函数

函数基础

**定义：**

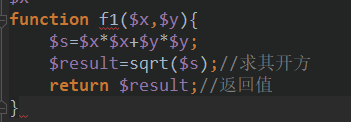
形式 function 函数名（【$形参1】，【$形参2】……）{

//函数体

}

说明：

1. 形参只能是变量，只能在函数内部使用的变量
2. 形参作为变量，其名字“自己定义”

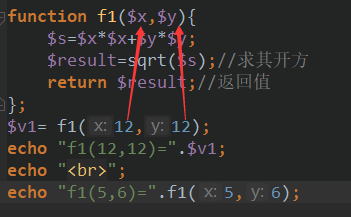


**函数调用：**

函数名（$实参1，$实参2）

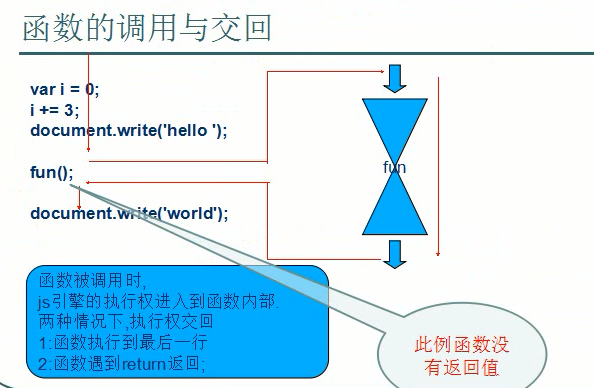
说明：

1. 实参应该跟要调用的函数的形参一一对应
2. 实参就是数据值，可能是直接值，也可能是变量值



实参调用给形参

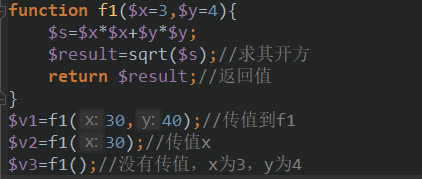
**函数调用详细过程：**

1. **首先，将函数调用时的实参速，传递给函数的形参（变量）**
2. **程序执行过程，进入函数内容可以认为是一个跟外界隔离的独立运行空间；**
3. **在函数内部，按正常的流程顺序，执行其中的代码；**
4. **直到函数结束，则退出该运行空间，而返回到原来调用函数的位置，继续执行后续代码。**
5. **如果在函数内部执行过程中，有return语句，则也会立即终止函数。并回到函数调用位置。**

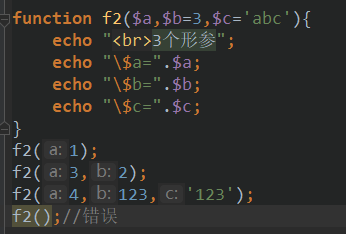
函数的参数问题：

**函数形参的默认值问题**

**我们可以给一个函数定义时的形参，赋值一个“默认值”，则中国函数调用的时候，该形参对应的实参，可以不给值**



函数的形参的默认值，可以只给部分形参数字默认值，但设置默认值性的形参，都要放在“右边”（后边）；



形参的传值问题

一句话，形参的传值问题，其实就是 变量之间的传值问题

其实无非就是实参变量，传值给形参变量问题

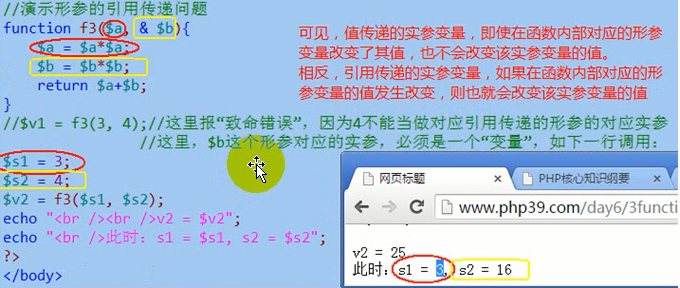
此时，也同样有两种传值方式

值传递

这是默认值，如果没有特别设定，参数传递都是值传递

引用传递

需要在形参前面添加&



函数参数数量问题

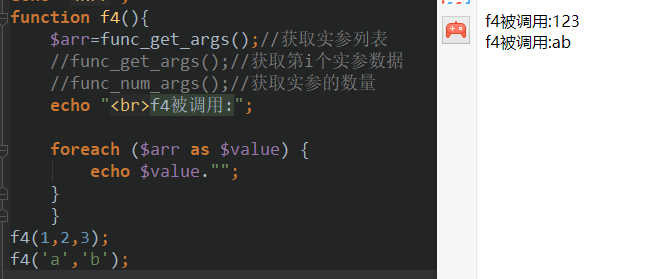
1. 实参数量应该形参数量保持一致
2. 形参有默认值，则对应的实参就可以进行一定程度的省略

