PHP开发手册

http://www.php1.cn/

http://www.php100.com/

http://www.oschina.net/

https://github.com/

**变量：**

**PHP 有三种不同的变量作用域：**

**local（局部）**

**global（全局）**

**static（静态）**

**Local 和 Global 作用域**

**函数之外声明的变量拥有 Global 作用域，只能在函数以外进行访问。**

**函数内部声明的变量拥有 LOCAL 作用域，只能在函数内部进行访问。**

如果要在内部访问外部的变量，则使用global来初始化声明。

global 关键词用于访问函数内的全局变量。

$x=5;

$y=10;

function myTest() {

**global** $x,$y;

$y=$x+$y;

}

myTest();

echo $y; // 输出 15

PHP 同时在名为 **$GLOBALS[index]** 的数组中存储了所有的全局变量。下标存有变量名。这个数组在函数内也可以访问，并能够用于直接更新全局变量。

常量：与变量不同，常量贯穿整个脚本是自动全局的，前面没有 $ 符号

设置 PHP 常量

方法一：使用 define() 函数 - 它使用三个参数：

首个参数定义常量的名称

第二个参数定义常量的值

可选的第三个参数规定常量名是否对大小写敏感。默认是 false。

方法二：采用const,例：

const GREETING = 'Welcome to ZBJ--by const';

echo GREETING;

**Use：获取上层作用域变量：**

<?php

function getMoney() {

$rmb = 1;

$dollar = 6;

$func = function() **use** ( $rmb ) {

echo $rmb."\n";

echo $dollar;

};

$func();

}

getMoney();//输出：1

//报错，找不到dorllar变量

?>

**函数变量：字符串函数：**

<?php

function go($name = "jack") { //定义go函数

echo $name."走了<p>";

}

function back($string) //定义back函数

{

echo "又回来了，$string<p>";

}

function callBack($callback, $param) {

$callback($param);

}

$func = "go"; //给变量赋值

$func("Tom"); //使用变量函数来调用函数go()

$func = "back"; //重新给变量赋值

$func("Lily"); //使用变量函数来调用函数back();

callBack('go', 'Tom'); //等同于 go("Tom")

