函数

使用 function 关键字，函数名，形参。

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；

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)

```
function f1($x, $y){
    $s = $x * $x + $y * $y;
    $result = sqrt($s); //求其开方
    return $result; //返回该数据值
}
```

函数调用详细过程

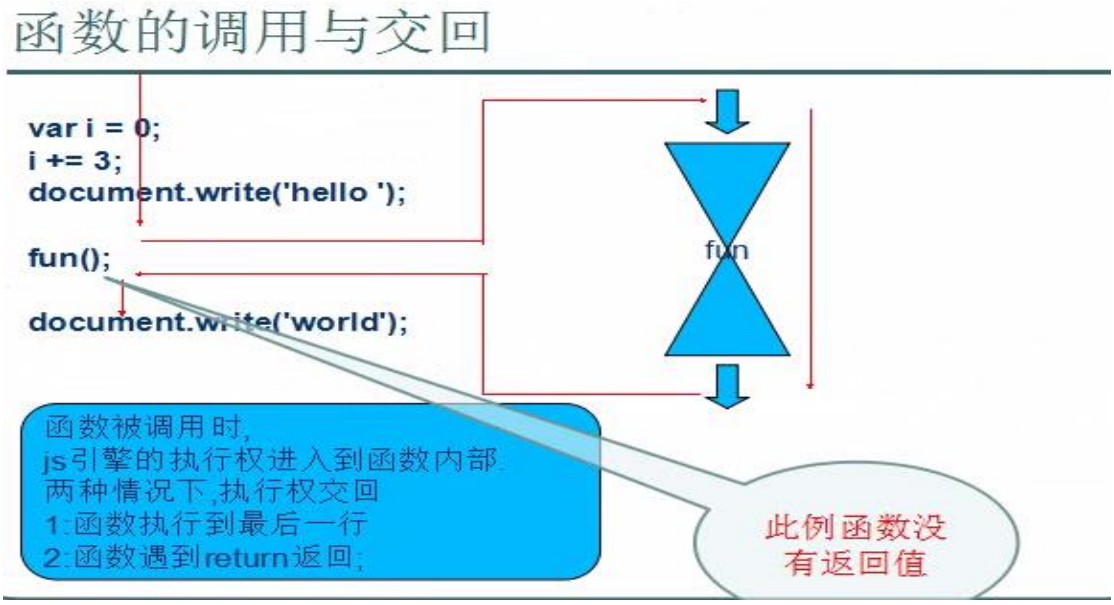
1，首先，将函数调用时的实参数据，传递（赋值）给函数的形参（变量）；

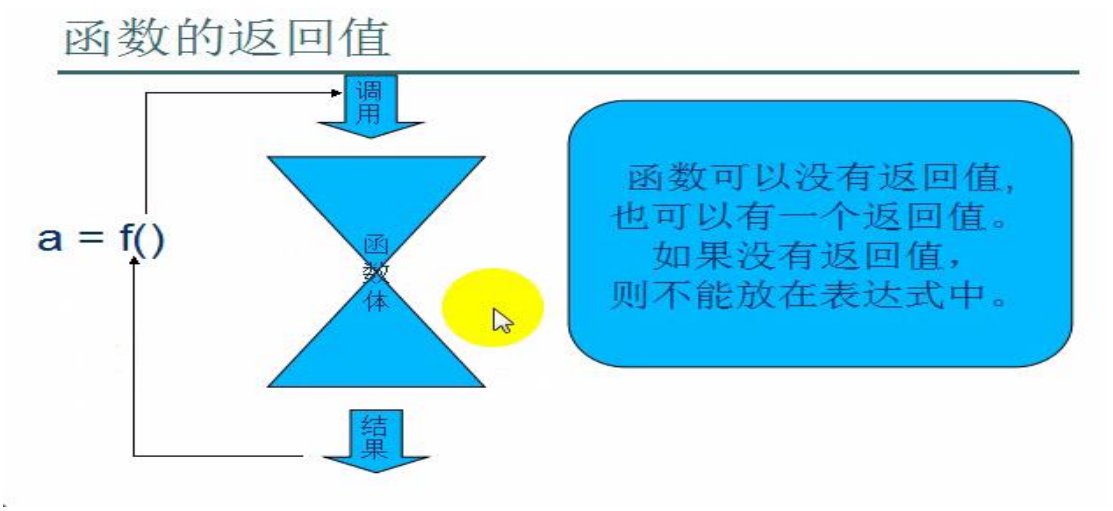
2，函数的执行流程，进入到函数内部----此时可以认为是一个跟外界“隔离”的“独立运行空间”。

3，在函数内部，则退出流程顺序，执行其中的代码；

4，直到函数结束，则提出该运行空，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。（有 return 语句立即终止，空的 return 也算）





