PART11

- **函数**

函数基础：

1.函数的定义：

形式：

function 函数名（【$形参1】【，$形参2】【，....】）{

//函数体。。。。。。

}

说明：

1，定义时使用的形参，其实就是一个变量；只能在该函数内部使用的变量

2，形参作为变量，其名字是“自己定义”；自然应该遵循命名规范；

eg：

```php
<?php
header('Content-type:text/html; charset=utf-8');
?>
<?php
function test1($x, $y){
    $s = $x * $x + $y * $y;
    $result = sqrt($s); //开方
    return $result;//返回该数据值
}

?>
```

2.函数的调用：

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（比如5，”abc”），也可能是变量值(比如$v1)

eg：

```php
<?php
header('Content-type:text/html; charset=utf-8');
?>
<?php
function test1($x, $y){
    $s = $x * $x + $y * $y;
    $result = sqrt($s); //开方
    return $result;//返回该数据值
}
$v1 = test1(3,4);
echo "v1 = $v1";
echo "<br>";
echo "v2 =  " . test1(5,12);
?>
```

```
v1 = 5
v2 = 13
```

3.函数调用的详细过程：

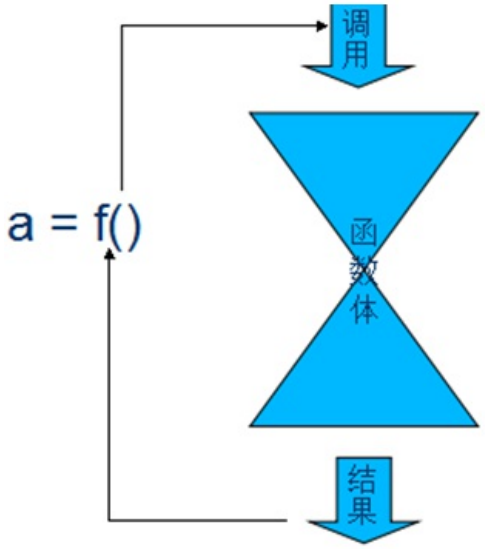
1，首先，将函数调用时的实参数据，传递（赋值）给函数的形参（变量）；

2，程序的执行流程，进入到函数内部——此时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，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的过程中，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**

1.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”，则这个函数调用的时候，该形参对应的实参，可以不给值。
eg：

```php
<?php
function test2($x = 3, $y = 4){
    $s = $x * $x + $y * $y;
    $result = sqrt($s); //开方
    return $result;//返回该数据值
}
$v1 = test2(30,40);//传过去2个数据分别赋值给x,y;
$v2 = test2(30);//传值去1个数据，给予x，y自动获得默认值4
$v3 = test2();//没有传值过去，x自动获得3，y自动获得4

echo "<br>";
echo "<hr>";
echo "v1 = $v1";
echo "<br>";
echo "v2 =  $v2" ;
echo "<br>";
echo "v3 =  $v3" ;
?>
```

```
v1 = 50
v2 = 30.265491900843
v3 = 5
```

函数形参的默认值，可以只给部分形参设置默认值，但设置默认值性的形参，都要放在“右边”（后边）：
eg：

```php
<?php
echo "<br>";
echo "<hr>";
function test3($x, $y = 4, $z = "zx"){
echo "<br/> x= $x , y=$y, z= $z";
}
test3(1);
test3(1,2);
test3(1,3,'cool');
test3();//错误的写法
?>
```

```
x= 1 , y=4, z= zx
x= 1 , y=2, z= zx
x= 1 , y=3, z= cool
Warning: Missing argument 1 for test3(), called in E:\wamp\zhandian\function.php on line 46 and defined in E:\wamp\zhandian\function.php on line 40

Notice: Undefined variable: x in E:\wamp\zhandian\function.php on line 41

x= , y=4, z= zx
```

