数组

Php 数组中的元素顺序，与下标无关！

数组定义（赋值）：

```
$arr1 = array(3, 11, 5, 18, 2 );//这是最常见的数组，下标为“默认下标”，就是从 0 开始的整数；
$arr2 = array("a"=>3, "bb"=>11, "cc123"=>5, 'd1'=>18, 'xyz'=>2 );关联数组，下标为字符串，常见
$arr3 = array(1=>3, 10=>11, 3=>5, 0=>18, 88=>2 );下标可以人为给定；
$arr4 = array(1=>3, 'a1'=>11, 3=>5, 'mn'=>18, 88=>2 );下标可以数字和字符串混合使用；
$arr5 = array(5=>3, 11, 3=>5, 'mn'=>18, 2 );//有指定下标，也有“自动下标”，
                                            //此时下标为：5，6，3，"mn", 7
                                            //可见，自动下标为“前面最大数字下标+1”
$arr6 = array(5=>3, 7.7=>11, 3=>5, 'mn'=>18, 2 );//此时下标为：5，7，3，"mn", 8
$arr7 = array(5=>3, true=>11, false=>5, 'mn'=>18, 2 );//此时下标为：5，1，0，"mn", 6

$arr8 = array(1=>3, 3=>33, true=>11, );//此时下标为：1，3,其对应值为：11, 33
                                       //下标如果有重复，后面的值覆盖前面的值；

$arr9 = array(1=>3, -3=>33, 11, );      //此时下标为：1， -3， 2

$arr9 = array(1=>3, -3=>33, 11, );      //此时下标为：1， -3， 2，注意：最后一个逗号“可以有”。
```

其他形式：

```
$arr10[] = 3;
$arr10[] = 11;
$arr10[] = 5;  //该数组下标为 0,1,2，常规情况
$arr11['a'] = 3;
$arr11['bb'] = 11;
$arr11['cc123'] = 5;     //该数组下标为'a','bb','cc123'，常规情况
$arr12[1] = 3;
$arr12[] = 11; //此时下标为 2
$arr13['cc123'] = 5;     //该数组下标为 1,2,'cc123'
```

特别注意：php 中，数组元素的顺序是由“放入”的顺序决定，而不是下标。

数组取值：

```
$v1 = $arr[0];
$i = 3;
$v2 = $arr[$i];
```

总体上，可以将取得一个数组的单元的值，看组取得一个变量的值完全一样！！！

数组分类

按键值关系来分：

索引数组：通常认为，如果一个数组的下标是严格按照从 0 开始的连续的整数作为下标，则称其为索引数组——就是类似 js 数组的下标。例如：

```
$arr1 = array(3,  11,  5,  18,  2  );//这是最常见的数组，下标为“默认下标”，就是从 0 开始的整数；
```

关联数组：通常认为，如果一个数组的下标都是一个“字符串”并一定程度上表名了该单元的“含义”，则称为关联数组，例如：

```
$conf = array(
    'host'=>"localhost"  ,
    'port'=>3306  ,
    'username'=>'root'  ,
    'password' => '123'  ,
);
```

混合数组：既有数字下标，也有字符下标的情况：

```
$arr4 = array(1=>3,  'a1'=>11,  3=>5,  'mn'=>18,  88=>2  );下标可以数字和字符串混合使用；
```

在 PHP 中创建数组

在 PHP 中，array() 函数用于创建数组：

```
array();
```

在 PHP 中，有三种类型的数组：

- **数值数组** - 带有数字 ID 键的数组
- **关联数组** - 带有指定的键的数组，每个键关联一个值
- **多维数组** - 包含一个或多个数组的数组

按数组的维数（复杂程度）分类：

一维数组：

```
$a = array(1, 11, 111);
$b = array(2, 22, 222);
$c = array(3, 33, 333);
```

二维数组：

```
$dd = array(
    array(1, 11, 111),
    array(2, 22, 222),
    array(3, 33, 333)
);
```

多维数组：无非就是继续里面再用数组代替。

数组的基本使用

求一维数组的平均值：

```php
<meta http-equiv="Content-Type" content="text/html;charset=UTF-8" />
<?php
//这一行php的作用，跟上一行meta的作用一样：告诉浏览器，请用utf8显示本网页
header("Content-Type:text/html;charset=UTF-8");

//求一个一维数组的平均值：
$a = array(1, 11, 111, 1111);
$len = count($a);    //长度
$sum = 0;    //用于总和
$c = 0;      //用于存储数组的个数
for($i = 0; $i<$len; ++$i){
    $sum += $a[$i]; //累加思想
    ++$c;           //计数思想
}
echo "<br />平均值为：" . ($sum/$c);
```

