Static（静态）关键字[¶](https://www.php.net/manual/zh/language.oop5.static.php" \l "language.oop5.static)

声明类属性或方法为静态，就可以不实例化类而直接访问。静态属性不能通过一个类已实例化的对象来访问（但静态方法可以）。

为了兼容 PHP 4，如果没有指定[访问控制](https://www.php.net/manual/zh/language.oop5.visibility.php)，属性和方法默认为公有。

由于静态方法不需要通过对象即可调用，所以伪变量 $this 在静态方法中不可用。

静态属性不可以由对象通过 -> 操作符来访问。

用静态方式调用一个非静态方法会导致一个 E\_STRICT 级别的错误。

就像其它所有的 PHP 静态变量一样，静态属性只能被初始化为文字或常量，不能使用表达式。所以可以把静态属性初始化为整数或数组，但不能初始化为另一个变量或函数返回值，也不能指向一个对象。

自 PHP 5.3.0 起，可以用一个变量来动态调用类。但该变量的值不能为关键字 *self*，*parent* 或 *static*。

Example #1 静态属性示例

<?php  
class Foo  
{  
    public static $my\_static = 'foo';  
  
    public function staticValue() {  
        return self::$my\_static;  
    }  
}  
  
class Bar extends Foo  
{  
    public function fooStatic() {  
        return parent::$my\_static;  
    }  
}  
  
  
print Foo::$my\_static . "\n";  
  
$foo = new Foo();  
print $foo->staticValue() . "\n";  
print $foo->my\_static . "\n";      // Undefined "Property" my\_static   
  
print $foo::$my\_static . "\n";  
$classname = 'Foo';  
print $classname::$my\_static . "\n"; // As of PHP 5.3.0  
  
print Bar::$my\_static . "\n";  
$bar = new Bar();  
print $bar->fooStatic() . "\n";  
?>  
   </programlisting>  
  </example>  
  
  <example>  
   <title>静态方法示例</title>  
    <programlisting role="php">  
<![CDATA[  
<?php  
class Foo {  
    public static function aStaticMethod() {  
        // ...  
    }  
}  
  
Foo::aStaticMethod();  
$classname = 'Foo';  
$classname::aStaticMethod(); // 自 PHP 5.3.0 起  
?>

**静态变量只存在于函数作用域内，也就是说，静态变量只存活在栈中。一般的函数内变量在函数结束后会释放，比如局部变量，但是静态变量却不会。就是说，下次再调用这个函数的时候，该变量的值会保留下来。【不会受重新赋值影响，包括局部赋值和全局赋值】在同样的方法或者类的局部保存该变量值；**

static用法如下：

**1.static 放在函数内部修饰变量**

**2.static放在类里修饰属性，或方法**

**3.static放在类的方法里修饰变量**

**4.static修饰在全局作用域的变量**

所表示的不同含义如下：

**1.在函数执行完后，变量值仍然保存**

如下所示：

<?php  
function test(){  
  
 static $num=1;  
 echo $num."<br>";  
 return $num++;  
}  
  
echo "ok1"."<br>";  
echo test()."<br>";  
  
$num=1;  
echo "ok2"."<br>";  
echo test()."<br>";  
  
$num=1;  
echo "ok3"."<br>";  
echo test()."<br>";  
?>

[复制代码](javascript:void(0);)

**2.修饰类的属性或方法，可以通过类名访问，如果是修饰的是类的属性，保留值**

如下所示：

[复制代码](javascript:void(0);)

<?php

class Person {

static $id = 0;

function \_\_construct() {

self::$id++;

}

static function getId() {

return self::$id;

}

}

echo Person::$id; //output 0

echo "<br/>";

$p1=new Person();

$p2=new Person();

$p3=new Person();

echo Person::$id; //output 3

?>

[复制代码](javascript:void(0);)

**3.修饰类的方法里面的变量，不论是静态方法还是对象方法**

如下所示：

[复制代码](javascript:void(0);)

<?php

class Person {

static function tellAge() {

static $age = 0;

$age++;

echo "The age is: $age";

}

}

echo Person::tellAge(); //output 'The age is: 1'

echo Person::tellAge(); //output 'The age is: 2'

echo Person::tellAge(); //output 'The age is: 3'

echo Person::tellAge(); //output 'The age is: 4'

[复制代码](javascript:void(0);)

**4.修饰全局作用域的变量，没有实际意义（存在着作用域的问题，**[详情查看](http://www.cnblogs.com/article/detail/217" \o "PHP之static静态变量详解（一）" \t "_blank)**）**

如下所示：

<?php

static $name = 1;

$name++;

echo $name;

?>

另外：考虑到PHP变量作用域

[复制代码](javascript:void(0);)

<?php  
class Person {  
 static *$id* = 0;  
  
 function \_\_construct() {  
 self::*$id*++;  
 }  
  
 static function getId() {  
 return self::*$id*;  
 }  
}  
class Person1 {  
 static *$id* = 0;  
  
 function \_\_construct() {  
 self::*$id*++;  
 }  
  
 static function getId() {  
 return self::*$id*;  
 }  
}  
echo Person::*$id*; //output 0  
echo "<br/>";  
  
$p1=new Person1();  
echo Person::*$id*; //output 0  
echo "<br/>";  
  
$p2=new Person();  
echo Person::*$id*; //output 1  
echo "<br/>";  
  
$p3=new Person1();  
echo Person::*$id*; //output 1  
?>

[复制代码](javascript:void(0);)

可以看出：这3个变量是不相互影响的，另外，**PHP里面只有全局作用域和函数作用域，没有块作用域**

如下所示：

[复制代码](javascript:void(0);)

<?php

include 'ChromePhp.php';

$age = 0;

$age++;

for ($i=0; $i<10; $i++) {

$age++;

}

ChromePhp::log($i); //output 10;

ChromePhp::log($age); //output 11;