此文是根据php官方文档与php程序设计一书完成的。

## php简介

php(HyperText preprocessor,超文本预处理器)，是一种被广泛应用的开源代码的多用途脚本语言，可以嵌入到html中，尤其适合web开发。

与c写的脚本的区别—与用大量的命令来编写程序以输出html不同的是，php页面就是html.只不过在其中嵌入了一些代码来做一些事情

与客户端的javascript不同的是，php代码运行在服务端。

## php能做什么

### 服务端脚本

创建动态的web内容

### 命令行脚本

Php通过使用命令行脚本来执行系统管理任务，例如备份和日志解析

### 客户端GUI应用

Php可以通过PHP-GTK来编写客户端应用程序，php-GTK是php的一个扩展

## 基本语法

### Php标记

<? 和?>,解析php文件，php会寻找起始标记和结束标记，这种解析方式使得php可以嵌入到各种不同的文档中去，而任何起始和结束标记之外的部分都会被php解析器忽略

### 指令分割符

Php每个语句后用分号结束指令

### instanceof

用于确定一个 PHP 变量是否属于某一类 [class](https://www.php.net/manual/zh/language.oop5.basic.php#language.oop5.basic.class) 的实例：

返回bool类型

### method\_exists

检查类的方法是否存在

**method\_exists** ( [mixed](https://www.php.net/manual/zh/language.pseudo-types.php" \l "language.types.mixed) $object , string $method\_name ) : bool

### 注释

//

/\* \*/

## 词法结构

### 大小写

和内置结构的关键字(echo,class,while)一样,用户定义的类名和函数名也是不区分大小写的，

但是变量区分大小写

### 空白符与换行符

Php中空白符无关紧要，一条语句可以跨多行，也可以紧密的连在一起

### 标识符

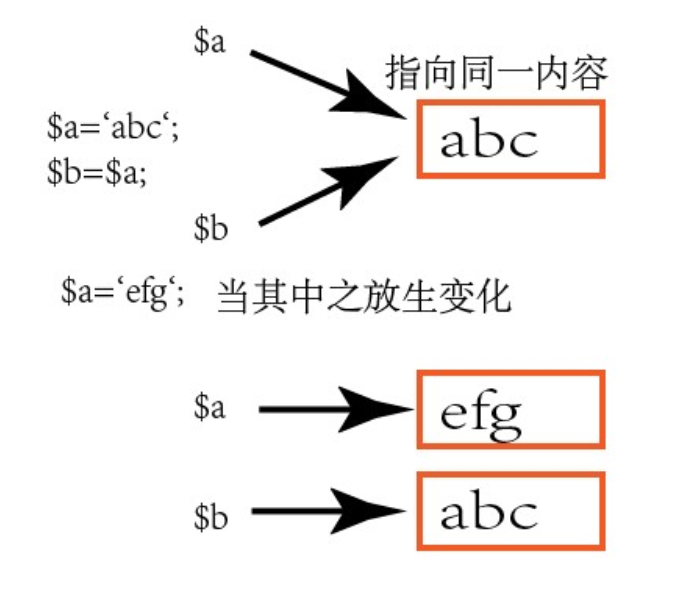
标识符只是一个名字，在php里标识符用来命名变量、函数、常量和类

关键字

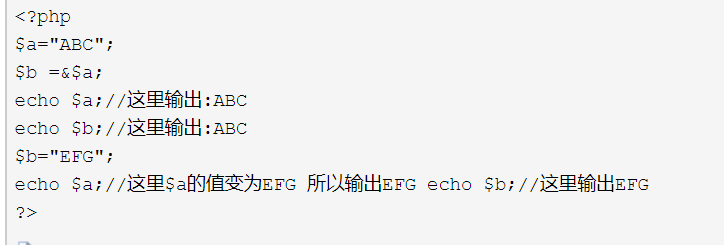
是语言为其核心功能而预留的单词，在对变量、函数、类或常量命名时不能使用和关键字相同的名字。

### PHP\_EOL等于\n 换行符

## 引用

‘’

### 变量引用



### 函数引用

<?php

function test(&$a){

$a=$a+100;

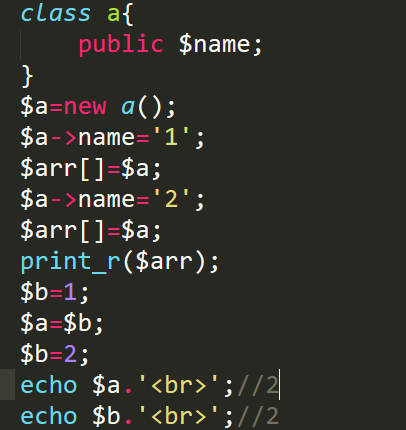
}

$b=1;

echo $b;//输出１ test($b); //这里$b传递给函数的其实是$b的变量内容所处的内存地址，通过在函数里改变$a的值　就可以改变$b的值了 echo "<br>"; echo $b;//输出101

?>

### 对象引用



## 八大数据类型

### 标量数据层类型

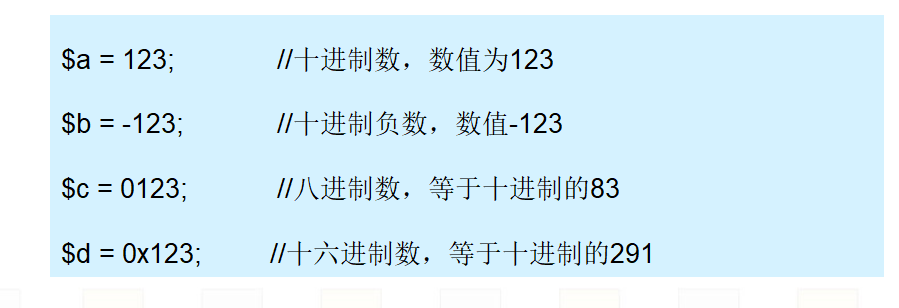
#### 布尔类型（boolean）

#### 整数型(integer)

整数型用来表示不包含小数部分的数，它可以用十进制、十六进制、八进制、二进制、指定。

当使用八进制，数字前加0

当使用十六进制，数字前加0x

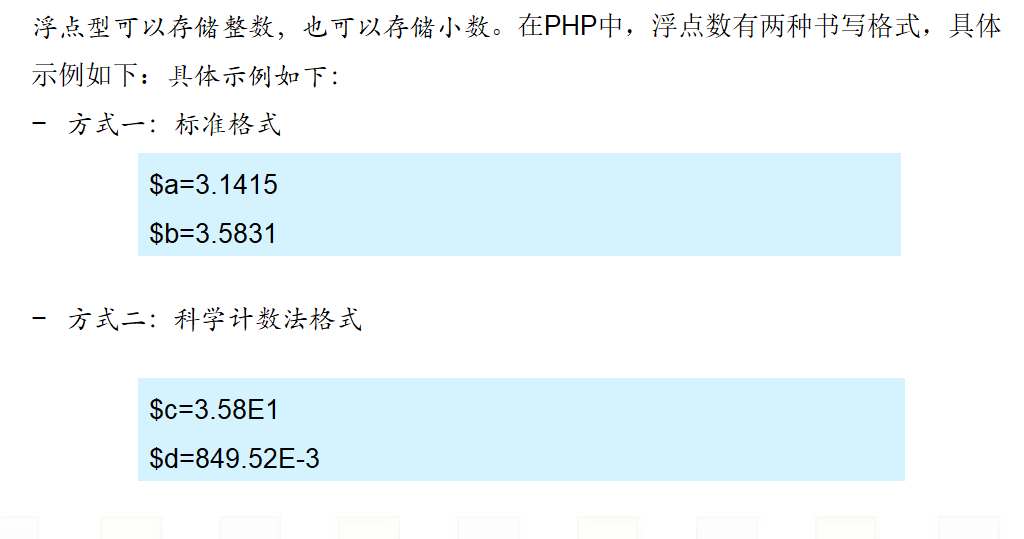


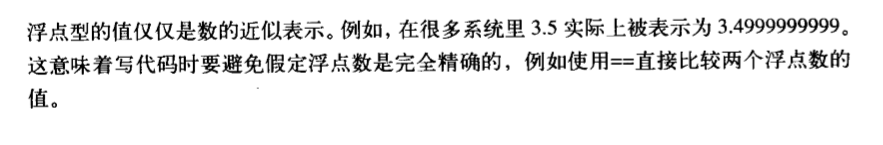
#### 浮点型(float)

在php中float与double都是一样的

浮点型既可以存储小数、也可以存整数

浮点型的值仅仅是数的近似表示





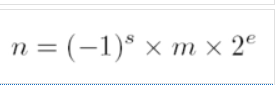
0.1+0.2==0.3 判断结果为false --- 移动小数点将浮点型变成整数比较

##### 浮点型的由来

在计算机系统发展的过程中，曾提出过很多方法表示实数，但是到目前为止使用最广泛的是[浮点表示法](https://baike.baidu.com/item/%E6%B5%AE%E7%82%B9%E8%A1%A8%E7%A4%BA%E6%B3%95/6267535)。

相对于定点数而言，浮点数利用指数使小数点的位置可以根据需要而上下浮动，从而可以灵活地表达更大范围的实数。

浮点数是一串0和1构成的位序列，从逻辑上用三元组{S,E,M}表示一个数N,如图《浮点数公式1》所示：

[](https://baike.baidu.com/pic/%E6%B5%AE%E7%82%B9%E5%9E%8B/10765495/0/0e655ca7225f11d6d1435895?fr=lemma&ct=single)浮点数公式1

S:符号位

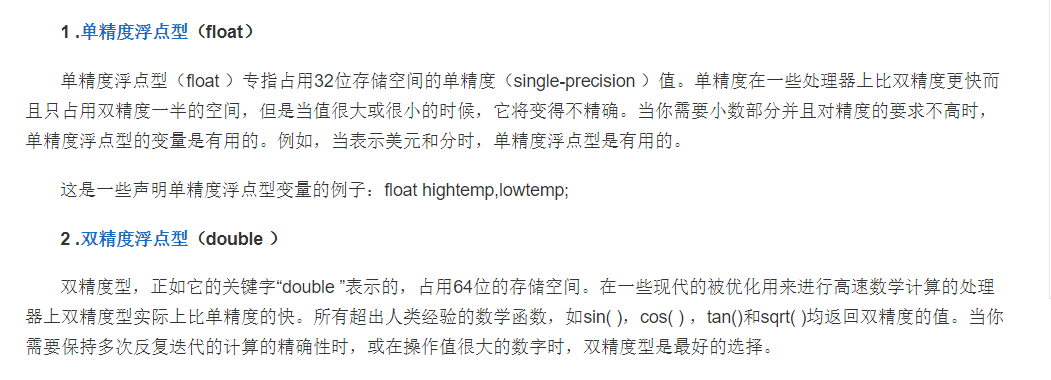
E：指数位

M:尾数位

现在采用的是IEEE 754格式，

Float通常为32位，double为64位

 long double型通常是比double型精度更高的类型，但各平台的实现有较大差异。在x86平台上，大多数[编译器](https://baike.baidu.com/item/%E7%BC%96%E8%AF%91%E5%99%A8)实现的long double型是80位，因为x86的[浮点运算单元](https://baike.baidu.com/item/%E6%B5%AE%E7%82%B9%E8%BF%90%E7%AE%97%E5%8D%95%E5%85%83)具有80位精度，gcc实现的long double型是12[字节](https://baike.baidu.com/item/%E5%AD%97%E8%8A%82)（96位），这是为了对齐到4字节边界，也有些编译器实现的long double型和double型精度相同，没有充分利用x86浮点运算单元的精度。其它体系结构的浮点运算单元的精度不同，编译器实现也会不同，例如PowerPC上的long double型通常是128位。



##### Php 浮点型

浮点数的精度有限。尽管取决于系统，PHP 通常使用 IEEE 754 双精度格式 即64位

所以永远不要相信浮点数结果精确到了最后一位，也永远不要比较两个浮点数是否相等

##### Php浮点型的比较和计算通过BC数学函数

* [bcadd](https://www.php.net/manual/en/function.bcadd.php) —添加两个任意精度数字
* [bccomp](https://www.php.net/manual/en/function.bccomp.php) —比较两个任意精度的数字
* [bcdiv](https://www.php.net/manual/en/function.bcdiv.php) —除以两个任意精度的数字
* [bcmod](https://www.php.net/manual/en/function.bcmod.php) —获取任意精度数的模
* [bcmul](https://www.php.net/manual/en/function.bcmul.php) —将两个任意精度数字相乘
* [bcpow](https://www.php.net/manual/en/function.bcpow.php) —将任意精度数提高到另一个
* [bcpowmod](https://www.php.net/manual/en/function.bcpowmod.php) —将任意精度数提高为另一个，并减少指定的模数
* [bcscale](https://www.php.net/manual/en/function.bcscale.php) —为所有bc数学函数设置或获取默认的scale参数
* [bcsqrt](https://www.php.net/manual/en/function.bcsqrt.php) —获取任意精度数的平方根
* [bcsub](https://www.php.net/manual/en/function.bcsub.php) —从另一个减去一个任意精度数

#### 字符串(String)

### 复合数据类型

数组

对象

### 特殊数据类型

资源（resource）

Null

## 变量

### Unset删除不存在变量不会报错

### 预定义变量

#### $\_SERVER

$\_SERVER[‘SERVER\_ADDR’]------服务器所在IP地址

$\_SERVER[‘SERVER\_PORT’]-----服务器所使用的端口

$\_SERVER['REMOTE\_ADDR']---- 正在浏览当前页面用户的 IP 地址。

$\_SERVER[‘HTTP\_USER\_AGENT’]------当前用户使用的浏览器信息

$\_SERVER[‘PHP\_SELF’]------当前正在执行的脚本文件名

$\_SERVER[‘SERVER\_SIGNTURE’]-----当前服务器版本和虚拟主机名的字符串

参考：<https://blog.csdn.net/moliyiran/article/details/79806108>

## 常量

### 魔术常量

#### \_\_DIR\_\_

获取当前文件所在的目录地址

#### \_\_FILE\_\_

文件完整路径和文件名

#### \_\_FUNCTION\_\_

返回当前函数名称

#### \_CLASS\_

返回当前类名称

#### \_\_METHOD\_\_

返回类的方法名 （区分大小写）

#### \_\_NAMESPACE\_\_

返回当前命名空间

## 表达式

## 进制

Php 数字前+0代表8进制 017+1=16

数字前加0x代表16进制

## 运算符

### ===、==与=

|  |  |  |
| --- | --- | --- |
| x == y | 相等 | 如果 x 和 y 具有相同的键/值对，则返回 true |
| x === y | 恒等 | 如果 x 和 y 具有相同的键/值对，且顺序相同类型相同，则返回 true |