求二维数组的平均值：

```php
//求一个二维数组的平均值：
$dd = array(
    array(1, 11, 111),
    array(2, 22, 222, 2222),
    array(3, 33, 333, 3333,33333)
);
$len = count($dd);//长度，这里是3
$sum = 0;    //用于总和
$c = 0;      //用于存储数组的个数
for($i = 0; $i < $len; ++$i){
    $temp = $dd[$i];    //这是一个一维数组！
    $len2 = count($temp);//该一维数组的长度
    for($k = 0; $k < $len2; ++$k){
        $sum += $temp[$k];//累加
        ++$c;                //计数
    }
}
echo "<br />平均值为：" . ($sum/$c);
```

```
平均值为：308.5
平均值为：3302.1666666667
```

求一维数组的最大值：

```php
//求一个一维数组的最大值，及其对应下标：
$arr3 = array(13,  8,  5,  11,  22,  2);
$max = $arr3[0];    //将第一项放入$max中，
                    //并试图使用该变量来存储最终的“最大值”
$pos = 0;//第一个下标，并试图使用该变量来存储最终的“最大值所在下标”
$len = count($arr3);    //长度
for($i = 0; $i < $len; ++$i){
    if($arr3[$i] > $max){
        $max = $arr3[$i];
    }
}
echo "<br />最大值为：" . ($max);
```

求交换一个一维数组的最大值和最小值的位置：

```php
$arr3 = array(13,  38,  5,  11,  22,  2);
echo "<br />交换之前：";  print_r($arr3);
$max = $arr3[0];          //将第一项放入$max中，
                          //并试图使用该变量来存储最终的“最大值”
$pos = 0;//第一个下标，并试图使用该变量来存储最终的“最大值所在下标”
$min = $arr3[0];     //同理！
$pos2 = 0;           //同理
$len = count($arr3);     //长度
for($i = 0; $i < $len; ++$i){
    if($arr3[$i] > $max){
        $max = $arr3[$i];
        $pos = $i;
    }
    if($arr3[$i] < $min){
        $min = $arr3[$i];
        $pos2 = $i;
    }
}
echo "<br />最大值为：" . ($max) . "，其下标为：$pos";
echo "<br />最小值为：" . ($min) . "，其下标为：$pos2";

//然后才开始交换：
$t = $arr3[$pos];    //$arr3[$pos]就是数组的最大值的单元
$arr3[$pos] = $arr3[$pos2];//$arr3[$pos2]就是数组的最小值的单元
$arr3[$pos2] = $t;
```

有关交换，再说两句：

```
$a = array( 3,  11,  5,  7,  20,  18);          //下标是 0,1,2,3,4,5
需求 1：交换数组第 0 项和第 3 项：
    $v1 = $a[0];
    $v2 = $a[3];
    $t = $v1;
    $v1 = $v2;
    $v2 = $t;      //这种做法根本不行，因为 v1，v2 只是 2 个变量，跟数组没有关系了！
    正确的做法是：
    $t = $a[0];
    $a[0] = $a[3];
    $a[3] = $t;
```

数组的基本遍历

foreach 基本语法

```
foreach( $数组变量名  as  |【$key =>】  $value ){
    //循环体；这里可以去“使用” $key 和 value；
    //$key 和 $value 就是该遍历语句一次次取得的数组的每一个单元（项）的下标和对应值。
    //而且，它总是从数组的开头往后按顺序取数据。
}
```

数组的指针操作及遍历原理：

首先，看看数组的一个“形象图”：

$arr4 = array(1=>3, 'a1'=>11, 3=>5, 'mn'=>18, 88=>2);

可以将其以视觉化的方式理解为：

数组下标：	1	“a1”	3	“mn”	88
对应数据：	8	11	5	18	2

其中，该箭头，就是数组内部的所谓“指针”——注意，不可见，不可输出，只是一种辅助理解的图形！

说明：

1，该箭头，就是数组内部的所谓“指针”