函数的参数问题

函数形参的默认值问题

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。

```
function f1($x = 3, $y = 4){
    $s = $x * $x + $y * $y;
    $result = sqrt($s); //求其开方
    return $result; //返回该数据值
}
$v1 = f1(30, 40);    //传过去2个数据分别给予x和y
$v2 = f1(30);        //传值去1个数据，给予x，y自动获得默认值4
$v3 = f1(); //没有传值过去，x自动获得3，y自动获得4
```

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”（后面）。

```php
function f2($a, $b=3, $c = 'abc'){
    echo "<br />这是只是演示多个形参，部分有默认值情况";
    echo "<br />a=$a, b=$b, c=$c";
}
f2(1);
f2(1,2);
f2(1,2, 'xyz');
f2();   //这种做法是错误的！
```

形参的传值问题

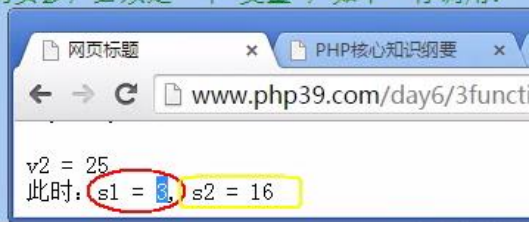
形参的传值问题，其实就是“变量之间的传值问题”：实参变量传值给形参变量的问题。

值传递：无特别设定，参数传值都是值传递。

引用传递：需要在形参的面前加&符号。

```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



函数参数的数量问题

1，通常，实参数量应该跟形参数量保持一致。
2，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（不常见）：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);    //ok!
```

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：

```
//系统函数func_get_args()可以获取函数调用时传递过来的
//所有实参数据，并且都放入一个数组中！
```

```
function f4(){
    //系统函数func_get_args()可以获取函数调用时传递过来的
    //所有实参数据，并且都放入一个数组中！
    $arr = func_get_args();
    echo "<p>函数f4被调用，其实参为：";
    foreach($arr as $value){
        echo $value . " ";
    }
}
f4(1, 2, 3);
f4('aa', 'bb');
```

这种函数依赖的是三个系统函数:

```
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）
```

函数返回值问题

函数的返回值不是语法规定，是应用所需；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。

函数的其它形式：

可变函数

先想想可变变量：

```
$v1 ="abc";
$abc = 123;
```

```
echo  $$v1;  //输出 123，这就是所谓的可变变量。
```

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！

```
function f1(){
    echo "<br />这是一个普通的函数而已！";
}
$v1 = "f1";
$v1();  //这就是可变函数！
//可变函数其实就是在调用函数的时候，使用一个变量名而已。
//该变量的内部，就是该函数名！
```

```
function jpg(){ echo "<br />处理jpg图片";}
function png(){ echo "<br />处理png图片";}
function gif(){ echo "<br />处理gig图片";}
$file = "abc.png";//代表用户上传的图片；
                  //其后缀肯能是png，jpg，gif等
$houzhui = strrchr($file, ".");
    //strrchr($s1,$s2)函数用于获取字符串$s1中最后一次
    //出现的字符$s2之后的所有字符内容（含$s2本身）
echo "<br />$houzhui";
```

```
//echo "<br />$houzhui";
$houzhui = substr($houzhui,1);//获得该字符串从位置1开始之后的所有字符
$houzhui(); //可变函数！
```

匿名函数

两种形式匿名函数：

1，将一个匿名函数“赋值”给一个变量，------此时变量就代表该匿名函数。

```
<?php
//将一个匿名函数，赋值给一个变量f1
$f1 = function (){
    echo "<br />这是一个匿名函数！";
};
$f1(); //调用该匿名函数，就使用该变量
       //可见其形式跟调用可变函数一样！
```

```php
//在演示一个带参数的匿名函数：
$f2 = function ($p1, $p2){
    $result = $p1 + $p2;
    return $result;
}
$re1 = $f2(3,4);     //将函数的返回值赋值给$re1;
```

2，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！

```php
$f1 = function ($m1, $m2){
    echo "m1=$m1, m2 = $m2";
};
function func1( $x, $y, $z ){
    $s1 = $x+$y;
    $s2 = $x-$y;
    $z($s1, $s2);
}
```

应该为

```php
function func1( $x, $y, $z ){
    $s1 = $x+$y;
    $s2 = $x-$y;
    $z($s1, $s2);
}

func1(3, 4, function ($m1, $m2){
    echo "m1=$m1, m2 = $m2";
});
```

```php
function func1( $x, $y, $z ){
    $s1 = $x+$y;
    $s2 = $x-$y;
    $z($s1, $s2);
}

func1(3, 4,
    function ($m1, $m2){
        $n = $m1 * $m2;
        echo "<br />两个数是和乘以2个数的差的结果为：$n";
    }
);
```

此时，执行这个$z()函数，做的事情就跟前一次不一样了！