### ?:和??

$b=$a??$c 等价于$b=isset($a)?$a:$c

$b=$a?$a:$c 等价于$b=!empty($a)?$a:$c 等价于$b=$a?:$c

**function** test($name){  
 **return** $name?:**'123'**;  
}  
*var\_dump*(test($name=1));

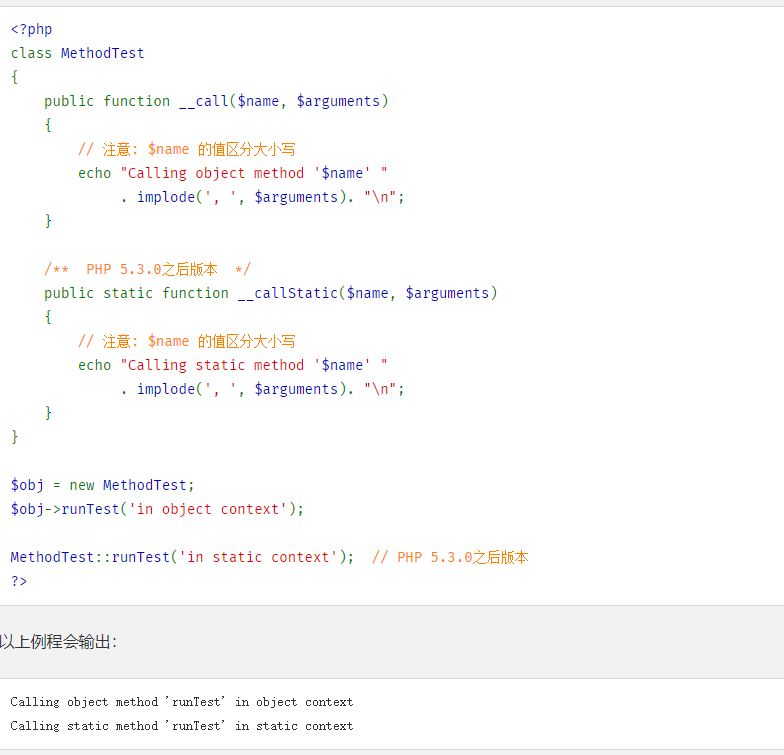
//输出1

## 魔术方法

### 方法重载

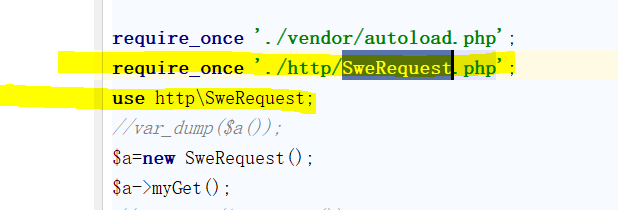
在对象中调用一个不可访问方法时，[\_\_call()](https://www.php.net/manual/zh/language.oop5.overloading.php#object.call) 会被调用。

在静态上下文中调用一个不可访问方法时，[\_\_callStatic()](https://www.php.net/manual/zh/language.oop5.overloading.php#object.callstatic) 会被调用。



## use、require、namespace区别

use只是进行声明、并不能实际载入某个类，require是引入某个文件 因此在使用use之前 要require该类



## 函数

### ftok（）该函数将现有可访问文件的文件名和项目标识符转换为整数

**ftok** ( string $filename , string $project\_id ) : int

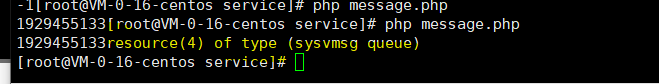
第二个参数长度只能为1

### msg\_get\_queue（）返回可用于访问具有给定密钥的SystemV消息队列的id

**msg\_get\_queue** ( int $key , int $permissions = 0666 ) : SysvMessageQueue|false

$key消息队列数字ID

$permission队列权限。默认为0666。如果消息队列已存在，则将忽略权限。



### msg\_send()向队列发送消息

msg\_send ( SysvMessageQueue $queue , int $message\_type , string|int|float|bool $message , bool $serialize = true , bool $blocking = true , int &$error\_code = null ) : bool

$queue 消息队列

$message\_type 消息的类型必须大于0

$message 消息内容

$serialize 是否序列化

$blocking

如果消息太大而无法放入队列，则脚本将等待另一个进程从队列中读取消息并释放足够的空间来发送消息。这称为阻塞。您可以通过将可选的阻塞参数设置为false来防止阻塞

$error\_code 错误码

### msg\_receive()接收队列消息

msg\_receive ( SysvMessageQueue $queue , int $desired\_message\_type , int &$received\_message\_type , int $max\_message\_size , [mixed](https://www.php.net/manual/en/language.types.declarations.php#language.types.declarations.mixed) &$message , bool $unserialize = true , int $flags = 0 , int &$error\_code = null ) : bool

$queue 消息队列

$**desired\_message\_type**

如果所需的消息类型为0，则返回队列前面的消息。如果所需的消息类型大于0，则返回该类型的第一条消息。如果所需消息类型小于0，则将读取队列中类型小于或等于所需消息类型绝对值的第一条消息

$received\_message\_typ 接收到的消息类型将存储在此参数中

$max\_message\_size 要接受的最大消息大小由最大消息大小指定；如果队列中的消息大于此大小，则函数将失败

$message 接收到的消息将存储在消息

$unserialize 序列化

$flags

可选标志允许您将标志传递给低级msgrcv系统调用。它的默认值为0，但您可以指定以下一个或多个值（通过将它们相加或进行O型运算）

### 获取请求的url

$scheme = $\_SERVER['REQUEST\_SCHEME']; *//协议*

$domain = $\_SERVER['HTTP\_HOST']; *//域名/主机*

$requestUri = $\_SERVER['REQUEST\_URI']; *//请求参数*

*//将得到的各项拼接起来*

$currentUrl = $scheme . "://" . $domain . $requestUri

### preg\_match

用于执行一个正则表达式的匹配

preg\_match ( string $pattern , string $subject [, array &$matches [, int $flags = 0 [, int $offset = 0 ]]] )

$pattern是正则表达式

$subject为输入的字符串

$matches: 如果提供了参数matches，它将被填充为搜索结果。 $matches[0]将包含完整模式匹配到的文本， $matches[1] 将包含第一个捕获子组匹配到的文本，以此类推

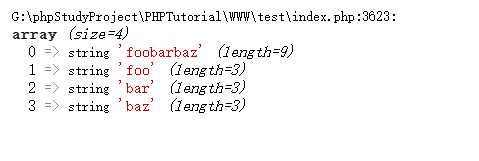
$flags

flags 可以被设置为以下标记值, PREG\_OFFSET\_CAPTURE: 如果传递了这个标记，对于每一个出现的匹配返回时会附加字符串偏移量(相对于目标字符串的)。 注意：这会改变填充到matches参数的数组，使其每个元素成为一个由 第0个元素是匹配到的字符串，第1个元素是该匹配字符串 在目标字符串subject中的偏移量

$offset:

#### $matches参数含义

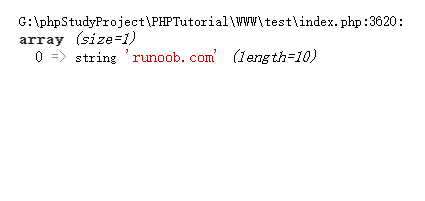
*//如果提供了参数matches，它将被填充为搜索结果。 $matches[0]将包含完整模式匹配到的文本，  
// $matches[1] 将包含第一个捕获子组匹配到的文本，以此类推  
preg\_match*(**'/(foo)(bar)(baz)/'**, **'foobarbaz'**, $matches);  
*var\_dump*($matches);



捕获子组:关于正则的分组见正则表达式word:使用（）对正则表达式进行分组

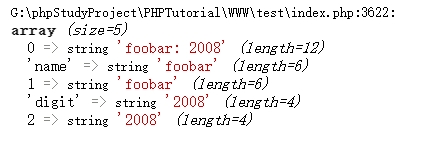
preg\_match(**'/[^.]+\.[^.]+$/'**, **'www.runoob.com'**, $matches);  
*var\_dump*($matches);

正则表达式没有分组，$matches数组只有一个值



*//使用命名分组*$str = **'foobar: 2008'**;  
*preg\_match*(**'/(?P<name>\w+): (?P<digit>\d+)/'**, $str, $matches);  
*var\_dump*($matches);

使用命名分组会将分组的命名和值作为键值对填充到$matches

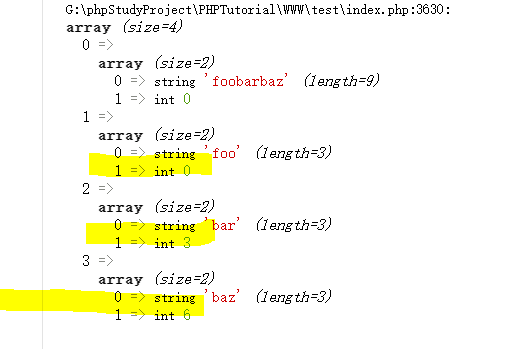


#### Flags参数含义

**Flags被设PREG\_OFFSET\_CAPTURE时**

对于每一个出现的匹配返回时会附加字符串偏移量(相对于目标字符串的字节数)。 注意：这会改变填充到matches参数的数组，使其每个元素成为一个由 第0个元素是匹配到的字符串，第1个元素是该匹配字符串 在目标字符串subject中的偏移量。

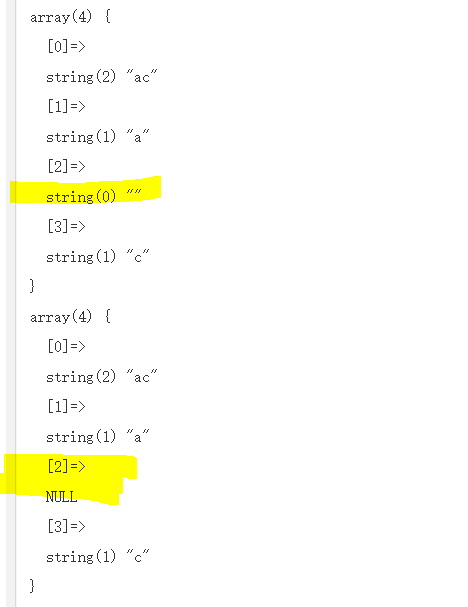
preg\_match('/(foo)(bar)(baz)/', 'foobarbaz', $matches, PREG\_OFFSET\_CAPTURE);  
var\_dump($matches);



**Flags为PREG\_UNMATCHED\_AS\_NULL时**

使用该标记，未匹配的子组会报告为 NULL；未使用时，报告为空的 string

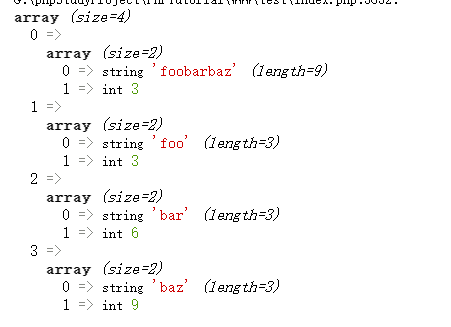
preg\_match('/(a)(b)\*(c)/', 'ac', $matches);  
var\_dump($matches);  
preg\_match('/(a)(b)\*(c)/', 'ac', $matches, PREG\_UNMATCHED\_AS\_NULL);  
var\_dump($matches);



#### Offset参数含义

通常，搜索从目标字符串的开始位置开始。可选参数 offset 用于 指定从目标字符串的某个位置开始搜索(单位是字节)

preg\_match(**'/(foo)(bar)(baz)/'**, **'333foobarbaz'**, $matches, ***PREG\_OFFSET\_CAPTURE***,3);  
*var\_dump*($matches);

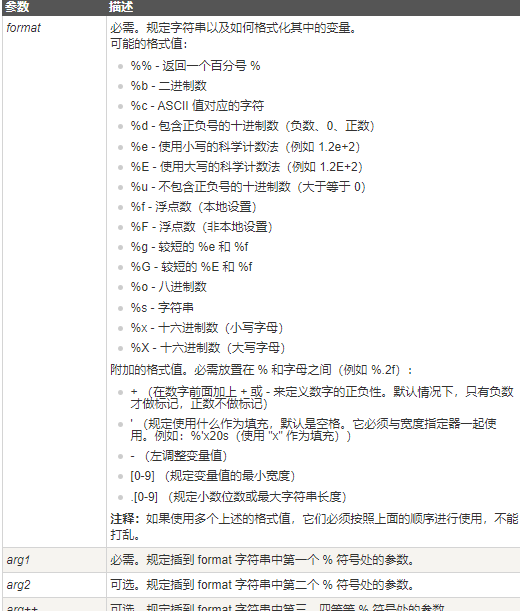


### Sprintf()

把格式化的字符串写入变量

把百分号（%）符号替换成一个作为参数进行传递的变量：

sprintf(format,arg1,arg2,arg++)



### crc32()

负数

### Current()

$people = array("Peter", "Joe", "Glenn", "Cleveland");

echo current($people) . "<br>";

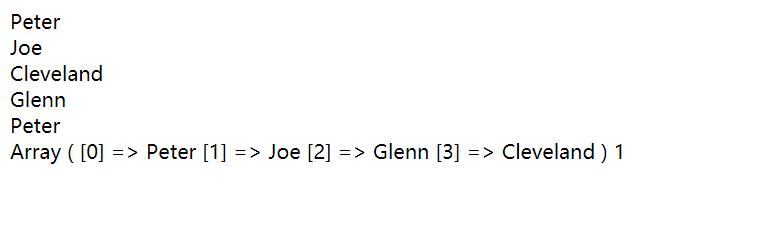
echo next($people) . "<br>";

echo end($people) . "<br>";

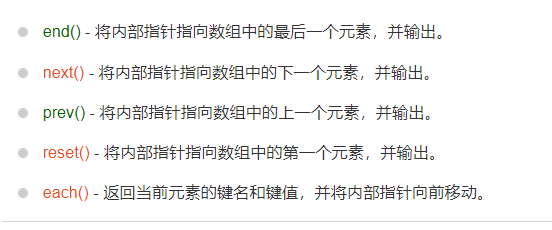
echo prev($people) . "<br>";

echo reset($people) . "<br>";

print\_r ($people) . "<br>";



current() 函数返回数组中的当前元素的值。



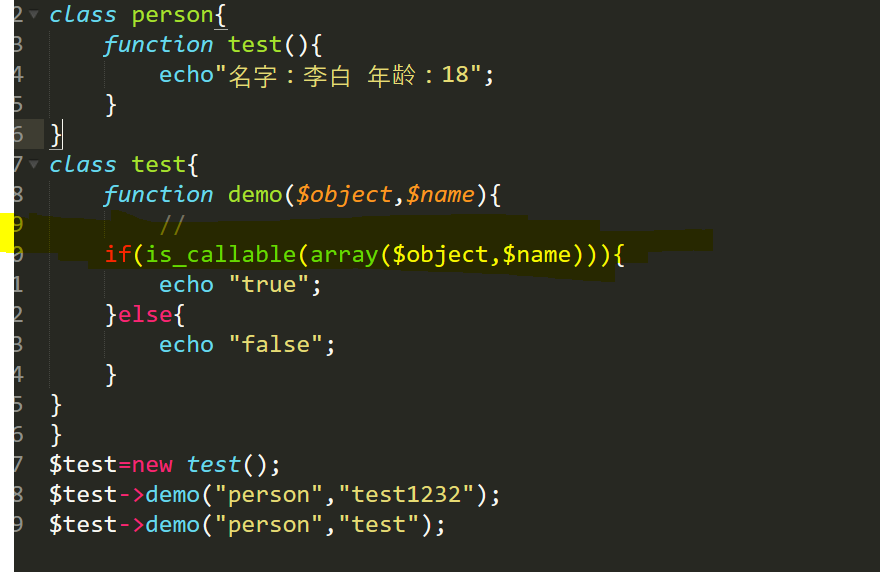
### **is\_callable()**

用于检测函数在当前环境中是否可调用

函数验证变量的内容能否作为函数调用。 这可以检查包含有效函数名的变量，或者一个数组，包含了正确编码的对象以及函数名

bool is\_callable ( callable $name [, bool $syntax\_only = false [, string &$callable\_name ]] )