?>

**function\_exists：检查函数是否存在，因为调用不存在的函数会报错，funName ? True : false也不行**

**字符串**

**strlen — 获取字符串长度**

**trim — 去除字符串首尾处的空白字符（或者其他字符）**

**join 、implode — 将一个一维数组的值转化为字符串**

**explode — 使用一个字符串分割另一个字符串**

**substr — 返回字符串的子串**

**substr\_replace — 替换字符串的子串**

**md5 — 计算字符串的 MD5 散列值**

**str\_replace — 子字符串替换**

**strip\_tags — 从字符串中去除 HTML 和 PHP 标记**

**htmlspecialchars — 将字符串中一些字符转换为HTML实体**

**strcmp — 二进制安全字符串比较**

**stripos — 查找字符串首次出现的位置（不区分大小写）**

**strrev — 反转字符串**

**strtolower — 将字符串转化为小写**

**strtoupper — 将字符串转化为大写**

**数组（基本已array开头）**

**array\_values($arr); 获得数组的值**

**array\_keys($arr); 获得数组的键名**

**array\_flip($arr); 数组中的值与键名互换（如果有重复前面的会被后面的覆盖）**

**in\_array("apple",$arr); 在数组中检索apple**

**array\_search("apple",$arr); 在数组中检索apple ，如果存在返回键名**

**array\_key\_exists("apple",$arr); 检索给定的键名是否存在数组中**

**isset($arr[apple]): 检索给定的键名是否存在数组中**

**数组的分段**

**array\_slice($arr,0,3); 可以将数组中的一段取出，此函数忽略键名**

**array\_splice($arr,0,3，array("black","maroon")); 可以将数组中的一段取出，与上个函数不同在于返回的序列从原数组中删除**

**分割多个数组**

**array\_chunk($arr,3,TRUE); 可以将一个数组分割成多个，TRUE为保留原数组的键名**

**数组的填充**

**array\_pad($arr,5,'x'); 将一个数组填补到制定长度**

**三、数组与栈**

**array\_push($arr,"apple","pear"); 将一个或多个元素压入数组栈的末尾（入栈），返回入栈元素的个数**

**array\_pop($arr); 将数组栈的最后一个元素弹出（出栈）**

**四、数组与列队**

**array\_shift($arr);数组中的第一个元素移出并作为结果返回（数组长度减1，其他元素向前移动一位，数字键名改为从零技术，文字键名不变）**

**array\_unshift($arr,"a",array(1,2));在数组的开头插入一个或多个元素**

**五、数组的排序**

**通过元素值对数组排序**

**sort($arr); 由小到大的顺序排序（第二个参数为按什么方式排序）忽略键名的数组排序**

**rsort($arr); 由大到小的顺序排序（第二个参数为按什么方式排序）忽略键名的数组排序**

**usort($arr,"function"); 使用用户自定义的比较函数对数组中的值进行排序（function中有两个参数，0表示相等，正数表示第一个大于第二个，负数表示第一个小于第二个）忽略键名的数组排序**

**asort($arr); 由小到大的顺序排序（第二个参数为按什么方式排序）保留键名的数组排序**

**arsort($arr); 由大到小的顺序排序（第二个参数为按什么方式排序）保留键名的数组排序**

**uasort($arr,"function"); 使用用户自定义的比较函数对数组中的值进行排序（function中有两个参数，0表示相等，正数表示第一个大于第二个，负数表示第一个小于第二个）保留键名的数组排序**

**六、其他常用数组函数**

**array\_merge($arr1,$arr2); 合并两个或多个数组（相同的字符串键名，后面的覆盖前面的，相同的数字键名，后面的不会做覆盖操作，而是附加到后面）**

**range(0,12); 创建一个包含指定范围单元的数组**

**array\_unique($arr); 移除数组中重复的值，新的数组中会保留原始的键名**

**array\_reverse($arr,TRUE); 返回一个单元顺序与原数组相反的数组，如果第二个参数为TRUE保留原来的键名**

**array\_rand($arr,2); 从数组中随机取出一个或 多个元素**

**shuffle($arr); 将数组的顺序打乱**

**文件函数**

**basename(); 获得文件名**

**dirname(); 得到目录部分**

**pathinfo(); 得到路径关联数组**

**fopen() -- 打开文件或者 URL**

**fclose -- 关闭文件句柄**

**filesize();返回文件大小的字节数**

**opendir -- 打开目录句柄，打开一个目录句柄，可用于之后的 closedir()，readdir() 调用中。**

**Closedir -- 关闭目录句柄**

**readdir -- 从目录句柄中读取条目，返回目录中下一个文件的文件名。文件名以在文件系统中的排序返回。**

**file\_exists();检查文件是否存在**

**file\_get\_contents(); 将整个文件读入一个字符串**

**file\_put\_contents(); 将一个字符串写入文件**

**数学函数**

**函数语法：**

**string number\_format(float number, int [decimals], string [dec\_point], string [thousands\_sep])**

**函数说明：**

**本函数用来将浮点参数number按照指定格式输出。若没加参数decimals则传回的字串只取整数部分，加了此参数才依参数指定的小数点位数传回。参数dec\_point表示小数点的表示方法，内定值是“.”，若需要转换成其他的符号就可以将这个参数改掉。**

**函数例程：**

**<?php**

**$number = 2584729724.6912;**

**echo number\_format($number); // 2,584,729,725**

**echo number\_format($number, 3, ".", ","); // 2,584,729,724.691**

**echo number\_format($number, 2, ",", " "); /\*法式数字写法：2 584 729 724,69 小数点用“,”\*/**

**echo number\_format($number, 2, ".", ""); // 2584729724.69**

**?>**

**日期函数**

**array getdate([int $timestamp])**

**函数作用：**

**取得参数时间戳的日期时间相关数据，返回一个数组，每个元素存储一种时间信息，有年、月、日、时、分、秒、星期等，如果参数为空，则返回程序运行的当前时刻的日期时间相关数据。**

**int time(void)**

**函数作用：**

**返回自Unix纪元开始(格林尼治时间1970年1月1日00:00:00)到当前时间的秒数。**

**int strtotime(string $time [,int $now=time()])**

**函数作用：**

**将任何英文文本的日期时间描述解析为 Unix 时间戳**

**date -- 格式化日期**

**函数语法：**

**string date(string $format[, int $timestamp])**

**函数作用：**

**将指定的时刻$timestamp按照格式字符串$format的格式输出，参数$timestamp为空时，默认使用当前时刻time()。**

**参考手册阅读**

**函数例程：**

**<?php**

**echo date("Y-m-d"); //2008-08-07**

**echo date("n/j/Y"); //8/7/2008**

**echo date("Y年n月j日"); //2008年8月7日**

**echo date("Y年n月j日", mktime(0, 0, 0, 8, 10, 2008)); //2008年8月10日**

**?>**

**对象函数**

**get\_class() //获取对象的类名称**

**get\_class\_methods() //获取对象的方法名称**

**class\_exists() //判断类是否已定义**

**method\_exists() //判断对象的方法是否存在**

**声明对象**

**class Car**

**{**

**var $color;**

**function Car($color="green") {**

**$this->color = $color;**

**}**

**function what\_color() {**

**return $this->color;**

**}**

**}**

**$car = new Car("big car");**

**echo $car->color; // big car**