2，默认情况下，该指针指向数组的第一个单元。

3，数组的有关单元的操作，如果没有指定下标，则就是指对该指针指向的单元的操作。

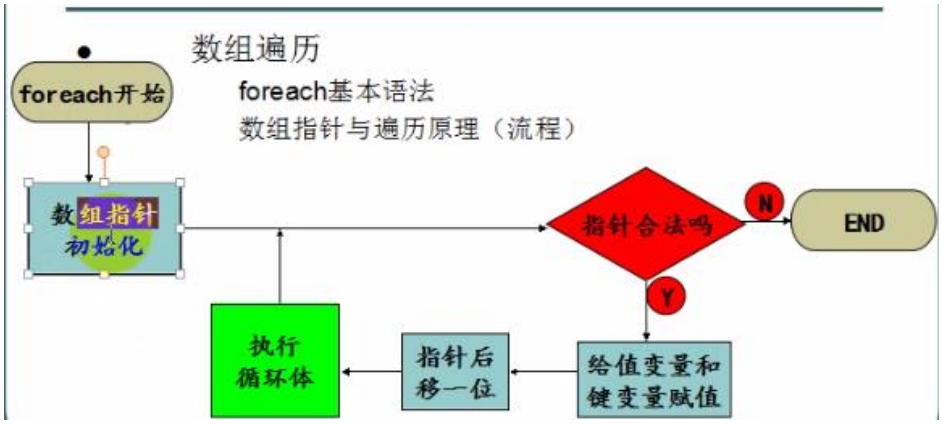
4，所谓遍历，其实就是一次次取得当前单元的键和值，并放入对应的变量$key, $value,然后移动指针到下一个单元。

则，数组，作为一个“总体数据单位”，有如下指针操作函数可以使用：

```
1，$v1 = current($数组);        //获得数组的当做指针所在单元的“值”；
2，$v2 = key($数组);            //获得数组的当做指针所在单元的“键”（下标）；
3，$v3 = next($数组);           //先将数组的指针移向下（后）一个单元，然后取得该新单元的值；
4，$v4 = prev($数组);           //先将数组的指针移向上（前）一个单元，然后取得该新单元的值；
5，$v5 = end($数组);            //先将数组的指针直接移向最后一个单元，然后取得该新单元的值；
6，$v6 = reset($数组);          //先将数组的指针直接移向第一个单元，然后取得该新单元的值；
```

foreache 遍历流程原理图：

```
foreach( $数组变量名  as  $key => $value ){
    //循环体；这里可以去“使用” $key 和 value；
    //$key 和 $value 就是该遍历语句一次次取得的数组的每一个单元（项）的下标和对应值。
    //而且，它总是从数组的开头往后按顺序取数据。
}
```



遍历之后的指针位置

```php
echo "<h1>下面研究遍历之后的指针位置：</h1>";
$arr5 = array(1=>3,  'a1'=>11,  3=>5,  "mn"=>18,  88=>2  );
foreach($arr5  as  $key => $value){
    echo "<br />$key => $value";
}
$k = key($arr5);
$v = current($arr5);
echo "<br />此时（遍历之后）,“位置”为："; var_dump($k);
echo "<br />此时（遍历之后）,对应“值”为："; var_dump($v);
```

输出结果为：

```
初始，单元的下标和值分别为：1,3
然后，现在当前单元的下标和值分别为：a1,11
连移3次next后，则当前单元的下标和值分别为：88,2
指针到最后，然后再移动一下，则结果为：“下标和值”分别为：,
实际情况，此时v3（值） 为：bool(false)
实际情况，此时v4（键） 为：NULL
```

下面研究遍历之后的指针位置：

```
1 => 3
a1 => 11
3 => 5
mn => 18
88 => 2
此时（遍历之后）,位置为：NULL
此时（遍历之后）,对应“值”为：bool(false)
```

可见：遍历之后，指针已经超出数组合理位置了。

使用 for 和 next 遍历数组

```php
<?php
//需求：使用for循环和next()函数，遍历以下数组（输出其下标和对应值）：
$arr4 = array(1=>3,  'a1'=>11,  3=>5,  "mn"=>18,  88=>2  );
$len = count($arr4);    //取得数组长度
for($i = 0; $i < $len; ++$i){    //控制循环的次数
    $key = key($arr4);      //取得“当前项”的键
    $value = current($arr4);    //取得“当前项”的值
    echo "<br />$key => $value ";
    next($arr4);    //当对“当前项”的数据处理完毕，就将指针后移一位
}
?>
```