检测参数是否为合法的可调用结构



### Create\_function

创建匿名函数

Php7.2已弃用

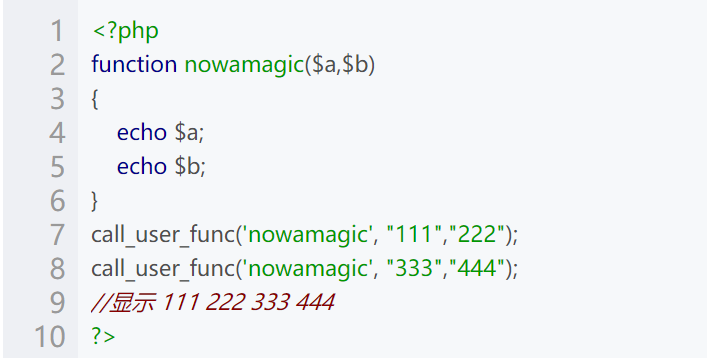
### Call\_user\_func

**call\_user\_func** ( [callable](https://www.php.net/manual/zh/language.types.callable.php) $callback [, [mixed](https://www.php.net/manual/zh/language.pseudo-types.php#language.types.mixed) $parameter [, [mixed](https://www.php.net/manual/zh/language.pseudo-types.php#language.types.mixed) $... ]] ) : [mixed](https://www.php.net/manual/zh/language.pseudo-types.php#language.types.mixed)

把第一个参数作为回调函数调用

call\_user\_func\_array是以数组的形式接收回调函数的参数的，





### call\_user\_func\_array、call\_user\_func回调函数

参考：<https://www.cnblogs.com/splendid/p/10658190.html>

// 1、 调用自定义函数

function test($a, $b)

{

echo $a + $b;

}

// 字符串传参

call\_user\_func('test', 1, 2); // 3

// 数组式传参

call\_user\_func\_array('test', [1, 2]); // 3

#### // 2、 调用匿名函数

call\_user\_func(function($a, $b){ echo $a + $b ;}, 1, 2); // 3

call\_user\_func\_array(function($a, $b){ echo $a + $b ;}, [1, 2]); // 3

// 3、 调用系统函数

echo call\_user\_func('strtoupper', 'abc'); // ABC

echo call\_user\_func\_array('strtoupper', ['abc']); // ABC

// 4、 调用类中的函数

class Test

{

static public function demo($a, $b)

{

echo $a + $b;

}

public function show($a, $b)

{

echo $a + $b;

}

}

#### // 调用类中的静态方法

// 类名方法名以数据形式

call\_user\_func(['Test', 'demo'], 1, 2); // 3

call\_user\_func\_array(['Test', 'demo'], [1, 2]); // 3

// 类名方法名以字符串形式

call\_user\_func('Test::demo', 1, 2); // 3

call\_user\_func\_array('Test::demo', [1, 2]); // 3

#### // 调用类中的动态方法，对象和方法必须通过数组形式传递

call\_user\_func([new Test, 'show'], 1, 2); // 3

call\_user\_func\_array([new Test, 'show'], [1, 2]); // 3

### spl\_autoload\_register自动加载

参考：https://www.php.cn/php-weizijiaocheng-400053.html

在了解这个函数之前先来看另一个函数：\_\_autoload

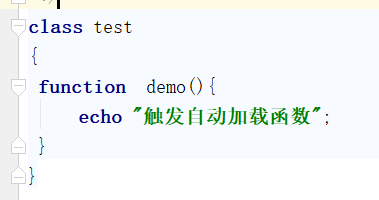
#### \_\_autoload

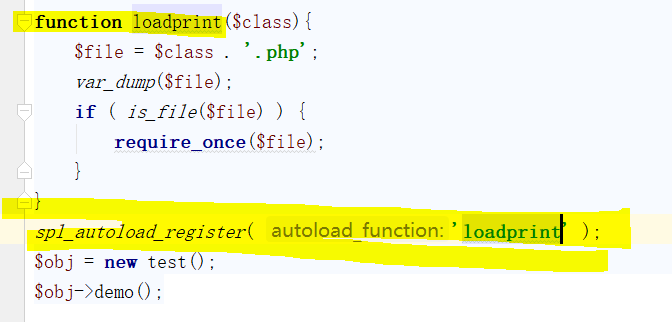
自动加载函数，在php5中，当实例化一个未定义的类时，会触发此函数

\_\_autoload在7.2已废弃



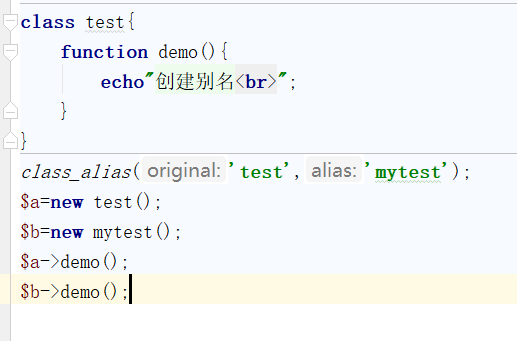
#### spl\_autoload\_register自动加载

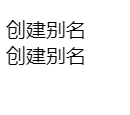




### class\_alias — 为一个类创建别名

class\_alias ( string $original , string $alias [, bool $autoload = TRUE ] ) : bool





### Closure匿名函数、闭包

匿名函数：没有名字的函数，最经常用作回调函数的参数值

#### 1.匿名函数赋值

$a=function($name){

echo '1231'.$name;

};

//调用闭包函数

$a('123');

##### 在函数中定义闭包函数并调用

//在函数中定义闭包函数并调用

function test(){

$fun=function($str){

echo $str;

};

$fun('aaaa');

}

test();//输出aaaa

##### 在函数中把匿名函数返回，并调用它

// 在函数中把匿名函数返回，并调用它

function test(){

return function($str){

echo $str;

};

}

$fun=test();

$fun('aaaa');

##### 匿名函数当做参数传递，并且调用它

// 把匿名函数当做参数传递，并且调用它

function test($fun){

$fun('aaaa');

}

$printStr=function($str){

echo $str;

};

test($printStr);//输出aaaa

#### 2.use关键字用于连接闭包和外界变量

use关键字用于连接闭包和外界变量

function getMoney() {

$rmb = 1;

$dollar = 8;

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

echo $rmb;

echo $dollar;

};

$func();

}

getMoney();

//输出：1

dollar没有在use关键字中声明，在这个匿名函数里也就不能获取到它

**use引用的变量是一个副本，在匿名函数中修改引用的变量，原来的外界变量是不变的**

function getMoney() {

$rmb = 1;

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

echo $rmb.'<br>';

//把$rmb的值加1

$rmb++;

};

$func();

echo $rmb;

}

getMoney();

//输出：

//1

//1

可以用完全引用来改变外界变量

function getMoney() {

$rmb = 1;

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

echo $rmb.'<br>';

//把$rmb的值加1

$rmb++;

};

$func();

echo $rmb;

}

getMoney();

//输出：

//1

//2

#### 3.闭包绑定

通过bindTo将闭包内部状态绑定到对象上，

class A {

private static $sfoo = 1;

private $ifoo = 2;

}

$cl1 = static function() {

return A::$sfoo;

};

$cl2 = function() {

//匿名函数中莫名奇妙的有个this,这个this关键词

//就是说明这个匿名函数是需要绑定在类中的。

return $this->ifoo;

};

$bcl1 = Closure::bind($cl1, null, 'A');

$bcl2 = Closure::bind($cl2, new A(), 'A');

echo $bcl1(), "\n";//输出 1

echo $bcl2(), "\n";//输出 2

<https://www.cnblogs.com/phpper/p/6993093.html>

字符串常用的处理函数

https://www.cnblogs.com/yuuje/p/10993128.html

### 长度

[strlen](https://www.php.net/manual/zh/function.strlen.php)($string):得到字符串长度

### 字符串查找

[strpos](https://www.php.net/manual/zh/function.strpos.php)($string, $search[, $offset]):在指定字符串中查找目标字符串第一次出现的位置

[stripos](https://www.php.net/manual/zh/function.stripos.php)($string, $search[, $offset]):忽略大小写的去查找

[strrpos](https://www.php.net/manual/zh/function.strrpos.php)($string, $search[, $offset]):在指定字符串中查找目标字符串最后一次出现的位置

[strripos](https://www.php.net/manual/zh/function.strripos.php)($string, $search[, $offset]):忽略大小写的去查找最后一次出现的位置

[strstr](https://www.php.net/manual/zh/function.strstr.php)|[strchr](https://www.php.net/manual/zh/function.strchr.php)($string, $search[, $before\_needle]):查找字符串的首次出现，返回的是字符串

[stristr](https://www.php.net/manual/zh/function.stristr.php)($string, $search[, $before\_needle]):忽略大小写的去查找

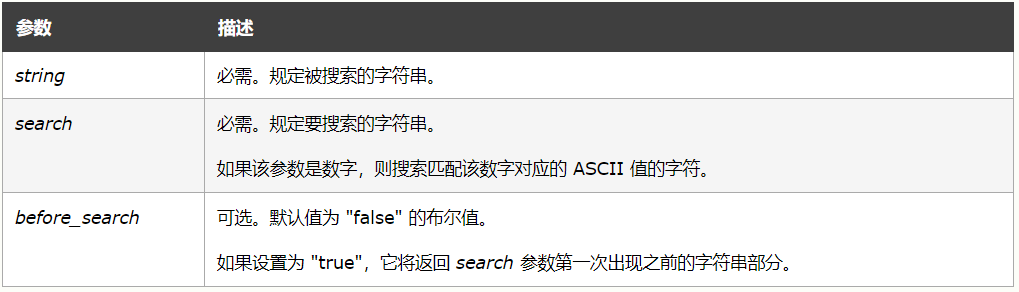
[strrchr](https://www.php.net/manual/zh/function.strrchr.php)($string, $search):查找指定字符在字符串中的最后一次出现，并返回从该位置到字符串结尾的所有字符

[str\_replace](https://www.php.net/manual/zh/function.str-replace.php)($search, $replace, $string):在指定字符串中查找另外一个字符串，找到之后将其替换成指定字符串

[str\_ireplace](https://www.php.net/manual/zh/function.str-ireplace.php)($search, $replace, $string):忽略大小写的去查找并替换

#### stristr搜索字符串在另一字符串中的第一次出现

stristr(*string*,*search*,*before\_search*)



|  |  |
| --- | --- |
| 返回值： | 返回字符串的剩余部分（从匹配点）。如果未找到所搜索的字符串，则返回 FALSE。 |

#### [str\_replace](https://www.php.net/manual/zh/function.str-replace.php)以其他字符替换字符串中的一些字符（区分大小写



Find可以是数组

$namespace=*str\_replace*(['"',';',' '],'',$namespace[1]);

### 字符串大小写

[strtolower](https://www.php.net/manual/zh/function.strtolower.php)($string):返回小写之后的字符串

[strtoupper](https://www.php.net/manual/zh/function.strtoupper.php)($string):返回大写之后的字符串

[ucwords](https://www.php.net/manual/zh/function.ucwords.php)($string):单词的首字母大写

[ucfirst](https://www.php.net/manual/zh/function.ucfirst.php)($string):字符串的单词的首字母大写

[lcfirst](https://www.php.net/manual/zh/function.lcfirst.php)($string):字符串的单词的首字母小写

### 字符串截取

[substr](https://www.php.net/manual/zh/function.substr.php)($string, $start[, $length]):截取字符串

[substr\_replace](https://www.php.net/manual/zh/function.substr-replace.php)($string, $replace, $start[, $length]):替换字符串的子串

### 字符ASCII

[ord](https://www.php.net/manual/zh/function.ord.php)($char):得到指定字符的ASCII

[chr](https://www.php.net/manual/zh/function.chr.php)($ascii):根据ASCII得到指定的字符

### 加密

[md5](https://www.php.net/manual/zh/function.md5.php)($string):计算字符串的md5的散列值，返回32位长度的字符串

[sha1](https://www.php.net/manual/zh/function.sha1.php)($string):计算字符串的sha1的散列值，返回40位长度的字符串

### 过滤

[trim](https://www.php.net/manual/zh/function.trim.php)($string[, $charlist]):默认过滤字符串两端的空格，也可以过滤指定字符串

[ltrim](https://www.php.net/manual/zh/function.ltrim.php)($string[, $charlist]):过滤字符串左端

[rtrim](https://www.php.net/manual/zh/function.rtrim.php)|[chop](https://www.php.net/manual/zh/function.chop.php)($string[, $charlist]):过滤字符串右端

[strip\_tags](https://www.php.net/manual/zh/function.strip-tags.php)($string[, $allowTag]):过滤字符串中的HTML标记

[addslashes](https://www.php.net/manual/zh/function.addslashes.php)($string):使用反斜线引用字符串中的特殊字符

[htmlentities](https://www.php.net/manual/zh/function.htmlentities.php)($string[, $flag=ENT\_COMPAT]):将所有字符转换成HTML实体

[htmlspecialchars](https://www.php.net/manual/zh/function.htmlspecialchars.php)($string[, $flag=ENT\_COMPAT]):将字符串中的特殊字符转换成HTML实体

[nl2br](https://www.php.net/manual/zh/function.nl2br.php)($string):将字符串中的\n用<br/>替换

### 比较

[strcmp](https://www.php.net/manual/zh/function.strcmp.php)($string1, $string2):比较两个字符串的大小

[strcasecmp](https://www.php.net/manual/zh/function.strcasecmp.php)($string1, $string2):忽略大小写的比较字符串的大小

[strnatcasecmp](https://www.php.net/manual/zh/function.strnatcasecmp.php)($string1, $string2):使用自然顺序算法比较字符串

[strncasecmp](https://www.php.net/manual/zh/function.strncasecmp.php)($string1, $string2, $len):比较字符串的开始的若干个字符

### 拆分、合并

[explode](https://www.php.net/manual/zh/function.explode.php)($delimiter, $string):将指定字符串拆分成数组

[implode](https://www.php.net/manual/zh/function.implode.php)|[join](https://www.php.net/manual/zh/function.join.php)($delimiter, $array):以指定分隔符将数组中的键值连接成字符串

