**简单的对闭包做一个理解：**

1.闭包产生是：函数内嵌套函数；

2.闭包函数在外面被调用，访问；

3.闭包可以访问外部的变量，但是外部的变量不能访问内部；

4.闭包可以保持对定义时的作用域的引用，使之不被销毁，长生不老。

在看闭包之前我们还需要理解**作用域**。

**作用域主要分为两种：全局变量和局部变量。**

**局部变量:**只能用于定义它函数内部。对于其他的函数或脚本代码是不可用的。

这里定义的n就是局部变量。

function myTest1(){

var n = 3;

return n;

}

console.log(n);//n is not defined

console.log( myTest1());//3

**全局变量：**函数可以访问由函数内部定义的变量。

这里定义的m就是局部变量。

var m = 3;//全局变量

function myTest2(){

return m;

}

console.log(m);//3

console.log( myTest2());//3

从这里我们就可以知道，函数内部可以调用全局变量，但是函数外部无法读取函数内的局部变量。函数内部声明变量的时候，必须使用var命令去声明一个变量。如果不用的话，就会声明了一个全局变量。

function myTest3(){

var j = 4;

k=3;

return j,k;

}

console.log( myTest3());//4,3

console.log(k);//3

console.log(j);//j is not defined

在这里我们j和k都是在函数内部声明的，但是因为j使用了var，而k没有使用var声明，所以在外部能够使用k；也就是所谓的声明了一个全局变量。

在理解了上半部分之后，我们先写一个简单的闭包实例给大家看看：

function myTest4(){

var i = 4;

return function(){

return i;

}

}

var func = myTest4();

console.log(func());//4

在这里我们是不是有点明白了呢：如果我们想用一个变量或者得到一个变量，但是我们又不想全局声明定义这个变量，到这的时候我们就可以通过闭包来调用函数内部的变量。

但是我们也应该知道闭包产生的值是一直保存在内存中的。这里通过一个例子给大家看看：

function myTest5(){

var i = 4;

return function(){

return i++;

}

}

var func = myTest5();

func();//4

func();//5

console.log(func());//6

不知道这个例子大家理解了没，也就是说调用一次，然后i的值就变了一次并且会一直被保存，然后下次调用的时候就是上次保存的值了。

**回收**

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浏览器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

function myTest6(){

var i = 4;

return function(){

return i++;

}

}

var func = myTest5();

console.log(func());//4

func = null;//n被回收

var func = myTest5();

console.log(func());//4

**总结**

闭包的缺点就是常驻内存，闭包会使变量始终保存在内存中，如果使用不当会增大内存使用量，很容易造成内存泄露。

正常情况下外部函数是不能访问内部函数的变量的，但是可以利用闭包来实现对函数内部变量的访问。

一般函数执行完毕后，局部活动对象就被销毁，内存中仅仅保存全局作用域。但闭包的情况不同！

真正深入研究的话肯定不止我这些东西的，我也是看了视频和网上的介绍之后对自己所理解的闭包进行总结！

---------------------

作者：天空影

原文：https://blog.csdn.net/jx950915/article/details/79043087