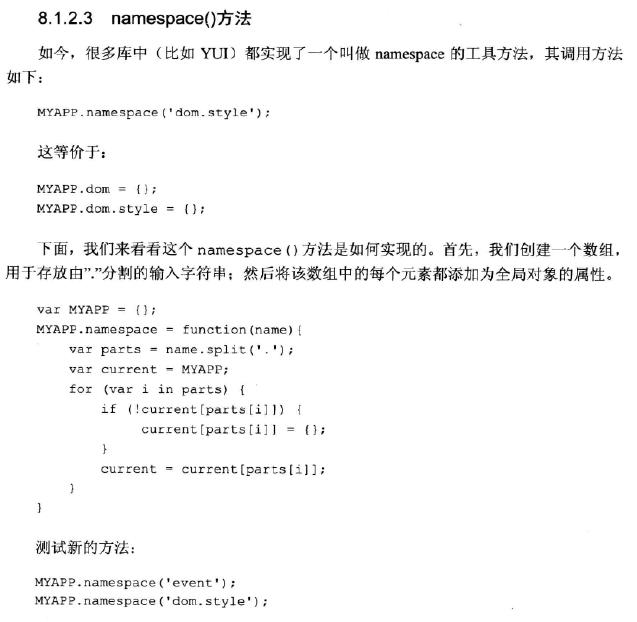


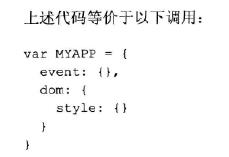


构造器借用:子对象构造器可以通过call()或apply()方法来调用父对象的构造器。









私有函数公有化，用闭包可实现：

  var MYAPP = {};

  MYAPP.dom = (function(){

     var \_setStyle = function(el,prop,value){

        console.log('setStyle');

     };

     var \_getStyle = function(el,prop){

         console.log('getStyle');

     };

     return {

        setStyle: \_setStyle, getStyle:\_getStyle,yetAnother:\_setStyle

     }

  })();

闭包占内存。

块作用域：

在javascript中，变量i是定义在ouputNubmers()的活动对象中的，因此从它有定义开始，就可以在函数内部随处访问它.即使像下面这样错误地重新声明同一变量，也不会改变它的值：

function outputNumbers(count){

    for (var i=0;i<count;i++){

        alert(i);

    }

   var i;

    alert(i);//count

}