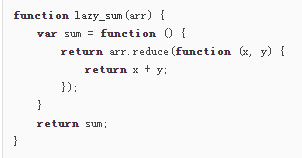
# 闭包

## 函数作为返回值

高阶函数除了可以接受函数作为参数外，还可以把函数作为结果值返回。



当我们调用lazy\_sum()时，返回的并不是求和的结果，而是求和的函数。

IMG_256

调用函数f时，才是真正计算求和的结果。

然后调用f

IMG_256

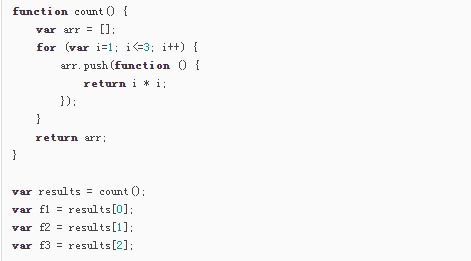
**☆**在这个例子中，我们在函数lazy\_sum中又定义了函数sum，并且函数sum可以引用函数lazy\_sum的参数和局部变量，当lazy\_sum返回函数sum时，相关参数和变量都保存在返回的函数中，这种称为“闭包”的程序结构拥有极大的威力。

同时我们的每次调用lazy\_sum()时，每次调用都会返回一个新的函数，即使传入相同的参数。

***可以理解成函数调用两次。***

## 闭包

使用闭包：返回函数不要引用任何循环变量，或者后续会发生变化的变量。



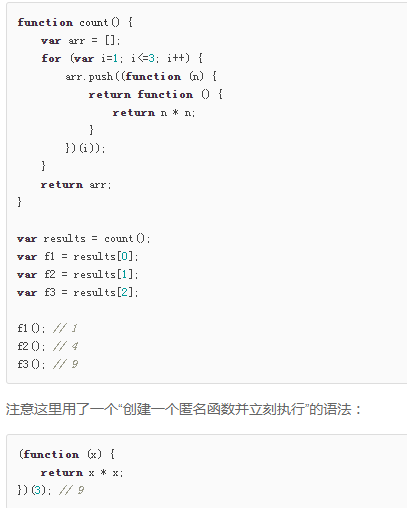
解析



原理：var results=count(); 此时results是一个包含3个函数的数组，这是哪个函数都引用了外部变量i，此时经过循环，外部变量i早已经变成了4，然后在调用这3个函数是，都会返回16.

### 解决方案

再次创建一个函数，用该函数的参数绑定循环变量当前的值，无论该循环白能量后续如何更改，已绑定到函数参数的值不变。



### 作用

借助闭包，同样可以封装一个私有变量，下面是一个计数器。