[str\_split](https://www.php.net/manual/zh/function.str-split.php)($string[, $split\_length=1]):将字符串转换成数组

### 格式化字符串

[printf](https://www.php.net/manual/zh/function.printf.php)($format[, $args]):格式化输出字符串

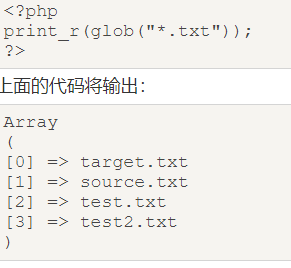
[sprintf](https://www.php.net/manual/zh/function.sprintf.php)($format[, $args]):格式化输出字符串

### define()定义常量

### glob()返回一个包含匹配指定模式的文件名或目录的数组

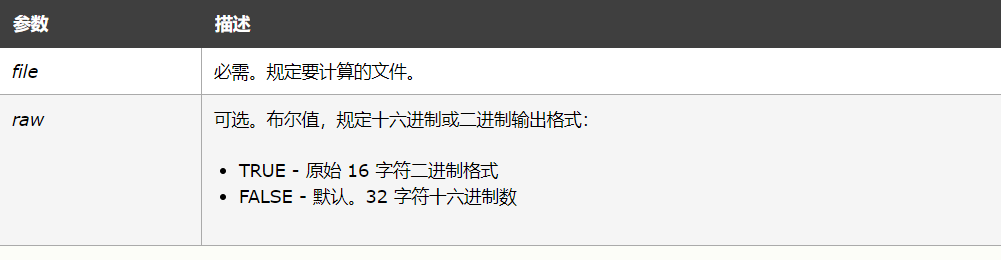
glob(pattern,flags)





### Md5\_file()计算文件的MD5散列

md5\_file(*file*,*raw*)



文件内容改变散列值会发生变化

### 高精度函数

https://blog.tuine.me/30.html

### 其他常用

[strrev](https://www.php.net/manual/zh/function.strrev.php)($string):反转字符串

[str\_shuffle](https://www.php.net/manual/zh/function.str-shuffle.php)($string):随机打乱字符串

[str\_repeat](https://www.php.net/manual/zh/function.str-repeat.php)($string):重复字符串

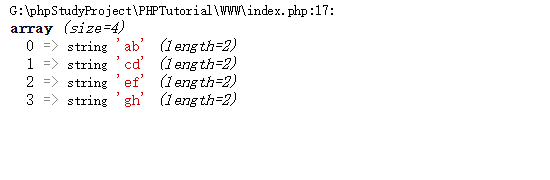
[str\_getcsv](https://www.php.net/manual/zh/function.str-getcsv.php)($input[, $delimiter=','[, $enclosure=''[, $escape='\']]]):解析CSV字符串为一个数组

[parse\_str](https://www.php.net/manual/zh/function.parse-str.php)($str[, $arr]):将字符串解析成多个变量

### str\_split()字符串按位分割

$str = 'abcdefgh';

$arr = str\_split($str,2);



str\_split() 函数把字符串分割到数组中。

str\_split(*string*,*length*) length不填默认为1

## 字符串处理函数

获取字符串长度 函数strlen()

字符串比较函数 strcmp() 和 strcasecmp()

查找和替换函数 substr()

strrpos() 函数查找在字符串中最后一次出现的位置

字符串裁剪函数 **trim()**、**rtrim()**和**ltrim()**

### 2.1并置运算符

并置运算符 (.) 用于把两个字符串值连接起来。

$txt1="Hello world!";   
$txt2="What a nice day!";   
echo $txt1 . " " . $txt2;

### 2.2strlen() 函数字符串长度

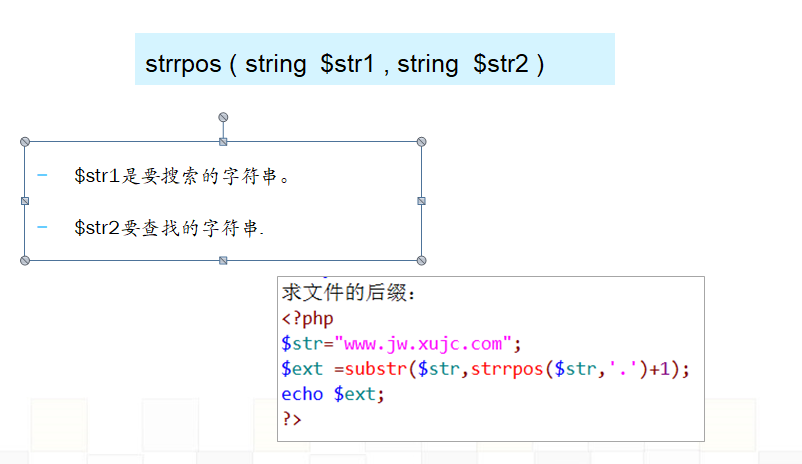
返回字符串的长度

strlen(“hello world!”);返回12

### 2.3Strpos()函数

文件后缀名一般1~4个，所以如果要是想获取文件扩展名，还可用**strrpos() 函数**

查找字符串在另一字符串中最后一次出现的位置



查找一个字符或一个文本，返回第一个匹配的字符位置

strops("Hello world!","world");返回6

strpos(string,find,start) 区分大小写的

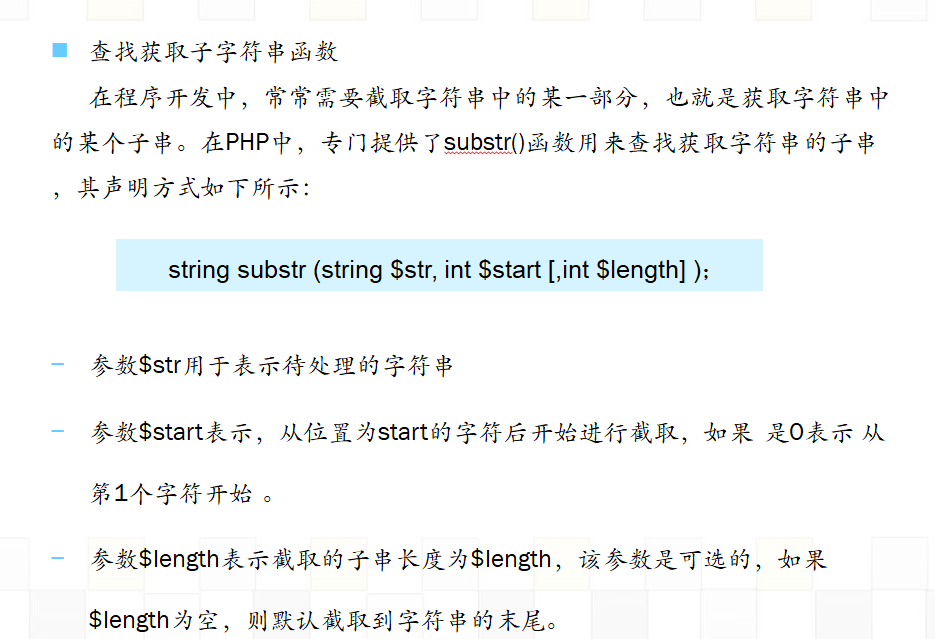
[strrpos()](https://www.runoob.com/php/func-string-strrpos.html) - 查找字符串在另一字符串中最后一次出现的位置（区分大小写）

[stripos()](https://www.runoob.com/php/func-string-stripos.html)- 查找字符串在另一字符串中第一次出现的位置（不区分大小写）

[strripos()](https://www.runoob.com/php/func-string-strripos.html) -查找字符串在另一字符串中最后一次出现的位置（不区分大小写）

|  |  |
| --- | --- |
| *string* | 必需。规定被搜索的字符串。 |
| *find* | 必需。规定要查找的字符。 |
| *start* | 可选。规定开始搜索的位置。 |

### 2.4substr()函数字符串截取





substr(str,start,length)

start>0,length没有，从左往右，从指定的start位置到字符串结尾

echo substr("Hello world",10)."<br>";===》d

start <0,length没有,从左往右，从指定的start位置到字符串结尾

echo substr("Hello world",-1)."<br>";===》d

指定length

Start>0 ; 返回start-length 不包括length

echo substr("Hello world",0,10)."<br>"; Hello worl

echo substr("Hello world",0,-1)."<br>" Hello worl

Start<0;返回start-length 不包括length

echo substr("Hello world",-10,-2)."<br>"; ello wor

echo substr("Hello world",-10,4); ell

### 2.5array\_count\_values() 函数

用于统计数组中所有值出现的次数

### 2.6 strcmp 进行字符串二进制比较

0 - 如果两个字符串相等

<0 - 如果 string1 小于 string2

>0 - 如果 string1 大于 string2

### 2.7define()定义一个常量

在设定以后，常量的值无法更改

常量名不需要开头的美元符号 ($)

作用域不影响对常量的访问

常量值只能是字符串或数字

### 2.8 {$a}和$a区别

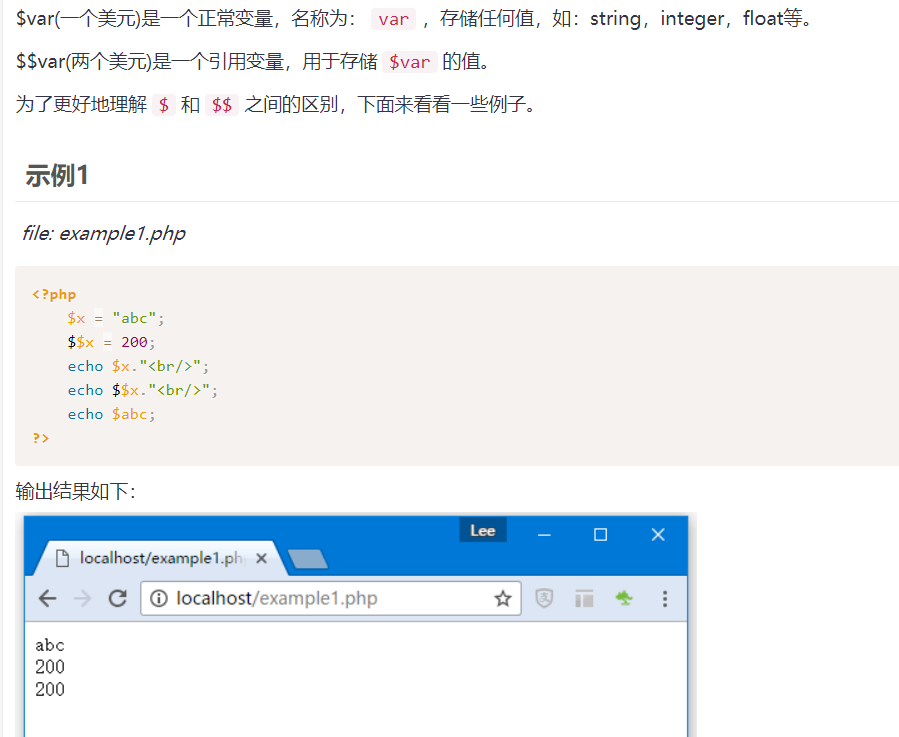
$Try[2]['key']="Jim";

echo "hello,$Try[2]['key']"; //hello,Array['key']

echo "</br>";

echo "hello,{$Try[2]['key']}";//输出的是hello，jim

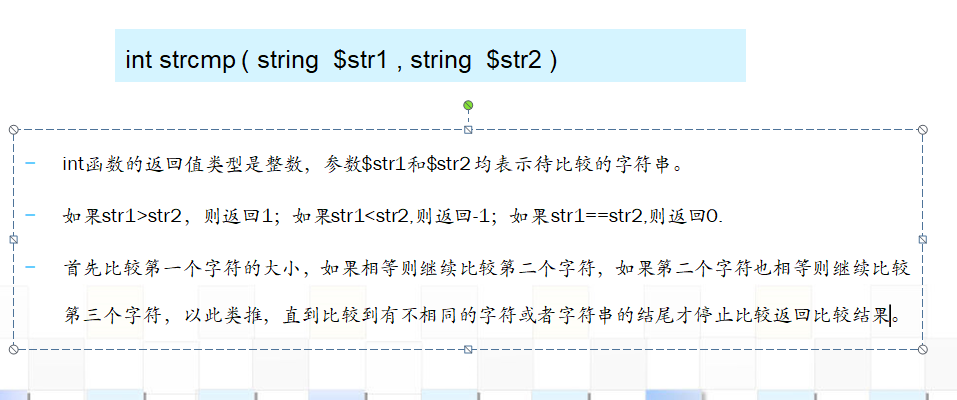
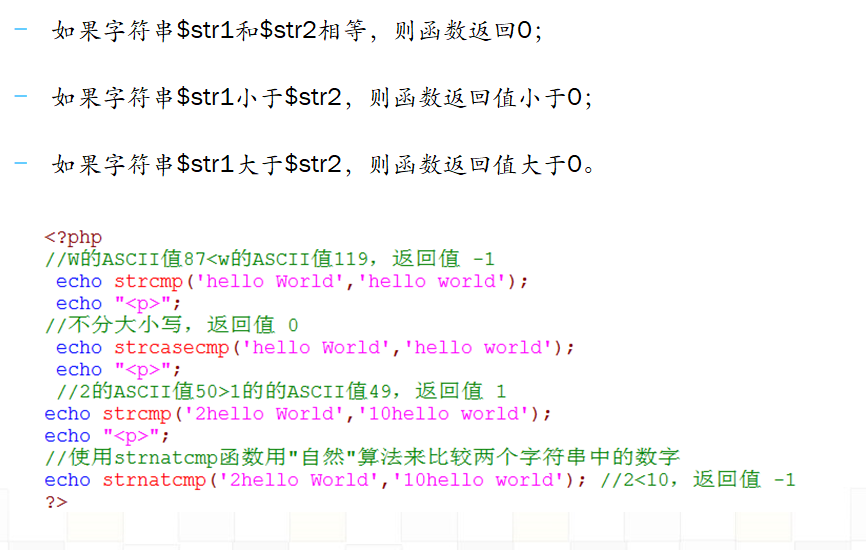
### 2.9 $和$$之间的区别