2.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，传值给形参变量的问题。

即：此时，也同样有两种传值方式：

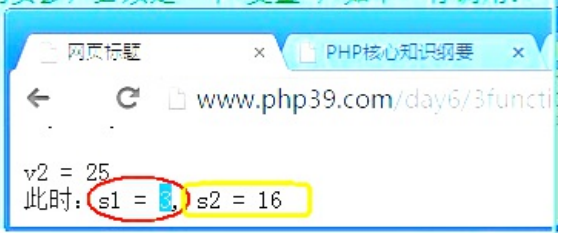
值传递：这是默认值。如果没有特别设定，参数传值都是值传递。

引用传递：需要在形参的前面加 &符号

例子：

```php
//演示形参的引用传递问题
function f3($a, & $b){
    $a = $a*$a;
    $b = $b*$b;
    return $a+$b;
}
//$v1 = f3(3, 4);//这里报“致命错误”，因为4不能当做对应引用传递的形参的对应实参
                 //这里，$b这个形参对应的实参，必须是一个“变量”，如下一行调用：
$s1 = 3;
$s2 = 4;
$v2 = f3($s1, $s2);
echo "<br /><br />v2 = $v2";
echo "<br />此时：s1 = $s1, s2 = $s2";
?>
</body>
```

可见，值传递的实参变量，即使在函数内部对应的形参变量改变了其值，也不会改变该实参变量的值。
相反，引用传递的实参变量，如果在函数内部对应的形参变量的值发生改变，则也就会改变该实参变量的值



3.参数的数量问题：

1，通常，函数调用时的实参数量，应该跟函数定义时的形参数量保持一致。

2，如果函数定义时，形参有默认值，则对应的实参就可以进行一定程度的省略：注意：省略只能从右往左进行。

3，有一种定义和使用函数的特别形式（并不常见）：它不定义形参，而实参任意给出。其实，系统中也有类似的函数：，比如：

var_dump($v1);

var_dump($v1, $v2, $v3);

可见，该函数就可以接受任意个数的实参；

我们自己也可以定义这种函数。其实，这种函数，依赖的是以下3个系统函数来获取相应的信息，以得到实参数据的处理：

1: func_get_args(); //获取实参数据列表，成为一个数组

2: func_get_arg($i); //获取第$i个实参数据，$i从0开始算起；

3: func_num_args(); //获取实参的数量（个数）

eg：

```php
<?php
echo "<br>";
echo "<hr>";
function test4(){
    //函数func_get_arg( )可以获取函数调用时传递过来的所有实参参数。
 $arr = func_get_args( );
 echo "<p>函数test4被调用其参数为：";
 foreach ($arr as  $value) {
    echo $value . " ";
 }
}
test4(1,2,3,4);
test4('zx','cool');

?>
```

```
函数test4被调用其参数为：1 2 3 4

函数test4被调用其参数为：zx cool
```

4.函数的返回值问题

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，不需要就无需返回。

返回值，一定是通过return语句！

形式：

function 函数名(....)

{

//。。。。。。

return XX数据;

}

注意：

return语句的作用，不管后面跟不跟数据值，都会立即终止函数的执行，返回到函数调用的位置并继续后续工作。

5.函数的其他形式：

可变函数

先想想可变变量：

$v1 =" abc" ;

$abc = 123;

echo $$v1; //输出123，这就是可变变量。

可变变量：一个变量的名字还是一个变量！

可变函数：一个函数的名字是一个变量！

eg:

```php
<?php
echo "<br>";
echo "<hr>";
function f1(){
echo "<br/>123 ";
}
$v1 = "f1";
$v1();//这个就是可变函数;

?>

<?php
echo "<br>";
echo "<hr>";
function jpg(){echo "<br/>处理jpg图片";}
function gif(){echo "<br/>处理gif图片";}
function png(){echo "<br/>处理png图片";}
$file = "abc.jpg";//代表用户上传的图片，其后缀可能是png，gif，jpg等；
$houzhui = strrchr($file,".");//strrchr($s1,$s2)函数用于获取字符串$s1中最后一次出现字符$s2之后的所有字符内容（含$s2本身）
$houzhui = substr($houzhui, 1);//获得该字符串从位置1开始之后的所有字符
echo "<br/> $houzhui";

?>
```

123

jpg

6.匿名函数：

定义：匿名函数就是没有名字的函数。

有2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——此时该变量就代表该匿名函数了！

eg：

```php
<?php
echo "<br>";
echo "<hr>";
$f11 = function(){
echo "<br/>这是一个匿名函数！ ";
};
$f11();

$f21 = function($p1,$p2){
    $ressult1 = $p1 + $p2;
    return $ressult1;
};
$re1 = $f21(3,4);
echo "<br/>rel = $re1";
?>
```

这是一个匿名函数!

rel = 7

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A”的时候，使用一个匿名函数来当做实参。自然，在该函数A中，也就应该对该匿名函数当做一个函数来用！