省略只能从右往左

1. 不常见的形式，不定义形参，而实参任意给出，其实，系统也有类似的函数 比如：

Var\_dump（）；

func\_get\_args();//获取实参列表  
func\_get\_args();//获取第i个实参数据  
func\_num\_args();//获取实参的数量



# 函数的返回值

一个观念问题：

函数的返回值，不是语法规定，而是应用所需：需要就返回，不需要就不返回

返回值 一定是通过return语句

**形式：**

**Function 函数名（…）{**

**…….**

**Return xx数据**

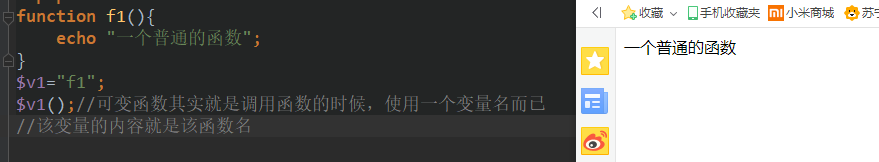
**}**

**注意：return1语句作用，不管后面跟不跟数据，都会立即终止函数使用，返回函数调用的位置并继续后续工作**

# 函数的其他形式

可变变量：一个变量的名字还是变量

可变函数：一个函数的名字是变量



可变函数灵活性



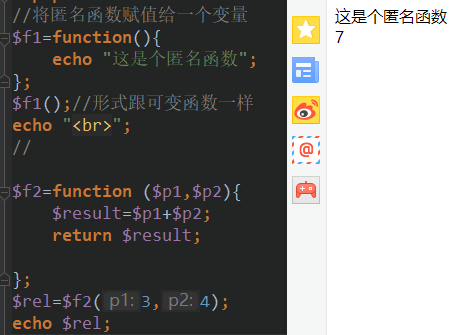
# 匿名函数

就是没有名字的函数。

有两种形式的匿名函数：

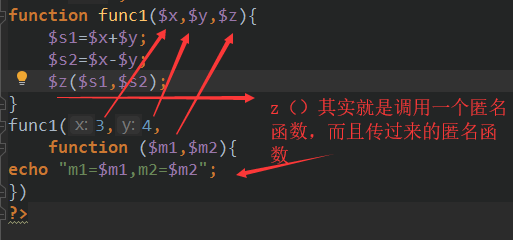
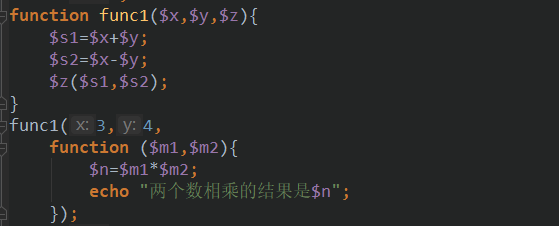
形式1：

将一个匿名函数“赋值”给一个变量——此时该变量就代表匿名函数；



形式2：

是直接将一个匿名函数，当做“实参”来使用！——即调用“别的函数”的时候，使用一个匿名函数来当做实参。自然，在函数A中，也就应该对该函数当做一个函数来用；

Closure闭包

# 作用域

Php中有三种变量作用域；

局部作用域

指一个函数的内部范围

对应这样的变量，就称为“局部变量”

超全局作用域

指一个函数的所有代码范围

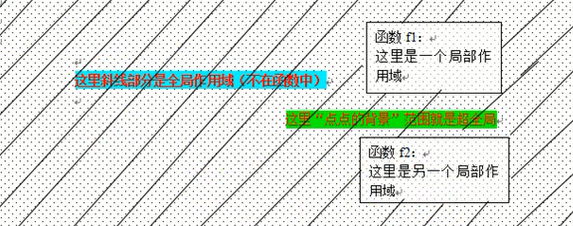
其实只有系统预定义的几个$\_GET,$\_POST,$\_SERVER…..