### 2.1.1字符串比较函数 strcmp() 和 strcasecmp()

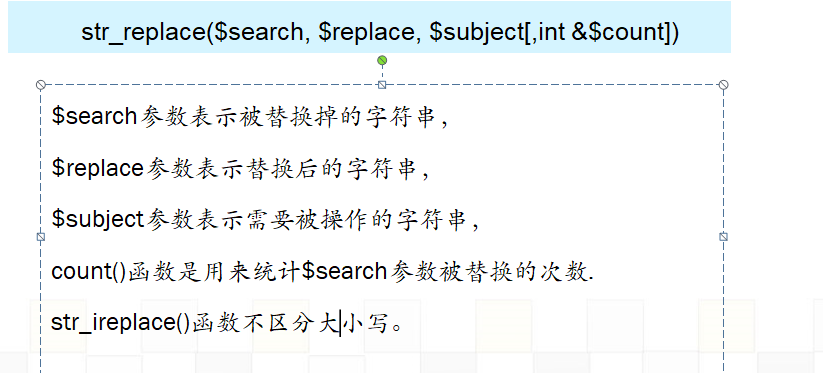
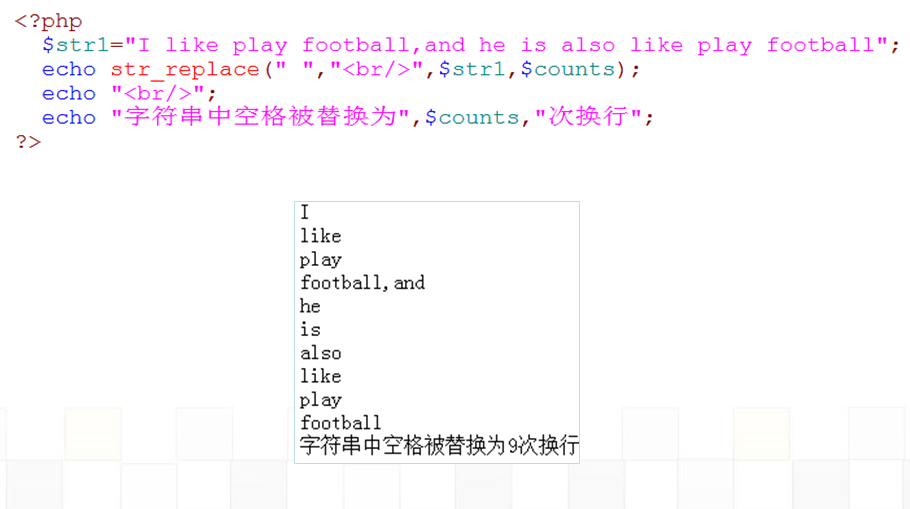
两者的区别在于strcasecmp()不区分比较字符串的大小写，

按照字节的ASCII值进行比较

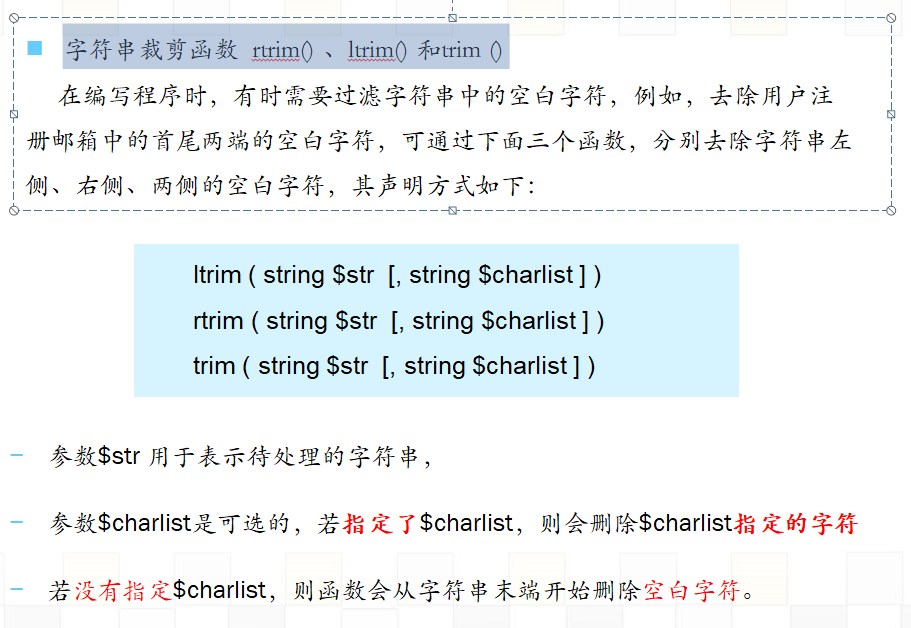
 

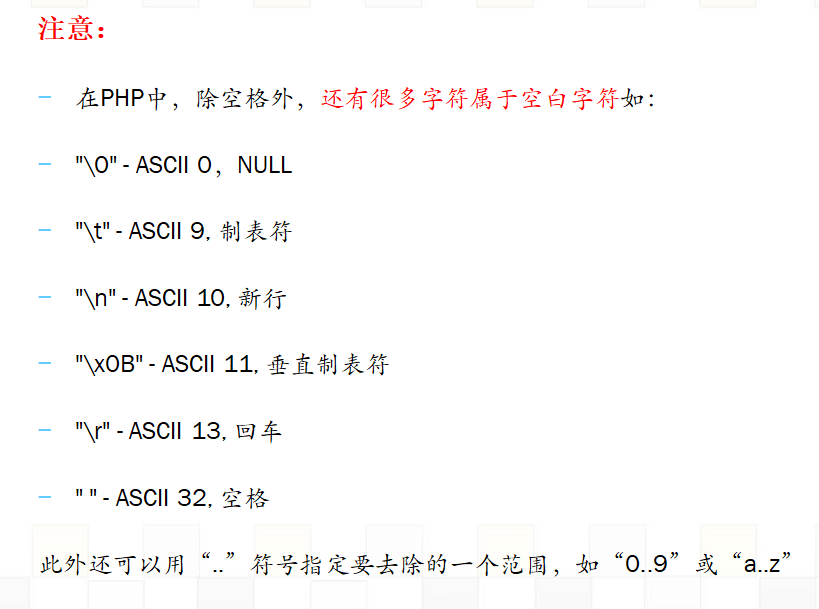
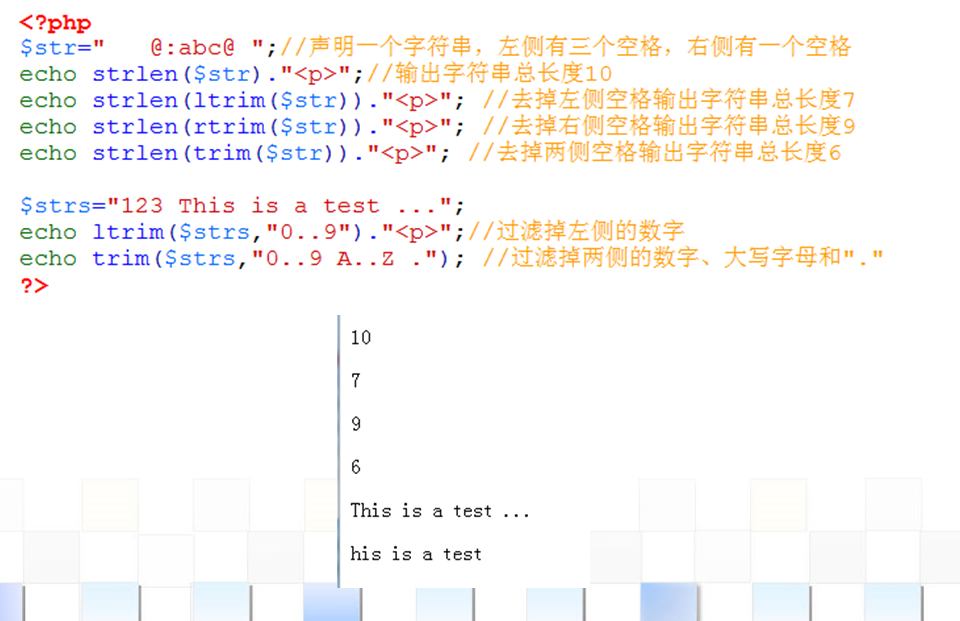
### 2.1.2字符串替换函数str\_replace()和str\_ireplace()

在程序开发中，经常会对字符串中的某些字符进行替换操作，这时可以通过str\_replace()函数和str\_ireplace()函数来完成。

### 2.1.3字符串裁剪函数 rtrim() 、ltrim() 和trim ()



### basename() 函数返回路径中的文件名部分。

### 字符串分割

Explode(‘,’,$arr)将字符串以逗号分割成数组

Implode(‘,’,$arr);将数组按逗号拼接成字符串

explode(separator,string,limit)

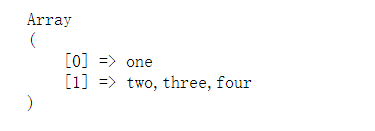
|  |  |
| --- | --- |
| **参数** | **描述** |
| *separator* | 必需。规定在哪里分割字符串。 |
| *string* | 必需。要分割的字符串。 |
| *limit* | 可选。规定所返回的数组元素的数目。  可能的值：   * 大于 0 - 返回包含最多 *limit* 个元素的数组 * 小于 0 - 返回包含除了最后的 *-limit* 个元素以外的所有元素的数组 * 0 - 会被当做 1, 返回包含一个元素的数组 |

$str = 'one,two,three,four';

// 数组元素为 2

print\_r(explode(',',$str,2));

print "<br>";



### mb\_strpos查找字符串在另一个字符串中首次出现的位置

查找字符串在另一个字符串中首次出现的位置

mb\_strpos ( string $haystack , string $needle [, int $offset =0 [, string $encoding = mb\_internal\_encoding() ]] ) : int

haystack

要被检查的 string。

needle

在 haystack 中查找这个字符串。 和 [strpos()](https://www.php.net/manual/zh/function.strpos.php) 不同的是，数字的值不会被当做字符的顺序值。

offset

搜索位置的偏移。如果没有提供该参数，将会使用 0。负数的 offset 会从字符串尾部开始统计。

encoding

encoding 参数为字符编码。如果省略，则使用内部字符编码。

返回 string 的 haystack 中 needle 首次出现位置的数值。 如果没有找到 needle，它将返回 FALSE。

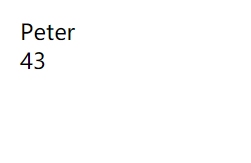
### parse\_str将字符解析到变量中

parse\_str(string,array)

|  |  |
| --- | --- |
| *string* | 必需。规定要解析的字符串。 |
| *array* | 可选。规定存储变量的数组名称。该参数指示变量存储到数组中。 |

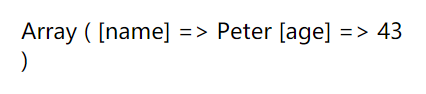
Array不设置的情况下，会解析成多个变量

parse\_str("name=Peter&age=43");  
echo $name."<br>";  
echo $age;

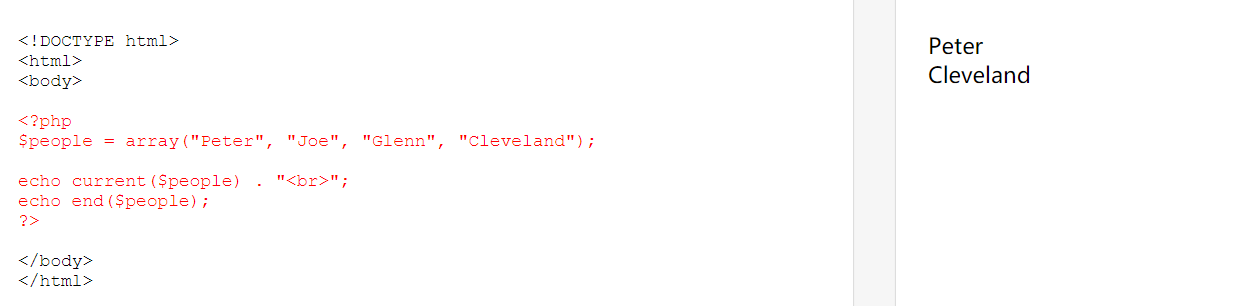


Array设置的情况下，会解析成多个变量并有一个数组存储

parse\_str("name=Peter&age=43",$myArray);  
print\_r($myArray);



### end() current()获取数组最后一个 第一个元素



## 数组

### 数组常见操作

<http://www.nowamagic.net/librarys/veda/detail/1175>

#### 一、数组操作的基本函数

数组的键名和值

* 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

数组的内部指针

* current($arr);返回数组中的当前单元
* pos($arr);返回数组中的当前单元
* key($arr);返回数组中当前单元的键名
* prev($arr);将数组中的内部指针倒回一位
* next($arr);将数组中的内部指针向前移动一位
* end($arr);将数组中的内部指针指向最后一个单元
* reset($arr;将数组中的内部指针指向第一个单元
* each($arr);将返回数组当前元素的一个键名/值的构造数组，并使数组指针向前移动一位
* list($key,$value)=each($arr);获得数组当前元素的键名和值

数组和变量之间的转换

extract($arr);用于把数组中的元素转换成变量导入到当前文件中，键名当作变量名，值作为变量值。注：（第二个参数很重要，可以看手册使用）使用方法 echo $a; compact(var1,var2,var3);用给定的变量名创建一个数组

#### 二、数组的分段和填充

数组的分段

* array\_slice($arr,0,3);可以将数组中的一段取出，此函数忽略键名 从第0个开始包括第0个
* array\_splice($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、数组与队列

* 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、回调函数

* array\_walk($arr,'function','words');使用用户函数对数组中的每个成员进行处理（第三个参数传递给回调函数function）
* array\_mpa("function",$arr1,$arr2);可以处理多个数组（当使用两个或更多数组时，他们的长度应该相同）
* array\_filter($arr,"function");使用回调函数过滤数组中的每个元素，如果回调函数为TRUE，数组的当前元素会被包含在返回的结果数组中，数组的键名保留不变
* array\_reduce($arr,"function","\*");转化为单值函数（\*为数组的第一个值）

#### 六、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

通过键名对数组排序

* ksort($arr);按照键名正序排序
* krsort($arr);按照键名逆序排序
* uksort($arr,"function");使用用户自定义的比较函数对数组中的键名进行排序（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）

自然排序法排序

* natsort($arr);自然排序（忽略键名）
* natcasesort($arr);自然排序（忽略大小写，忽略键名）

#### 七、数组的计算

数组元素的求和

* array\_sum($arr);对数组内部的所有元素做求和运算

数组的合并

* 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） “+”$arr1+$arr2;对于相同的键名只保留后一个
* array\_merge\_recursive($arr1,$arr2);递归合并操作，如果数组中有相同的字符串键名，这些值将被合并到一个数组中去。如果一个值本身是一个数组，将按照相应的键名把它合并为另一个数组。当数组具有相同的数组键名时，后一个值将不会覆盖原来的值，而是附加到后面

数组的差集

* array\_diff($arr1,$arr2);返回差集结果数组
* array\_diff\_assoc($arr1,$arr2,$arr3);返回差集结果数组，键名也做比较

数组的交集

* array\_intersect($arr1,$arr2);返回交集结果数组
* array\_intersect\_assoc($arr1,$arr2);返回交集结果数组，键名也做比较

#### 八、其他的数组函数

数组元素的求和

* 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
* srand((float)microtime()\*10000000); 随机种子触发器
* 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

### 索引数组与关联数组

索引数组的键名是整数并从零开始

关联数组的键名时字符串

### 识别数组元素

$arr[‘Fred’]

$arr[3]

键可以是一个字符串或一个整数，等价于证数但不以0开头的字符串被当成整数看待，

因此，$arr[3]和$array[‘3’]是一样的，$arr[‘03’]不一样

负数是有效的键

不一定需要将单个字符串加上引号，$arr[‘Fred’]和$arr[Fred]是一样的，但一般不这么做，

因为没有引号的键不能和常量区分开

Define(‘index’,5);