结果：

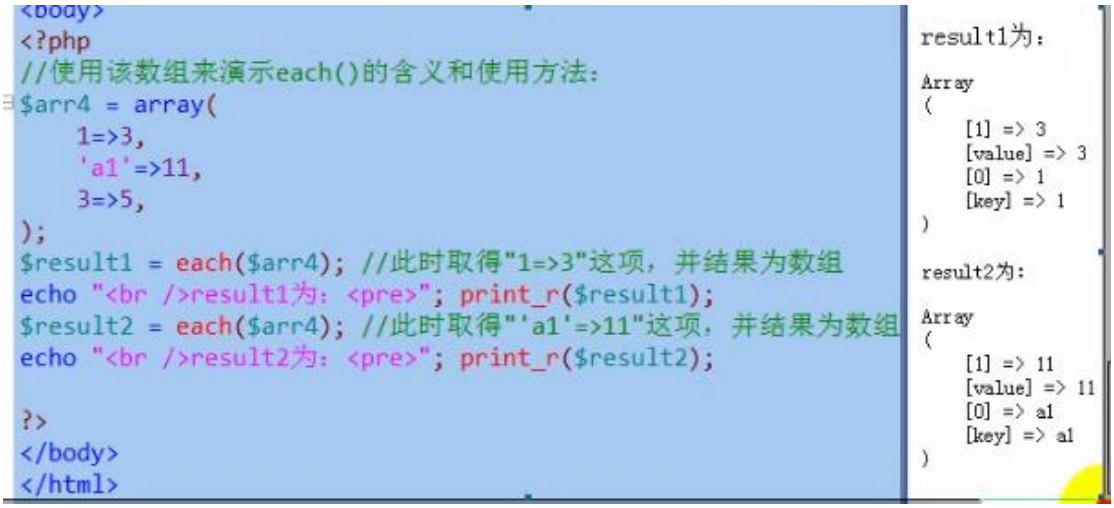
```
1 => 3
a1 => 11
3 => 5
mn => 18
88 => 2
```

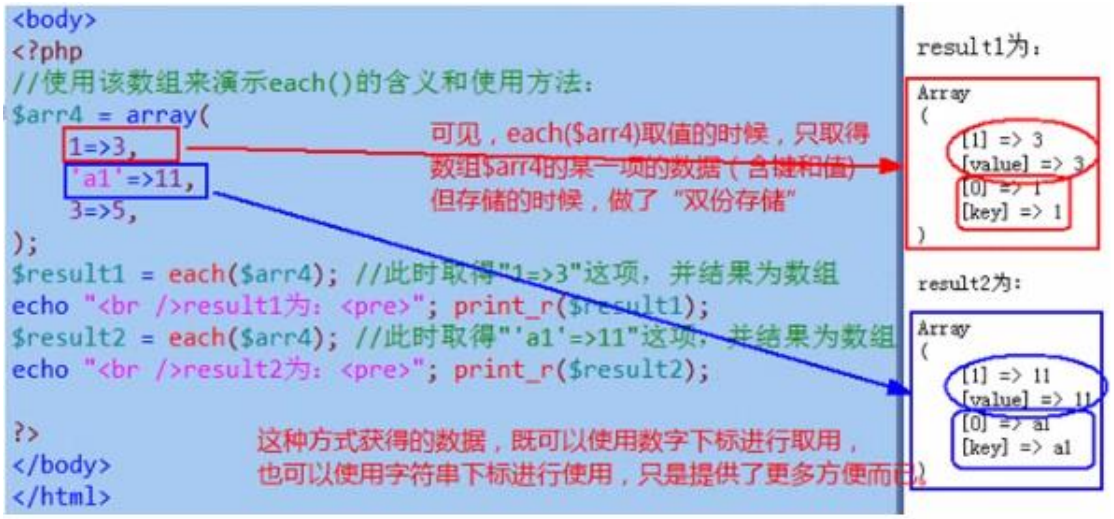
each()函数的使用：

eache()函数的作用：先去的一个数组的“当前单元”的下标和值（并放入一个数组），然后将指针移到下一个单元。

使用形式：

```
$a  =  each($数组名); //此时$a 就是一个数组了
```





list ()函数的使用：

while+each()+list()遍历数组