对应这样的变量，就称为“全1部变量”

全局作用域

不在函数内部的范围——函数外部

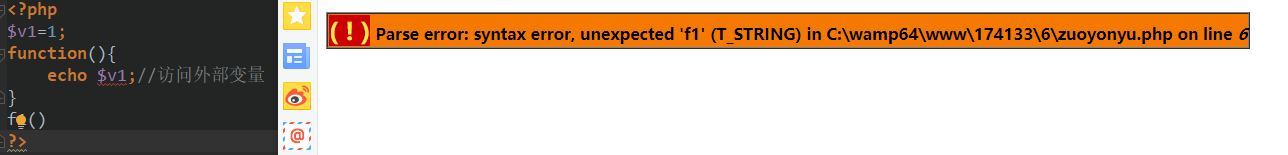
对应这样的变量，就称为“超全局变量”

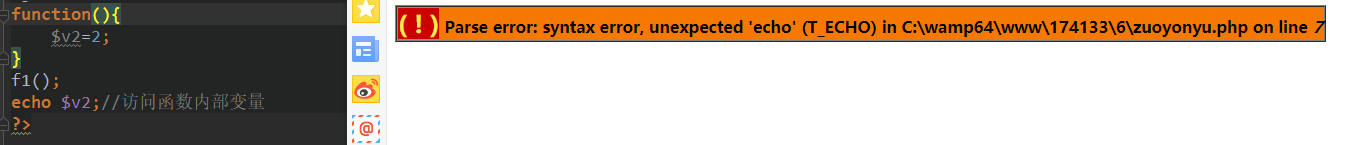


通常：

全局范围不能访问局部变量

局部范围不能访问全局变量





3.函数内部的变量，通常在函数调用后变量会被销毁

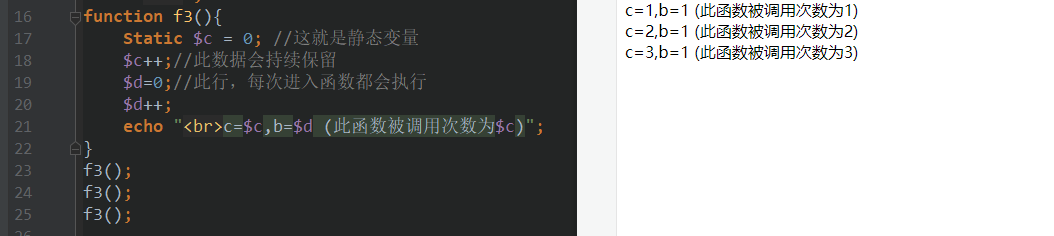
1. 但有一种局部变量，在函数调用结束后不被销毁，他叫做“静态变量”

形式：

Function（）{

Static $变量名 = 初始值 //这就是静态变量

}



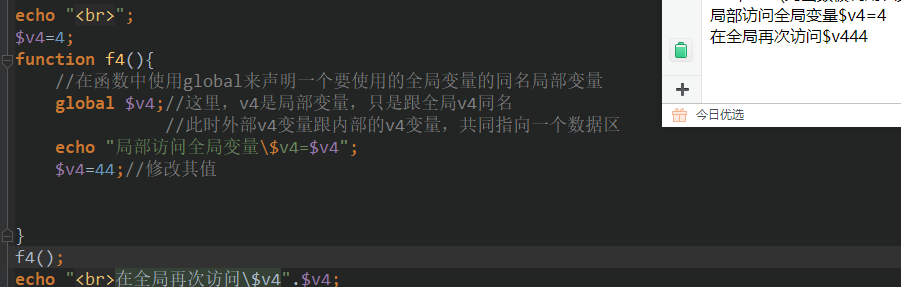
可见，静态变量可以用来统计函数被调用的次数，也可以用于保留函数使用的数据

如何在局部作用域访问全局变量（常见需求）

有两种做法：

做法一：

Global



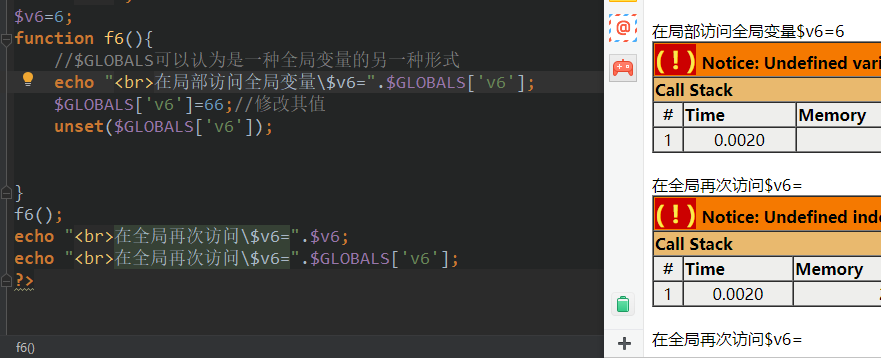
Unset的话也不会影响外部的v4的使用

做法二：

使用$GLOBALS超全局变量



但，如果我们对$GLOBALS变量的某个单元（也即小标），unset,则其就会完全对应销毁该变量；这是因为，$GLOBALS对全局变量的使用可以看做是全局变量的另一种语法形式而已，而不是“引用关系”：举例如下：



此时就是实实在在地销毁了该全局变量