Echo $arr[index];----输出的是$arr[5]

### 数组初始化

### 直接赋值

$arr[0]=1;

$arr[1]=2;

$arr[2]=3;

$test[‘color’]=’red’;

$test[‘size’]=13;

$tes[‘kind’]=’my’;

### Array初始化

$arr=array(1,2,3,4);

$test=array(‘color’=>’red’,’size’=>13,’kind’=>’my’)

可以用=》来指定一个初始索引，并在后面指定一些列键值

$days=array(1=>’mondy’,’tuesday’,’wednesday’)

数组的2号元素是Tuesday，3号元素是Wednesday

如果初始索引是一个非数字的字符串，那么后面的索引是从零开始的

$day=array(‘Friday’=>’black’,’brown’,’green’)

等价于

$day=array(‘Friday’=>’black’,0=>’brown’,1=>’green’)

### 在数组末尾添加值

使用[]语法,前提数该数组的索引是数字并且索引从零开始

$arr=array(‘color’,’size’,’kind’);

$arr[]=’name’;

所以$arr[3]=name

如果是关联数组会给新的元素指定数组

$arr[‘test’=>’color];

$arr[]=’size’;

结果是

$person[0]=size;

### 指定值的范围

range()函数创建一个连续的整数或字符数组，整数或字符的值在传递给函数的两个参数之间

$number=range(1,5); //等价于 $number=array(1,2,3,4,5);

$letter=range(‘a’,’f’); //保存字母到数字 $letter=range(‘a’,b’,c’,’d’,’e’,’f’);

$reversed=range(5,1);// $reversed=array(5,4,3,2,1);

字符串参数中只有第一个字母被用于限定范围

range(‘aaa’,’’zzz’);//等价于 range(‘a’,’z’);

### 获取数组大小

count()和sizeof()函数在用法和效果上是一样的

### 填充数组

array\_pad()函数用重复的内容来创建数组

array\_pad(第一个参数是函数，第二个参数是你想要拥有元素个数，第三个元素是要给每个元素的初始值)

$arr=array(1,2,3)

$pad=array\_pad($arr,5,0); //填充后的数组 array(1,2,3,0,0)

将新值填充到数组开头

$pad=array\_pad($arr,-5,0);//填充后的数组 array(0,0,1,2,3)

将array\_pad()函数的处理结果赋值给原来的数组，可达到改变原来位置的效果

$arr=array\_pad($arr,5,0);

如果你填充元素到一个关联数组，已经存在的键将被保留，新的元素将使用从0开始的数字键

### 多维数组

数组的值也可以是数组

$row\_0=array(1,2);

$row\_1=array(3,4);

$row\_2=array(5,6);

$test=array($row\_0,$row\_1,$row\_2);

要使用变量替换来访问多维数组的 中的元素，必须将整个元素用大括号括起来

Echo “this value is {$test[1][2]}”

不适用大括号结果如下

This value is array[2]

### 析取多个值

要把一个数组的所有值都赋值到变量中，可以使用list()结构

$person(‘lili’,23,’男’)；

list($name,$age,$gender)=$person;//$name=’lili’;$age=23 $gender=’男’

如果数组中的元素个数比list()中的变量个数多，则多余的数被忽略

list($name,$age,$gender)=$person;//$name=’lili’;$age=23

如果list()中的元素个数比数组中的多，则多余的值被设为NULL

$person(‘lili’,23’)；

list($name,$age,$gender)=$person;//$name=’lili’;$age=23 $gender=NULL

在list()中用两个或多个连续的逗号跳过数组中的值

$value=range(‘a’,’e’);

List($m,,$n,,$o)=$value; //$m=’a’,$n=’c’,$o=’e’;

### 切割数组

array\_slice()用来析取数组的一个子集

$subject=array\_slice(array,offset,length)

Offset指定要赋值元素的初始元素。Length指定要复制的值的个数

$people=array(1,2,3,4,5);

$middle=array\_slice($people,2,2) // 结果为 $middle=array(3,4)

只有在索引数组上使用array\_slice()才有效，

$person=array(‘name’=>’lili’,’age’=>35,’wife’=>’betty’);

$test=array\_slice($person,1,2);//结果为 $test=array(0=>35,1=>’betty’);

结合list()和array\_slice()使用

$order=array(1,2,3,4,5,6);

List($test1,$test2)=$array\_slice($order,1,2);

//结果为$test1=2;$test2=3;

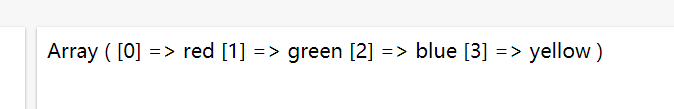
### 合并数组

Array\_merge($a1,$a2);

$a1=array("red","green");

$a2=array("blue","yellow");

print\_r(array\_merge($a1,$a2));



### 数组分块

array\_chunk()函数把数组划分成小数组或固定大小的数组

$chunk=array\_chunk(array,size,preserve\_keys)

Size为划分数组的长度

Preserve\_keys是一个布尔值，它指定新数组的元素是否有和原数组相同的键，还是从0开始的新数字键,默认是分配新键值的

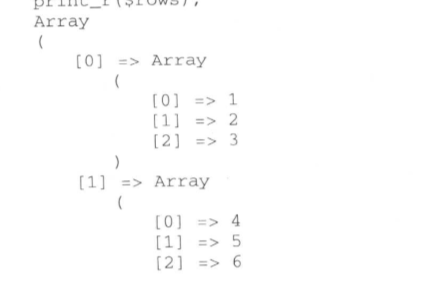
True 保留原始数组的键名

False 默认，每个新数组块使用从0开始

$num=range(1,7)

$rows=array\_chunk($num,3)

打印结果：



### compact()创建数组

创建一个包含变量名和它们的数值的数组

$firstname=”peter”;

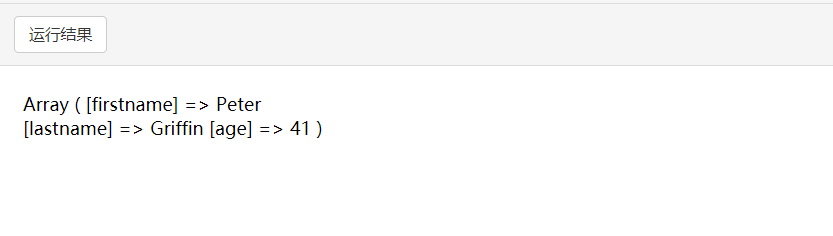
$lastname=”Griffin”;

$age=”41 ”;

$result=compact(“firstname”,”lastname”,”age”);

Print\_r($result);

结果：



### array\_push()

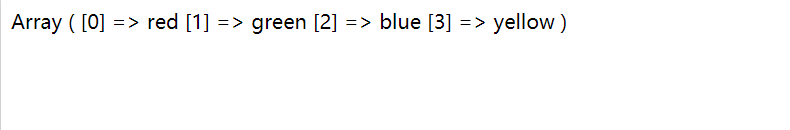
向第一个参数的数组的尾部添加一个或多个元素

$a=array(“red”,”green”);

array\_push($a,”blue”,”yellow”);

print($a);

结果：



### explode与implode

<https://www.cnblogs.com/usa007lhy/p/5278801.html>

explode(‘,’,$arr);按字符串分割数组

implode(‘,’,$arr);将素组按给定字符拼接成字符串

### 键和值

array\_keys()根据内部顺序返回一个仅由数组中的键组成的数组，等价于array\_of\_keys()

$person=array(‘name’=>’lili’,’age’=>13,’gender’=>’女’);

$keys=array\_keys($person); //结果 $key=array(‘name’,’age’,’gender’);

array\_values()根据内部顺序返回一个仅由数组中的值组成的数组

$person=array(‘name’=>’lili’,’age’=>13,’gender’=>’女’);

$values=array\_values($person); //结果 $ values =array(‘lili’,13,’女);

#### array\_keys()操作二维数组

$a=array(

array('x'=>'w'),

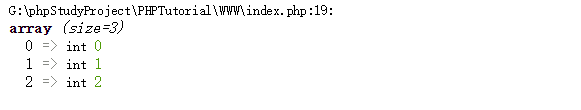
array('v'=>'e'),

array('b'=>'g')

);

var\_dump(array\_keys($a));

结果：



array\_values()操作二维数组

$a=array(

array('x'=>'w'),

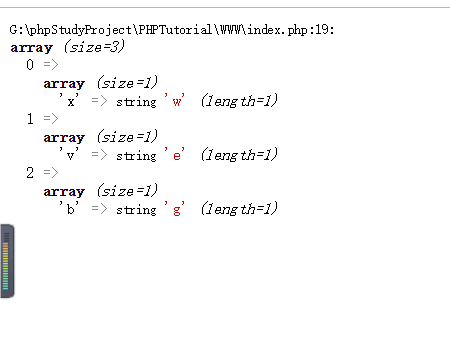
array('v'=>'e'),

array('b'=>'g')

);

var\_dump(array\_values($a));

结果：

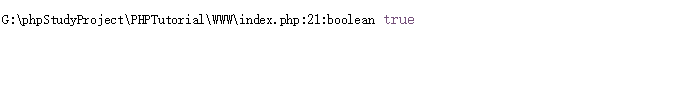


### 检查元素是否存在

Array\_key\_exists(key,array)用来检查数组中是否存在某个键值的元素，存在返回true，不存在返回false

$a=array('q'=>132,'e'=>'s');

var\_dump(array\_key\_exists('q',$a));



$a=array(

'b'=>array('x'=>'w'),

array('v'=>'e'),

array('b'=>'g')

);

var\_dump(array\_key\_exists('b',$a));



Isset也可以判断，不存在返回NULL存在返回false

### 在数组中删除和插入元素

array\_splice()可以在数组中删除和插入元素，并且可以用被删除的元素创建一个数组

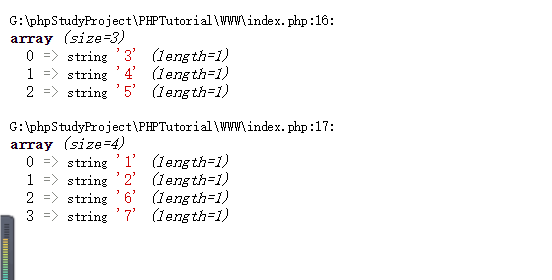
操作二维数组和一维数组一样，索引数组和关联数组一样

$subject=array('1','2','3','4','5','6','7');

$remove=array\_splice($subject, 2,3);

var\_dump($remove);

var\_dump($subject);



原数组从第二个位置往后删除三个元素，array\_splice()得到的是删除后的数组

$remove=array\_splice($subject,2);从第二个位置删除的最后

#### 在删除的位置加元素

被删除的元素个数不需要和插入的元素个数一致

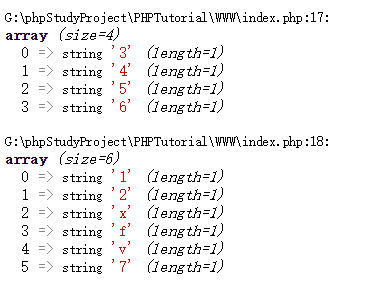
$subject=array('1','2','3','4','5','6','7');

$a=array('x','f','v');

$remove=array\_splice($subject, 2,4,$a);

var\_dump($remove);

var\_dump($subject);



#### 新元素插入数组

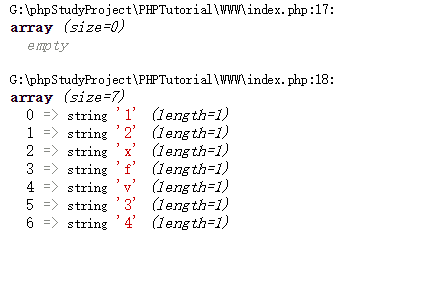
$subject=array('1','2','3','4');

$a=array('x','f','v');

$remove=array\_splice($subject, 2,0,$a);

var\_dump($remove);

var\_dump($subject);



### 在数组和变量间转换

**extract()将关联数组键值转成变量名，值转成变量的值**

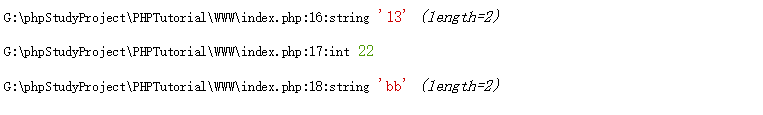
$test=array('a'=>'13','b'=>22,'c'=>'bb');

extract($test);

var\_dump($a);

var\_dump($b);

var\_dump($c);



**compact()将变量以变量名和值转成关联数组**

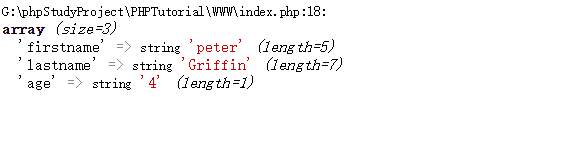
$firstname="peter";

$lastname="Griffin";

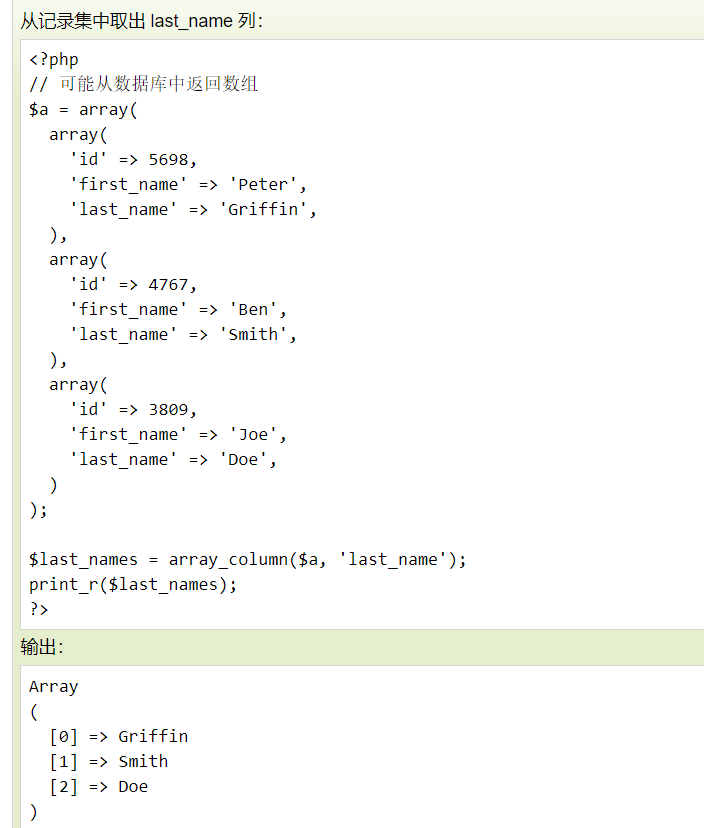
$age="4";

$result=compact("firstname","lastname","age");

var\_dump($result);



### Array\_column返回数组中某列的值



## array\_multisort按照数组中某个值排序

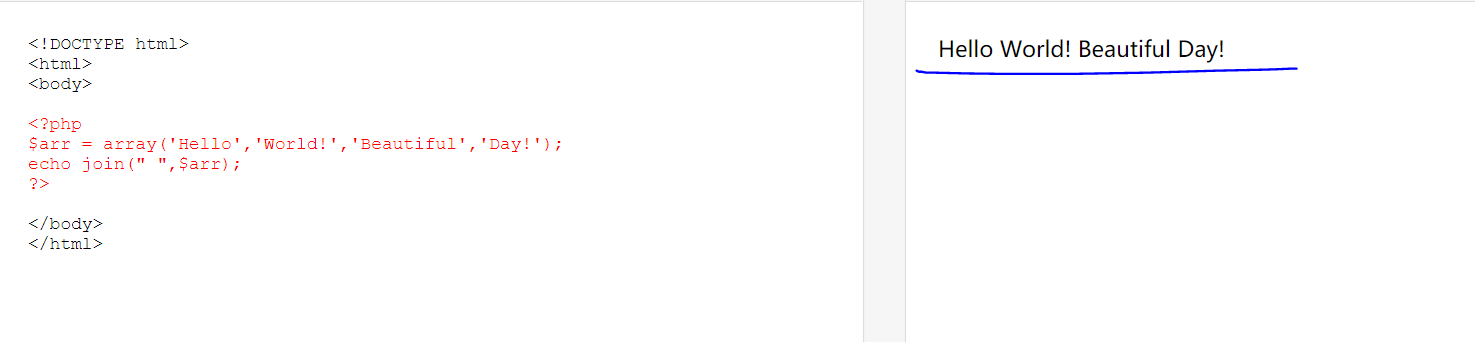
//根据字段last\_name对数组$data进行降序排列

**$last\_names = array\_column($data,'last\_name');**

**array\_multisort($last\_names,SORT\_DESC,$data);**

## join把数组元素组成一个字符串

join(separator,array)

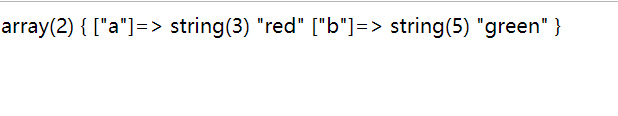


## Array\_unique();删除数组重复元素

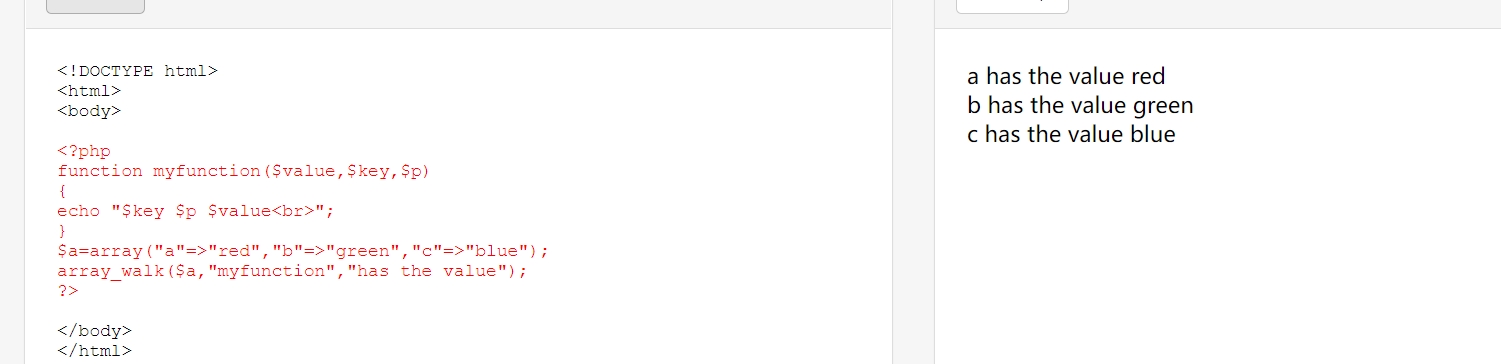
array\_unique() 函数用于移除数组中重复的值。如果两个或更多个数组值相同，只保留第一个值，其他的值被移除。

array("a"=>"red","b"=>"green","c"=>"red");

var\_dump(array\_unique($a));



## array\_walk()对数组中的每个元素应用用户自定义函数



array\_walk ( array &$array , callable $callback [, mixed $userdata = NULL ] ) : bool

回调函数的参数，第一个是元素值，第二个是元素键名，第三个是可选的 $userdata。

如果只想改变数组值，第一个参数可使用引用传递，即在参数前加上 &。

## in\_array判断数组是否存在某个值

in\_array(value,array,type)

如果给定的值 *search* 存在于数组 *array* 中则返回 true。如果第三个参数设置为 true，函数只有在元素存在于数组中且数据类型与给定值相同时才返回 true。

如果没有在数组中找到参数，函数返回 false。

## array\_search() 函数在数组中搜索某个键值，并返回对应的键名。

array\_search(value,array,strict)

<?php

$a=array("a"=>"red","b"=>"green","c"=>"blue");

echo array\_search("red",$a);

?>

输出a

## array\_reduce 函数发送数组中的值到用户自定义函数，并返回一个字符串

array\_reduce() 函数发送数组中的值到用户自定义函数，并返回一个字符串

参数：

|  |  |
| --- | --- |
| *array* | 必需。规定数组。 |
| *myfunction* | 必需。规定函数的名称。 |
| *initial* | 可选。规定发送到函数处理的第一个值。 |

function myfunction($v1,$v2)

{

return $v1 . "-" . $v2;

}

$a=array("Dog","Cat","Horse");

print\_r(array\_reduce($a,"myfunction",12));

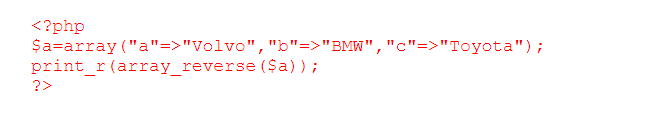
//输出 12-Dog-Cat-Horse

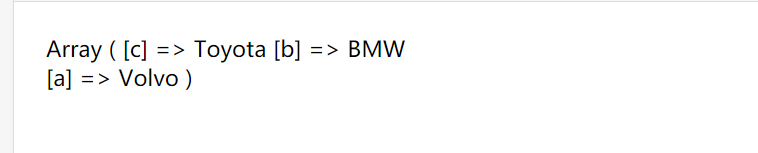
*Initial 和array只作为参数返送到myfunction处理返回的结果再次作为第一个参数 和array剩下的下一个数作为参数发送的myfunctio如此反复*

## array\_reverse() 函数返回翻转顺序的数组。

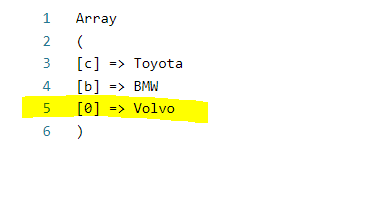
array\_reverse(array,preserve)

|  |  |
| --- | --- |
| *array* | 必需。规定数组。 |
| *preserve* | 可选。规定是否保留原始数组的键名。 如果设置为 TRUE 会保留数字的键。 非数字的键则不受这个设置的影响，总是会被保留。 可能的值：   * true * false |





$a=**array**(1=>**"Volvo"**,**"b"**=>**"BMW"**,**"c"**=>**"Toyota"**);  
*print\_r*(*array\_reverse*($a,**false**));



## array\_pad()将指定数量的带有指定值的与元素插入到数组

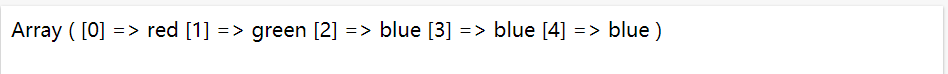
array\_pad(*array*,*size*,*value*)

将指定数量的带有指定值的与元素插入到数组

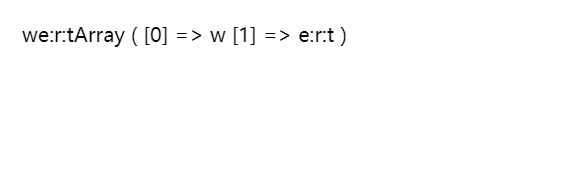
|  |  |
| --- | --- |
| **参数** | **描述** |
| *array* | 必需。规定数组。 |
| *size* | 必需。规定从函数返回的数组中元素的个数。 |
| *value* | 必需。规定从函数返回的数组中新元素的值。 |

$a=array("red","green");

print\_r(array\_pad($a,5,"blue"));



$pipe=**"w:e:r:t"**;  
[$name,$value]=*array\_pad*(*explode*(**':'**,$pipe,2),2,[]);  
*print\_r*($name);  
*print\_r*($value);  
*print\_r*([$name,$value]);

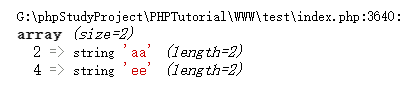


## array\_filter用回调函数过滤数组中的元素

用回调函数过滤数组中的元素

该函数把输入数组中的每个键值传给回调函数。如果回调函数返回 true，则把输入数组中的当前键值返回给结果数组。数组键名保持不变。

$parameters=**array**(**'1'**=>1,**'2'**=>**'aa'**,**'3'**=>2,**'4'**=>**'ee'**);  
$a=*array\_filter*($parameters, **function** ($value) {  
 **return** *is\_string*($value) && *strlen*($value) > 0;  
});  
*var\_dump*($a);



## array\_replace\_recursive递归地使用第二个数组（$a2）的值替换第一个数组（$a1）的值

递归地使用第二个数组（$a2）的值替换第一个数组（$a1）的值

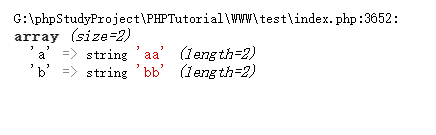
array\_replace\_recursive(array1,array2,array3...)

|  |  |
| --- | --- |
| **参数** | **描述** |
| *array1* | 必需。指定一个数组。 |
| *array2* | 可选。指定一个要替换 *array1* 的值的数组。 |
| *array3,...* | 可选。指定多个要替换 *array1* 和 *array2, ...* 的值的数组。后面数组的值将覆盖之前数组的值 |

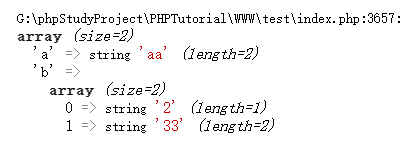
### 1. 一个键存在于第一个数组array1中同时也存在与第二个数组array2中

如果一个键存在于第一个数组array1中同时也存在与第二个数组array2中，则第一个数组array1中的值将被第二个数组中的值替换。

$a1=**array**(**'a'**=>**'123'**,**'b'**=>**'1231'**);  
$a2=**array**(**'a'**=>**'aa'**,**'b'**=>**'bb'**);  
$res=*array\_replace\_recursive*($a1,$a2);  
*var\_dump*($res);

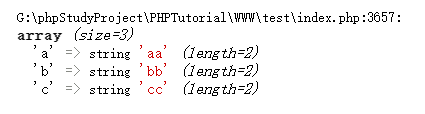


$a1=**array**(**'a'**=>**'123'**,**'b'**=>**'1'**);  
$a2=**array**(**'a'**=>**'aa'**,**'b'**=>**array**(**'2'**,**'33'**));  
$res=*array\_replace\_recursive*($a1,$a2);  
*var\_dump*($res);



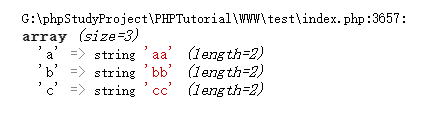
### 如果一个键仅存在于第一个数组array1中，它将保持不变

$a1=**array**(**'a'**=>**'123'**,**'b'**=>**array**(**'2'**,**'33'**),**'c'**=>**'cc'**);  
$a2=**array**(**'a'**=>**'aa'**,**'b'**=>**'bb'**);  
$res=*array\_replace\_recursive*($a1,$a2);  
*var\_dump*($res);



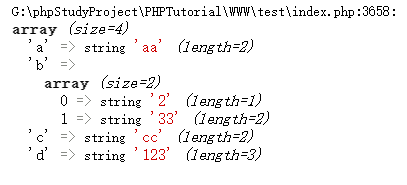
### 3.如果一个键存在于第二个数组array2中，但不存在于第一个数组array1中，则在第一个数组array1 中创建这个元素。

$a1=**array**(**'a'**=>**'123'**,**'b'**=>**array**(**'2'**,**'33'**));  
$a2=**array**(**'a'**=>**'aa'**,**'b'**=>**'bb'**,**'c'**=>**'cc'**);  
$res=*array\_replace\_recursive*($a1,$a2);  
*var\_dump*($res);



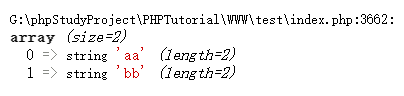
### 4.如果传递了多个替换数组，他们将被按顺序依次处理，后面数组的值将覆盖之前的数组

$a1=**array**(**'a'**=>**'123'**,**'b'**=>**array**(**'2'**,**'33'**));  
$a2=**array**(**'a'**=>**'aa'**,**'b'**=>**'bb'**,**'c'**=>**'cc'**);  
$a3=**array**(**'d'**=>**'123'**,**'b'**=>**array**(**'2'**,**'33'**));  
$res=*array\_replace\_recursive*($a1,$a2,$a3);  
*var\_dump*($res);



### 5.如果没有为每个数组指定一个键，改函数等同于array\_replace()

$a1=**array**(**'123'**,**'1231'**);  
$a2=**array**(**'aa'**,**'bb'**);  
$res=*array\_replace\_recursive*($a1,$a2);  
*var\_dump*($res);



## End()获取数组的最后一个元素

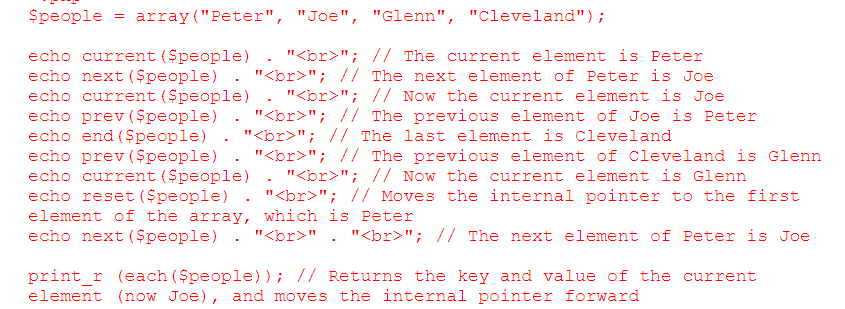
[current()](https://www.runoob.com/php/func-array-current.html) - 返回数组中的当前元素的值。

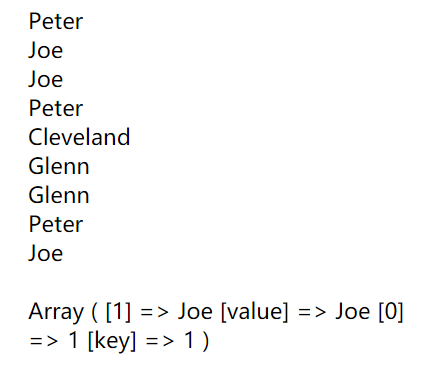
[next()](https://www.runoob.com/php/func-array-next.html) - 将内部指针指向数组中的下一个元素，并输出。

[prev()](https://www.runoob.com/php/func-array-prev.html) - 将内部指针指向数组中的上一个元素，并输出。

[reset()](https://www.runoob.com/php/func-array-reset.html) - 将内部指针指向数组中的第一个元素，并输出。

[each()](https://www.runoob.com/php/func-array-each.html) - 返回当前元素的键名和键值，并将内部指针向前移动。





## Empty(0)==true

Isset判断’‘为true

## chunk\_split()

|  |  |
| --- | --- |
| *string* | 必需。规定要分割的字符串。 |
| *length* | 可选。数字值，定义字符串块的长度。默认是 76。 |
| *end* | 可选。字符串值，定义在每个字符串块末端放置的内容。默认是 \r\n。 |

chunk\_split(*string*,*length*,*end*)

把字符串分割为一连串更小的部分

### 数组逆序

array\_reverse($arr)

## Base64\_encode和base64\_decode

## array\_unshift向数组插入元素

array\_unshift() 函数用于向数组插入新元素。新数组的值将被插入到数组的开头。

返回值：返回新数组中元素的个数。

array\_unshift(*array,value1,value2,value3...*)

可以插入一个或多个值

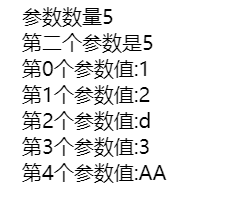
插入元素 "blue" 到数组中

$a=array("a"=>"red","b"=>"green");  
array\_unshift($a,"blue");  
print\_r($a);

## func\_get\_args

返回一个含有函数参数列表的数组

**function** test(){  
 *//参数数量* $number=*func\_num\_args*();  
 **echo "参数数量"**.$number.**"<br>"**;  
 **if**($number>=2){  
 **echo "第二个参数是"**.*func\_num\_args*(1).**"<br>"**;  
 }  
 $arg\_list=*func\_get\_args*();  
 **for** ($i = 0; $i < $number; $i++) {  
 **echo "第**{$i}**个参数值:**{$arg\_list[$i]}**<br />\n"**;  
 }  
}  
test(1,2,**'d'**,3,**'AA'**);



## is\_callable

检查参数是否为合法的可调用结构

**function** test(){  
 **return 'sss'**;  
}  
$falge=*is\_callable*(test());  
*var\_dump*($falge);*//输出true*

# 图像处理

### GD2函数库

Php通过GD2函数库操作图形

文档：

<https://www.php.cn/manual/view/5873.html>

php课件 4

### 画图操作步骤

#### 创建 一个背景图像（画布）

**创建空白的背景**

imageCreate(int x ,int y);

x背景宽度，单位像素

y背景高度，单位像素

**使用已有图形创建背景**

ImageCreateFormPNG(string $fileName);

imageCreateFormJpeg(string $filename);

imageCreateGif(string $filename)

**选择颜色**

imageColorAllocate(int $img,int red, int blue);//rgb的值

#### 在背景上绘制图像或输入文本。

计算文字所占区域

imageTTFBBox(int $size ,int $angle ,string $fontfile ,string $text);

$size 字形尺寸、$angle 字型角度、$fontfile 字体

$text 字符串内容

指定字体输出字符串

imagettftext(int img,int size,int angle,int x,int y,int col,string fontfile,strint text);

#### 输出图像

imagePNG(int $img string $filename);

imageJPEG(int $img string $filename);

imageGIF(int $img string $filename);

#### 释放所有资源

Imagedestory(int $img)

案列：

//创建画布

$im=imagecreate(150,40);

//配置红色背景

ImageColorAllocate($im,255,0,0);

//定义黑色文字

$textcolor= ImageColorAllocate($im,0,0,0);

//水平画一行子字

Imagestring($im,5,30,12,’Hello,world’,$ textcolor);

//输出图像

header(“content-type:image/png”);//设置图片类型

imagepng($img);

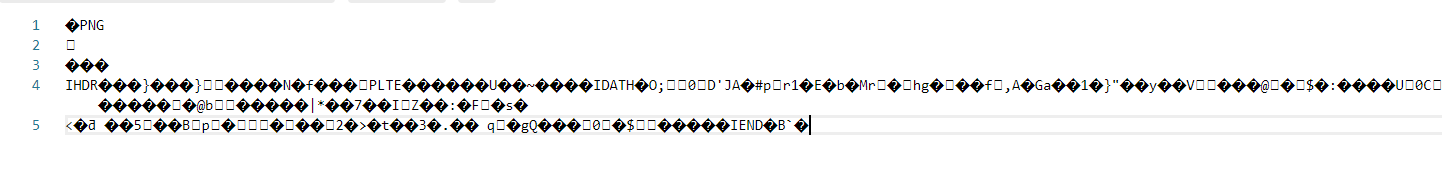
//释放资源

Imagedeatory($im);

## file\_get\_contents()

把整个文件读入一个字符串中

返回的是二进制流



## Imagecreatefromstring()

从字符串的图像流新建一图像

## Imagecopyresampled()

拷贝部分图像并调整大小

bool imagecopyresampled ( resource $dst\_image , resource $src\_image , int $dst\_x , int $dst\_y , int $src\_x , int $src\_y , int $dst\_w , int $dst\_h , int $src\_w , int $src\_h )



*/\*\*  
 \* 将logo嵌套在二维码中  
 \* 参数一：目标图象连接资源  
 \* 参数二：源图象连接资源。  
 \* 参数三：目标 X 坐标点  
 \* 参数四： 目标 Y 坐标点 logo图片嵌入后坐标  
 \* 参数五： 源的 X 坐标点 logo图片嵌入前坐标 相对移动原图X坐标  
 \* 参数六：源的 Y 坐标点  
 \* 参数七：目标宽度  
 \* 参数八：目标高度 logo图片嵌入后高度  
 \* 参数九：源图像宽度 logo图片嵌入前高度  
 \* 参数十：源图像高度  
 \*/***if** ($logo !== **FALSE**) {  
 $QR = *imagecreatefromstring*(*file\_get\_contents*($QR));  
 $logo = *imagecreatefromstring*(*file\_get\_contents*($logo));  
 $QR\_width = *imagesx*($QR);*//二维码图片宽度* $QR\_height = *imagesy*($QR);*//二维码图片高度* $logo\_width = *imagesx*($logo);*//logo图片宽度* $logo\_height = *imagesy*($logo);*//logo图片高度* $logo\_qr\_width = $QR\_width\*0.4; *//logo图片在二维码图片中宽度大小* $scale = $logo\_width/$logo\_qr\_width;  
 $logo\_qr\_height = $logo\_height/$scale; *//logo图片在二维码图片中高度大小  
 // 目标 X 坐标点* $from\_width = ($QR\_width - $logo\_qr\_width)/2;  
 *//重新组合图片并调整大小  
 imagecopyresampled*($QR, $logo, $from\_width, $from\_width, 0, 0, $logo\_qr\_width, $logo\_qr\_height, $logo\_width, $logo\_height);  
 *imagepng*($QR,**'SWE.png'**);  
}

## Imagesx()

获取图像高度

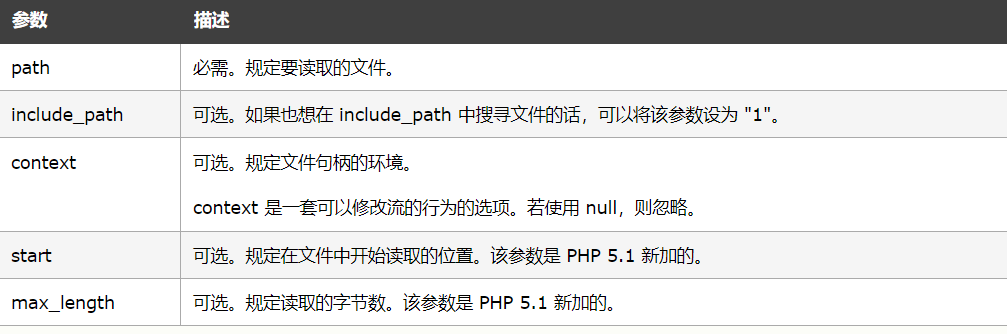
## imagesy()

获取图像宽度

# 文件函数

## file\_get\_contents读取文件

file\_get\_contents(*path*,*include\_path*,*context*,*start*,*max\_length*)



# Php 类::的问题

# Php反射机制

# Php7

$a ?? 0 等同于 isset($a) ? $a : 0

# Php5.3新特性

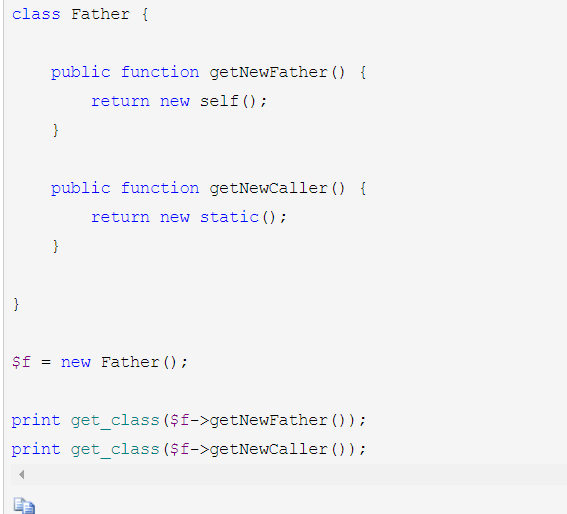
### 1.New static()

New static()是php5.3引入的新特性和new self（）一样都是new 一个新的对象

New static（）与new self()区别

参考：<https://www.cnblogs.com/shizqiang/p/6277091.html>

在没有继承的时候二者无区别，有继承的情况下，new static()返回调用者的实例，new self（）返回的实例的该类的实例



输出：Father Father

class Sun1 extends Father {

}

class Sun2 extends Father {

}

$sun1 = new Sun1();  
$sun2 = new Sun2();

print get\_class($sun1->getNewFather());//Father

print get\_class($sun1->getNewCaller());//sun1

print get\_class($sun2->getNewFather());//Father

print get\_class($sun2->getNewCaller());//sun2

### 2、…使用

#### 使用 ... 运算符定义变长参数函数

function f($req, $opt = null, ...$params) {  
    // $params 是一个包含了剩余参数的数组  
    printf('$req: %d; $opt: %d; number of params: %d'."\n",  
           $req, $opt, count($params));  
}  
  
f(1);  
f(1, 2);  
f(1, 2, 3);  
f(1, 2, 3, 4);  
f(1, 2, 3, 4, 5);

#### 使用 ... 运算符进行参数展开

在调用函数的时候，使用 ... 运算符， 将 [数组](https://www.php.net/manual/zh/language.types.array.php) 和 **可遍历** 对象展开为函数参数

function add($a, $b, $c) {  
    return $a + $b + $c;  
}  
  
$operators = [2, 3];  
echo add(1, ...$operators);

# Php反射

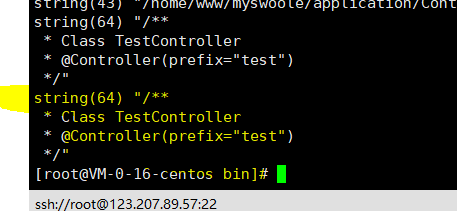
## 作用

PHP 5 具有完整的反射 API，添加了对类、接口、函数、方法和扩展进行反向工程的能力。 此外，反射 API 提供了方法来取出函数、类和方法中的文档注释。

## ReflectionClass 类

**ReflectionClass** 类报告了一个类的有关信息。

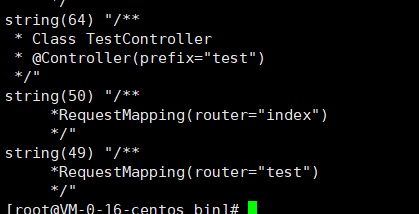
$obj=**new** TestController();  
$reflect=**new** \ReflectionClass($obj);*//类的注解  
var\_dump*($reflect->getDocComment());



### ReflectionClass::getMethods

获取类的方法的一个数组。

**foreach** ($reflect->getMethods() **as** $method){  
 *var\_dump*($method->getDocComment());  
}



## ReflectionMethod 类

**ReflectionMethod** 类报告了一个方法的有关信息。

# Php链式调用实现

# Php 数学函数

<https://www.php.net/manual/zh/function.bcdiv.php>

## [bcadd](https://www.php.net/manual/en/function.bcadd.php) —添加两个任意精度数字

**bcadd** （字符串 $num1 ，字符串 $num2 ， int | null $scale =**null** ）：字符串

$scale保留几位小数

## [bccomp](https://www.php.net/manual/en/function.bccomp.php) —比较两个任意精度的数字

**bccomp** （字符串 $num1 ，字符串 $num2 ， int | null $scale =**null** ）： int

scale

可选scale参数用于设置将在比较中使用的小数点后的位数。

如果两个操作数是相等的，返回0，

如果 num1是比较大的 num2，返回1

Num1<num2否则返回-1。

echo bccomp('1', '2') . "\n";   // -1  
echo bccomp('1.00001', '1', 3); // 0  
echo bccomp('1.00001', '1', 5); // 1

## [bcdiv](https://www.php.net/manual/en/function.bcdiv.php) —除以两个任意精度的数字

scale

此可选参数用于设置结果中小数点后的位数

echo bcdiv('105', '6.55957', 3);  // 16.007

## [bcmod](https://www.php.net/manual/en/function.bcmod.php) —获取任意精度数的模

## [bcmul](https://www.php.net/manual/en/function.bcmul.php) —将两个任意精度数字相乘

scale

此可选参数用于设置结果中小数点后的位数

echo bcmul('1.34747474747', '35', 3); // 47.161  
echo bcmul('2', '4'); // 8

## [bcpow](https://www.php.net/manual/en/function.bcpow.php) —将任意精度数提高到另一个

## [bcpowmod](https://www.php.net/manual/en/function.bcpowmod.php) —将任意精度数提高为另一个，并减少指定的模数

## [bcscale](https://www.php.net/manual/en/function.bcscale.php) —为所有bc数学函数设置或获取默认的scale参数

## [bcsqrt](https://www.php.net/manual/en/function.bcsqrt.php) —获取任意精度数的平方根

## [bcsub](https://www.php.net/manual/en/function.bcsub.php) —从另一个减去一个任意精度数

$a = '1.234';  
$b = '5';  
  
echo bcsub($a, $b);     // -3  
echo bcsub($a, $b, 4);  // -3.7660